# UNIT 1

## Introduction to Programming :

Program is defined as set of instructions. We programmers give instructions to system to fulfill the requirement of users, is called programming. We use computer languages to give instructions to computer (for programming). The computer languages are classified into two types.

Low level languages
High level languages

The languages that are capable of accessing processor features directly are called low level languages. w.r.t. systems, it is said that system understands low level languages.

But low level languages have a problem that a program designed in LLL for a specific modal processor cannot be executed on other modal processors because of change in features. This is called hardware dependency.

To overcome this problem, high level languages are developed. But HLL cannot access processor features directly, (w.r.t. system it is said system doesn't understand HLL). So they cannot be executed directly. For execution the code written in HLL must be converted into LL code/ machine code using compilers or interpreters.

Compilers, interpreters are software mechanisms used to convert HL code into PROCESSOR APPROPRIATE LL code.

The compilers convert HL code into LL code at once and stores as a file on hard disk. The interpreters convert HL code into LL code line after line and pass into processor for execution. They won't create a file. When compared compilers are faster than interpreters.

The languages C, C++, Java etc are provided with two phases of conversion of HL code into LL code to provide SECURITY TO HL CODE. The C has two compilers and the java has one compiler and one interpreter.

## Runtime Environment:

Every SW has its own environment, and the environment of programming languages is called RUNTIME ENVIRONMENT. (In short called RE). The RE of languages include no. of components using which we give instructions to system and under control of this RE the programs executes. The RE components are also called TOKENS of that language.

---

## Introduction to Object Oriented Programming, Objects and Classes

All high level languages support user defined functions. The user defined functions are used to increase modularity in the program. When we develop a project, it might include thousands of lines of code. As it becomes difficult of typing entire program in one function, all high level languages support dividing programs into functions.

General scope of variables in programming while using functions is such that, we can declare variables either globally or locally.

The variables declared with in a function definition are called local variables to that function and they are accessible only within that function. The variables declared outside of all functions are called global variables and they are accessible to all the functions from the location of their declaration.

But as global variables are accessible in all the functions, there is a chance of misuse of them, which leads to data corruption. So it is suggested to programmers to reduce usage of global variables. In this case, we will be using only local variables, but local variables are not accessible in other functions. They have their SCOPE only within that function in which they are declared. So functions provide a facility of passing values from one function into another through ( ) as arguments from function call into function definition. But as there is a facility of passing arguments there is a chance of passing wrong argument, which leads to data corruption.

To provide SECURITY to data, a new technique of programming is designed called OOPS (Object Oriented Programming System). In this programming system, the variables and functions are clubbed into one entity called object and when a function is called for an object, the function automatically processes the variables with which it is clubbed with. And there is no requirement of passing as arguments.

In this scenario, as there is no requirement of passing arguments, there is no chance of passing wrong argument. Thus the OOPS provide data security.

In structured programming and previous generation of languages, the concentration was only on functions. But in OOPS, the concentration is more focused on data and also on methods.

Moreover, the objects in the program are treated as practical live objects. We can understand and manipulate them as a real world object. We can treat and interact with them as real world entity. So this approach of programming will have more simple, practical.

To work with OOP languages, one should have knowledge of two things.

1. Class

2. Object

Class is an ADT (Abstract Data Type). Simply, it is a user defined data type. It is used to group the variables and functions that are required for an object. It is like a template of object or a blue print of object.

Object is a named instance of class. It is like a variable of a primitive data type. When we create an object to a class all the features of the class are abstracted to the object. We can make use of the features declared in class through object with a "." DOT operator called PERIOD operator.

Java is dynamic, i.e. each object in java must be allocated with memory with the keyword "new". When we declare a reference for class, it cannot be used directly without instantiation (created with instance i.e. allocated with memory). When the reference is instantiated, then it becomes object.

Q. give an introduction to oops, classes and objects

---

**Characteristics of OOPS :**

OOPS :
To work with OOP languages, one should have knowledge of two things.

1. Class

2. Object

Class is an ADT (Abstract Data Type). Simply, it is a user defined data type. It is used to group the variables and functions that are required for an object. It is like a template of object or a blue print of object.

Object is a named instance of class. It is like a variable of a primitive data type. When we create an object to a class all the features of the class are abstracted

to the object. We can make use of the features declared in class through object with a "." DOT operator called PERIOD operator.

A language to be an OOP, should have some standards and are called oops concepts. They are:

Abstraction : Abstracting features of class into object in different ways.

Encapsulation: Selective hiding of class members in object

Polymorphism : One thing in many forms

Inheritance: Getting features of a class into another.

Different benefits of oops achieved by these concepts are better memory management, high data security, easy identification of methods that process data, reusability of code etc.

By the above benefits, it becomes easy of managing large projects with OOPS. And in OOP approach, as we treat the objects as real world entities, it becomes easy in understanding them and interacting with them.

Questions that can be framed:

Q. What are basic concepts of OOPS ?

Q. What are the characteristics of OOPS?

---

**Difference between OOP and Procedure Oriented Programming:**

Procedural Programming can be defined as a programming model which is derived from structured programming, based upon the concept of calling procedure. Procedures, also known as routines, subroutines or functions, simply consist of a series of computational steps to be carried out. During a program's execution, any given procedure might be called at any point, including by other procedures or itself.

Object oriented programming can be defined as a programming model which is based upon the concept of objects. Objects contain data in the form of attributes and code in the form of methods. In object oriented programming,

computer programs are designed using the concept of objects that interact with real world. Object oriented programming languages are various but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types.

| Procedural Oriented Programming | Object Oriented Programming |
|---|---|
| In procedural programming, program is divided into small parts called ***functions***. | In object oriented programming, program is divided into small parts called ***objects***. |
| Procedural programming follows ***top down approach***. | Object oriented programming follows ***bottom up approach***. |
| There is no access specifier in procedural programming. | Object oriented programming have access specifiers like private, public, protected etc. |
| Adding new data and function is not easy. | Adding new data and function is easy. |
| Procedural programming does not have any proper way for hiding data so it is ***less secure***. | Object oriented programming provides data hiding so it is ***more secure***. |
| In procedural programming, overloading is not possible. | Overloading is possible in object oriented programming. |
| In procedural programming, function is more important than data. | In object oriented programming, data is more important than function. |
| Procedural programming is based on ***unreal world***. | Object oriented programming is based on ***real world***. |

Q. write differences between procedure oriented languages and oops.

---

# Introduction to Java Programming

Introduction :

Java language is developed by a team under leadership of James Goslings at SUN Micro Systems (Stanford University Networks). It was actually developed as electronic programming language and called OAK. It was used in set-top boxes, remotes etc. it was released during mid of 1991. Later it is reconstructed for systems and released in the year 1995 with name "java". In French, "JAVA" means "multi-purpose". The java is a general purpose programming language and also it is constructed as a platform (mini operating system).
The general purpose programming edition is called J2SE (Java 2 Standard Edition) and the other platforms released are J2ME (mobile edition) and J2EE (Enterprise Edition).

The Java has become more popular because of its features. Its features are generally referred as java BUZZ words.

**Object Oriented** − In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

**Platform Independent** – the java program when compiled, it generates byte code. The byte code generated on any OS can be executed on any OS. Because of this byte code the java programs are WORA (Write Once Run Anywhere) OR compile once run anywhere or platform independent.

**Simple** – As Java syntaxes and semantics are similar to C, C++ etc languages, it is simple to learn and write programs.

**Secure** − With Java's secure feature it enables to develop virus-free, tamper-free systems. It is managed by the Byte Code Verifier of the JVM.

**Portable** – as java is platform independent and as the byte code can be interpreted on any platform, it is portable to any system.

**Robust** – it means healthy. Because of the exception handling mechanism provided by the Java Debugger of JVM, the java programs are healthy and strong.

**Multithreaded** − With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously.

**Distributed** − Java as it is platform independent it can be distributed to any system on the internet.

**Dynamic** − Java is considered to be more dynamic than C or C++, as it has the best memory management systems.

Questions that can be framed :

Q. give a brief introduction to java
Q. write about java features
Q. write about java buzz words

---

Comparision of java with other languages:

**Java** is one of the most popular and widely used programming language and platform. A platform is an environment that helps to develop and run programs written in any programming language.
Java is fast, reliable and secure. From desktop to web applications, scientific supercomputers to gaming consoles, cell phones to the Internet, Java is used in every nook and corner.

Here we are comparing 3 other languages (Python, C++ and C) with Java.

**PYTHON**

- Python is a high-level language. It fully supports object-oriented programming. Python is not a pure object-oriented language.
- Python is an interpreted language whereas Java is not an interpreted language, it is a compiled language.
- Python is a scripting language whereas JAVA is a low-level implementation language.
- Python is easy to use whereas JAVA is not as simple as Python. Programmers prefer to use python instead of Java because python contains less line of code whereas Java is just opposite to it.
- Python programs are much shorter than JAVA programs.
- Many large organizations like Google, Yahoo, NASA, etc. are making use of Python. But Python programs are generally expected to run slower than Java programs.
- Java has much better library support for some use cases than Python which is the biggest advantage of JAVA.
- Python is very much slower than Java.

**C++**

- Java was basically derived from C++.
- C++ is both procedural and object-oriented programming language whereas Java is a pure object-oriented language.
- Both the languages have different objectives, i.e. their purpose of use are different.
- The main objective of C++ is to design a system of programming.
- Java doesn’t support operator overloading but C++ does support it.

- C++ also extends the C programming language whereas Java is basically created to support network computing.
- Java is much slower than C++ in terms of execution.
- In Java, there is an automatic garbage collection whereas this is not the case in C++. In C++ all objects are destroyed manually with the help of the code.
- C++ supports pointers which are variables which store addresses of other variables. But Java does not have any kind of variable which stores addresses of other variables.
- C++ executes its programs very fast compared to Java.

**C**

- C is very much like C++(which was used to derive Java). In fact, C++ is an updated form of C.
- C is a structure or procedure-oriented language whereas Java is an object-oriented programming language.
- Execution time for programs written in C is very less when compared to Java.
- C supports pointers whereas Java does not support variables for storing addresses of other variables.
- C cannot handle exceptions in its program whereas Java is very good at handling exceptions

Questions that can be framed:

Q. Give comparisons between java and other languages

---

Applications and Applets:

**Java Application:**

Java Application is just like a Java program that runs on an underlying operating system with the support of a virtual machine. It is also known as an **application program**. The graphical user interface is not necessary to execute the java applications, it can be run with or without it.

Java Applet**:**

An applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. An applet is embedded in an HTML page using the **APPLET** or **OBJECT** tag and hosted on a web server. Applets are used to make the web site more dynamic and entertaining.

---

Java Development Kit:

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop java applications and applets. It contains JRE + development tools.

When the java was released by SUN micro systems, the software was named J2SDK (Java 2 Standard Development Kit). But later it is called JDK, and the JDK is available in different editions for different purposes.

- Standard Edition (general edition for system programming)
- Enterprise Edition ( for enterprise web application)
- Micro Edition ( for mobile programming)

The JDK when installed, it installs an API and a JVM.

The java API is divided into packages and sub packages that need to be imported to each program to make use of its classes.

The JVM is Java Virtual Machine. The JVM acts as a mini operating system. When we install java into system, we must install the JDK the Java Development Kit. It also installs a public JRE that plugs into the browsers for running java applets. The JDK is designed for each operating system separately.
When we install the JDK into system, the JRE is installed and it includes two components viz., API and JVM. The JVMs are designed separately for each OS and kept within the JDK software. The JVM designed for windows would not work on UNIX or other operating systems. So we must download appropriate JDK and install into system.
The "javac" (java compiler) converts high level code into byte code that is JVM dependent. And the JVM further converts the BYTE code into Operating System's native code. Thus the JVM makes java **PLATFORM INDEPENDENT**.
The JVM is a software mechanism that includes many layers like byte code verifier, java debugger, interpreter and a compiler called Just In Time compiler.
**Byte Code Verifier**:
The java compiler when creates byte code file, it places a signature at beginning of the byte code file, called magic number. The byte code verifier of the JVM checks for this magic number and if it exists, then only it allows the byte code to enter into JVM for further process. Thus, the byte code verifier providing a firewall mechanism to provide security to system. Because of this the java is **SECURED**.
**Java Debugger**:

The java debugger is the next layer of JVM that checks for logical bugs in the program with the help of a software mechanism called exception handling. This JDB checks for unhandled exceptions OR runtime exceptions. Because of this, the java programs become **ROBUST**.

**Interpreter**: as the java applet programs are placed on a web server and executed on client and as the clients can have any operating system, the JVM has an interpreter to convert the byte code into required operating system's matching code called native code where ever , when ever required.
**JIT compiler**: many a times, the interpreter works slow when compared to compiler, so the JVM is integrated with a compiler called "Just In Time" compiler, that runs by default. It can be switched off, to convert byte code into native code with interpreter.
Thus the JVM converts the byte code into native code for execution by providing security and making java programs robust and platform independent.

Questions that can be framed :

Q. write about JDK
Q. what are the different components installed when JDK is installed
Q. write about the JVM

---

Java Source file structure, Prerequisites for Compiling and Running Java Programs:

Java is pure oops, so everything is placed within a class, including main() definition. No global code is allowed in java. The general structure of java program is similar to a C or C++ program.

> Documentation Section

> Package Statement

> Import Statements

> Class Definition

The document section specifies what the program does and why it is useful. It is optional part. It is placed in comments.
The package statement is used to place our classes in user defined packages.
The import statement is used import required package into program

The interfaces are design by contract. And it is optional part.
The class definition includes the actual code of program. It may have no. of classes required to fulfill the requirement of users. Generally we place main() defined in a separate class that acts as controller, by declaring objects of other classes in main().
Ex:

```
/*
This is document section
This is the first java program
*/
//package statement (optional)
//import statement (optional to this program)
class Test1{
public static void main(String []a){
System.out.println("hello java world");
}
}
```

Once program is written, it is saved as a file. The java naming convention is that we save the program with a name that of the class name. i.e. the file name and the class name are same.

The program is compiled at command prompt with the command "javac".
Syntax: javac <file_name>.java
The "javac" activates java compiler that converts high level code into intermediate code and saves with extension " .class" called BYTE code. The javac generates number of byte code files each for a class in the program.

The program is executed by activating JVM with command "java" at command prompt.
Syntax : java <ClassName>

** if file name and class name are same, the we give same name to "javac" and "java". If different then we must pay concentration on file name and class name.

**we must set PATH for "javac" command with OS to make use of it.

The java when installed, it gets installed into "C:\Program Files \Java\jdk_x.y\bin"

copy this path and then… get onto my computer -> properties -> advanced TAB -> click on Environment Variables -> select PATH in system variables-> click on

EDIT button, and in “value” field place a ; at end and paste the above copied path and click on all OK buttons.

**Questions that can be framed**:
Explain program structure of simple java program.
Explain how to implement / run a java program.
Q. What are the pre-requisites for running a java program?

---

More Complex Programs,

UNIT II

Java Language Fundamentals:

The building Blocks of Java: every language has its own set of components defined in runtime environment, using which we develop programs, called building blocks of the language.

The different JRE components or building blocks or LEXICAL tokens of java are:

Comments:

The Java language has three styles of comments:

// text

All characters from // to the end of the line are ignored.

/* text */

All characters from /* to */ are ignored.

/** text */

These comments are treated specially when they occur immediately before any declaration. They should not be used any other place in the code. These comments indicate that the enclosed text should be included in automatically generated documentation as a description of the declared item.

Identifiers : are names used for identification of components like variables, methods, classes, objects etc. the java has some naming rules for identifiers :

- Identifier must start with alphabet
- They may have numbers in between or at end
- Keywords are not supposed to be used
- Duplicate names not permitted
- Digits, underscore, currency symbols are allowed
- Java is case sensitive, so identifiers are also case sensitive
- The built in package / class names should not be used as identifiers.

Keywords: every language has its own set of important words reserved for a specific purpose called keywords. The java has 49 keywords.

Ex: int , char, float etc

Literals: Literal is a value, that is processed in the program. The java supports different types of literals like, integer literals, floating point literals, Boolean literals, character literals, string literals.

The integer literals are of 3 types:

decimal values (base 10) .. the general numbers what we use. Ex: 15

octal values (base 8) we use 0-7 numbers and the value is prefixed with a ZERO

ex: System.out.println(051) : will display 41

hexa-decimal values (base 16) we use 0 – 9 and then A-F , i.e.0-F all sixteen digits are used to represent a value in hexa decimal format. This value is prefixed with 0X to the value.

ex: System.out.println(0x64) : will display 100

Floating point literals: are fractional parts and can be represented as floats or doubles. The JREs default data type for floating points is double. To represent a value as float, we must suffix the letter f or F to the value to represent it as float data type value.

Character literals : are prefixed with a forward slash ( generally called escape sequences)

\\ : display back slash

\" : display "

\r :return carriage (takes cursor to beginning of line)

\b : back space

\n : new line …

Etc

String literals : String is group of characters. It si represented within " "

Ex : "hello"

Boolean literals : unlike in C and C++, where the Boolean values are represented with pulse / no pulse ( zero or non-zero) , the java has keywords true / false to represent Boolean literals.

---

Q. write about java building blocks.

**Data types**: The data types are used to provide proper treatment to data. The java supports both primitive data types and non-primitive data types (user defined).

The different primitive data types supported by java are byte, char, short, int, long, float, double and boolean.

byte: it occupies 1 byte memory in RAM and stores ASCII code of key pressed. It is equivalent to char data type of C.

char: it occupies 2 byte memory in RAM and stores and manages data in Unicode format. (this is not like char data type of C and C++)

short, int and long: these three are used to store and manage integer data. Where the short occupies 2 bytes, int occupies 4 bytes and long occupies 8 bytes in RAM. Though long occupies 8 bytes, it stores only whole numbers only.

float: it occupies 8 bytes memory used to store real numbers. i.e. fractional part also. When we specify a float value in java, the value must be suffixed with a letter F/f to the value. As JRE's default data type for real numbers is double, when we write float x=1.5; it raises error, as 1.5 by default is treated as double. So we must write float x=1.5F; or float x=1.5f;

double: it occupies 16 bytes memory. Used to store real numbers.

boolean: In C and C++ the booleans are managed with PULSE or NO-PULSE i.e. with ZERO or ONE, but the java works with booleans with keywords true / false directly. In java, we cannot use 0/1 in logics, we must use the keywords true/false. Ex : boolean b=true;

Variable declaration:

A variable is a named space reserved in main memory, used to store data given input by user. These are used for manipulation of data.

The variables can be declared with in a method definition or in a class or as a static variable in a class.

The variables declared with in a method definition are called local variables to that method and they have SCOPE only within that method. They are not accessible in other methods of the class. The local variables of a method cannot be static. And we must initialize them explicitly.

The variables declared with in a class are accessible to all the methods of that class and called instance variables. The instance variables hold the values for each instance of the class. The instance variables have their scope to all the methods of that class.

The static variables are allocated with memory once and shared by all objects. They have common copy in memory. The static variables cannot be declared with in a method. They can be declared only in a class. These will have their scope even if the object is not created.

The variables can be declared to any above data type as it is done in C or C++.

Syntax: data_type variable_name;

Ex: int a;
boolean b; etc.

The variables can be assigned with values either at the time of declaration or after declaration.

Ex: int a;
a=5;
or
int a=;

---

Q Write about data type of java

Q. write about primitive data types

Q. write about variable declarations in java

---

**Wrapper classes**:

When we take input of data from user, we declare variables of primitive data types and store data into them to do process on them. As java is pure oops, many a times we need to treat data as objects.

To treat data as objects, the java.lang package has number of classes each for a primitive data type, called wrapper classes.

The data type byte has a class Byte, char – Character, short-Short, int- Integer, long – Long, float – Float , double – Double and the boolean has class Boolean.

For example, if we write int a=5; now the data value 5 is treated as a simple data value. But if we write Integer x= new Integer(5), the same 5 is treated as an object.

The java.util package has number of utility classes for easy manipulation of data. all the utility classes called collections store and manage data as objects. So the simple data value is to be converted into objects and stored into these collection classes.

The simple data value when converted into objects, it is called boxing and when the same value is converted from object into a simple data type value, it is called un-boxing.

In older versions of java, the boxing – un-boxing was done explicitly. But the current versions of java, the auto-boxing is done. i.e. the boxing is done automatically.

When we store a simple data value 5 into a collection framework class, it is automatically converted into its corresponding wrapper class object. The un-boxing is still to be done explicitly.

The wrapper classes also have many static methods for data conversions from one data type into another and also from numeric into String and vice-versa.

The methods parseTYPE() where the TYPE is replaced with appropriate data type converts String data into numeric and the toString() converts numeric into String.

---

Questions that can be framed:

- ➔ Write about wrapper classes.
- ➔ What are the wrapper classes and why they are used
- ➔ What is boxing / un-boxing

---

**Operators and Assignment**:

Operators are used to do process on data. The operators are classified in different ways based on no. of operands (on which operation is performed) they take like unary operators, binary operators and ternary operators.

The operators that take one operand are called unary operators. The operators that take two operands are called binary operators. They are further classified into different types based on their purpose.

- ➔ Arithmetic Operators : the operators that are used to perform basic arithmetic operations are called arithmetic operators. They are :
  +, -, *, / and %
  As the division results two values, the quotient is retrieved with / operator and the reminder with % symbol called modulus.

** the modulus can be used only with int data type.

- ➔ Relational Operators: these are also called comparison operators. These are used to develop conditions and return true or false, called Boolean values named after the mathematician called George Bool. They are :
  >, >=, <, <=, ==, !=
  The == is called equality operator and the != is not equals to.

- ➔ Logical Operators: these are used to combine two conditions. These are:
  &&, || and !
  The && is logical AND operator, || is logical OR and the ! is NOT. The && results false even if one is false and the || results true even if one is true. The ! operator turns true into false and vice-versa.

- ➔ Assignment operator: this operator assigns RHS to LHS, so LHS must be a variable. It is '='.
  All arithmetic operators in combination with assignment operator performs dual operations like a=a+2; can be written as a+=2; we can use +=, -=, *=, /= and %=.

- ➔ Increment decrement operators: these are actually unary operators. They are ++ ,- - and -.
  The – negates the value.
  The ++ increases value by 1 and
  the - - decreases value by 1.

  These can be used either as pre-process increment/decrement or post process increment/decrement.
- ➔ Conditional operator : the languages C, C++ and java supports an operator
  ' ? : ' called conditional operator and it is a ternary operator. i.e. it takes three operands.
  Syntax:

Condition ? statement1 : statement2;

If the condition results true, statement1 is executed by skipping statement2 and vice-versa.

For simpler divisions of program into two parts, we prefer using a conditional operator instead of if-else branching as branching gives compiler overhead (burden on system/RE).

➔ Bitwise Operators: these perform operations on binary digits of the data. These can be used with integer or byte data types in java.

& - Binary AND
| - Binary OR
^ - Binary XOR (exclusive OR)
~ - Binary complement operator
<< - Binary left shift
>> - Binary right shift

➔ Special Operators: Java has some special operators like,
. (DOT) : this is called PERIOD operator. It is member access operator in java. The members of object are accessed through object using this period operator. Ex : obj.feature.

instanceof : this operator is used to check whether the object is of given class type or not. This operator returns true or false.

new: this operator is used to allocated memory for each instance dynamically.

---

Questions that can be framed:
Write about different operators in java.

---

**Control structures**:

The control structures are used for controlled execution of code. The different control structures in java are : branching using if-else and switch case, and iteration using loops.

The if-else can be used in different ways:

➔ Simple if : the if is a keyword that accepts a condition/Boolean expression for decision making and executes the influenced statement if the condition results true. The else part is optional part. We can use only simple if without else.

```
syntax :    if(condition)
            Statement ;
```

➔ If-else statement : we can also have an else part placed with if keyword. The statement under else part executes if the Boolean expression placed with "if" results false.

```
Syntax:     if(condition)
                Statement;
            else
                Statement;
```

If we need to place more than one statement under influence of any part, then we should group them using braces.

➔ Nesting if-else statement: placing of if or else with in another if or else is called nested if-else statement. If we need a condition to be checked based on the other if condition then we can place if within if, and it is called nested if statement.

➔ If-else-if ladder: When we need to divide program into more than two parts and execute only one among them, then we use the if-else-if ladder.

```
if(condition)
   Statement;
else if (condition)
   statement;
else
   statement;
```

Iteration :

The loops are used for iteration. Iteration is repeated execution of process. The loops are controlled with a Boolean expression. If the Boolean expression results true, the loops continue its execution and otherwise stops.

Based on placement of the condition, the loops are classified into two types. i) entry controlled loops, ii) exit controlled loops. The loops that have Boolean expression at end of loop body are called entry controlled loops and that have at end of loop body are called exit controlled loops.

While statement:

Syntax :

```
Initialization;
While(boolen expression)
{
        Statements;
        Counter;
}
```

Do-while : When we use the do-while, the loop body is executed at least once even if the Boolean expression results false.

Syntax :

```
Initialization;
do{
        statements;
        counter;
}while(Boolean expression) ;
```

for statement:

the for is entry controlled loop and it takes the Boolean expression at beginning of loop body.

Syntax :

```
Initialization;
for(  ; Boolean expression;  )
{
        Statements;
        Counter;
}
```

As in for loop, the program control comes to the location that is before first ; we can logically shift initialization to that location and as every time the program control comes to the location that is after second ; we can logically shift the counter to that location, thus forming logical syntax of for as follows:

```
for(initialization ; condition ; counter)
        Statement;
```

For - each loop:

The for in java takes another form also. It is generally used with arrays etc.

```
for ( type var : array) {
        code to be executed
}
```

Ex:

```
int arr[]={12,23,44,56,78};

for(int i:arr){
System.out.println(i);
}
```

Switch statement:

For decision making, the java supports switch-case statement also. The switch keyword

takes a variable and there can be no. of cases grouped with in braces under a switch.

The switch checks for EQUALITY of the variable against multiple values. The block under switch executes completely from matching case till end of block. So for controlling, we may place a “break” keyword after statements under each case.

The switch-case in C,C++ work with ASCII code. So a switch can have maximum 256 cases in C and C++. But in java, the switch works with byte, short, int, long data types and from the version of java 7, it works with String objects also. In java’s switch-case we can use WRAPPER classes also.

Syntax :

```
Switch(var)
{
        case 1: statements;
                        break;
        case 2: statements;
                        break;
                ……..
                ……..
        case n: statements;
                        break;
        Default :       statements;

}
```

---

Q. write about control structures in java (don’t write about switch-case)

Q. write about controlling constructs (don’t write about switch-case)

OR they may ask only branching or only switch case or only iteration for 5 marks

---

**Arrays**:

Array is arranged things. The things are arranged for easy and faster manipulation. The array is collection of similar type of elements that has contiguous memory location.

The array is indexed collection of elements. Where the first element is indexed with ZERO and the last element is indexed with its size-1.

The “[ ]” are used to declare and manipulate arrays. The [ ] are used to specify “number of elements in array / size of array” while declaring, and the same [ ] are used to specify index number of each element.

The java arrays are dynamic and treated as objects in java. When an array is declared in java, it is allocated with an extra integer variable memory called “length” that stores the size of array and can be used with a DOT operator with array name.

In java array declaration can be done in following ways:

Data_type []  arr_name; ([] near data type)

Data_type []arr_name; ([] near array name)
Data_type arr_name[]; ([] after aray name)

While declaring the array the size is not specified. The size of array is specified while allocating memory with new operator.

Ex: int [] a=new int[5];

In java the arrays can have primitive data type values and also objects as well.

The arrays can be initialized while declaring by specifying the values with in { } directly without specifying array size, as below:

int a[]={10,20,30,40};
int a[]=new int[]{10,20,30,40,50};
int a[]=new int[5]{10,20,30,40,50}; this code raises error.

Ex:

```
int a[]=new int[]{10,20,30,40,50};
for( int i=0;i<a.length;i++)
        System.out.println(a[i]);
```

In java, the “for–each” loop can be used for traversing the across array. The first in for-each is a variable of data type of array and the second is array /collection name, where each value in the array is copied into the variable and the variable is manipulated while iterating. But as we cannot traverse in reverse in for-each loop, it is fewer used.

Ex:

```
        int arr[]={12,13,14,44};
        int total=0;
        for(int i:arr)
                total=total+i;
        System.out.println("Total: "+total);
```

We can also pass an array as argument to methods :

```
class Testarray2{
//creating a method which receives an array as a parameter
static void min(int arr[]){
int min=arr[0];
for(int i=1;i<arr.length;i++)
         if(min>arr[i])
                min=arr[i];
System.out.println(min);
}
public static void main(String args[]){
int a[]={33,3,4,5};//declaring and initializing an array
min(a);//passing array to method
```

}}

The disadvantage with arrays is that, we can store only fixed size of elements in array. Its size is not increased during runtime of program.

---

Questions that can be framed:

- ➔ White about single dimensional arrays of java
- ➔ How to create and manipulate arrays

---

**Two dimensional arrays**:

In many programming languages, the arrays support managing data by providing more than one index referring, called multi dimensional arrays. In practical life, the two dimensional arrays are used much.

A 2D array can be declared in following ways:

```
Data_type[][]   arr;
Data_type  [][]arr;
Data_type    arr[][];
Data_type  []arr[];
```

Ex:

int arr[][]=new int[3][3]; though the memory is allocated contiguously, we can logically feel this as a 3 row and 3 column array.

We can initialize a multi dimensional array by passing values as groups within { }.

```
int arr[][]=new int[][]{ {1,2,3,4},{10,20,30,40},{100,200,300,400}};
```

it can be manipulated as below:

```
for(int i=0; i < arr.length ; i++)
{
        for(int j=0; j < arr[i].length ; j++)
                System.out.print(arr[i][j]);
        System.out.println();
}
```

Jagged arrays : the java supports jagged arrays while working with multi-dimensional arrays. The JAGGED ARRAY can hve different number of columns in each row.

Ex:

int arr [][] =new int[5] [ ]; // the 2nd dimension is not specified here.

```
arr[0]=new int[5];
arr[1]=new int[8];
arr[2]=new int[3];
arr[3]=new int[7];
arr[4]=new int[4];
```

it can be used when we need to store different number of columns for each row.

---

Questions that can be framed:

- ➔ write about 2 D arrays in java
- ➔ write about multi dimensional arrays
- ➔ write about jagged arrays in java

---

**Strings**: In programming languages the sequence of characters is called a String. In C and C++, the char data type arrays are used to store and manage strings. In java, the java.lang package has classes String, StringBuffer and StringBuilder to store and manage string data.

The String objects are managed in a special memory location called STRING POOL within the memory. The string pool would not store duplicate values within itself. When the same value is stored in multiple string objects, the data is stored in string pool once and all String references point to the same data in string pool, thus providing better memory management.

When multiple string objects point to same data, if data modified through object then the data of other objects also get modified and it leads to data corruption. So the String objects are designed as IMMUTABLE (NOT MODIFIABLE with in itself).

The StringBuffer, StringBuilder classes are mutable (modifiable with in itself) as their data is not managed with in string pool.

In java, when a string literal is represented within " " , it is treated as an instance of String class. So we can initialize the String object either with a new operator or simply assigning string literal to the reference directly.

Ex:

```
String s=new String("HELLO");
String s="HELLO";
```

The string class has number of constructors that take byte[], char[], string constant, string object etc to construct the string object.

The different methods of String class are :

- ➔ Concat() : this method concatenates a given string without modifying data within the object, returns data as a new String object.

Ex:
```
String s= "sri ";
System.out.println(s);//displays –sri
s.concat("gnanambica");
System.out.println(s);//still displays –sri
```

As the concat() has not modified data within the object s. As these methods return string after doing process, we can store it into another object.

```
String s1=s.concat("gnanambica");
System.out.println(s+"  "+s1);//display –sri   sri gnanambica
```

If we need to modify data within itself, then as must store data back into same object as….

s=s.concat(“gnanambica”);

- ➔ equals() : returns a Boolean value by checking equality of strings
- ➔ equalsIgnoreCase():returns a Boolean value by checking equality of strings by ignoring its case
- ➔ toLowercase() : converts string into lower case letters and returns as new string
- ➔ toUppercase() : converts string into upper case letters and returns as new string
- ➔ trim() : removes trailing blank spaces
- ➔ split(char) : splits string into String[] at the char specified.
- ➔ charAt(int index) : returns the character at given string
- ➔ startsWith() / endsWith() : for strings searching
- ➔ compareTo() : for lexicographic comparison of strings

* The String class is overloaded with operator + for string concatenation

**String Buffer Class**:

Java StringBuffer class is used to create mutable (modifiable) String objects. The StringBuffer class in Java is the same as String class except it is mutable i.e. it can be changed with in itself.

```
class StringBufferExample{
public static void main(String args[]){
StringBuffer sb=new StringBuffer("Shri ");
sb.append("Gnanambica");//now original string is changed
System.out.println(sb);//prints ShriGnanambica
}  }
```

The String class can be assigned with value directly as String s=”HELLO”; as the string literal is treated as String class instance. But in case of StringBuffer objects, it it not like so. We must explicitly allocate memory with new.

StringBuffer sb=new StringBuffer(“Hello”);

---

Questions :

- ➔ write about strings in java
- ➔ how strings are managed with in string pool
- ➔ Write about differences between String and StringBuffer classes

---

Java as an OOP Language

**Defining classes**:

Class, Objects and Methods:

Defining a class: A class is a basic building block in OOPS, to which the objects are created. A class is defined with the keyword 'class'. It is used to group the data members called fields and methods. The class describes the behavior of the object.
Syntax :
class <ClassName> {
    Fields declaration
    Method definitions
}

Field declaration:
    The instance variables or class variables are declared with in the class directly. These are initialized with ZERO by default. We can also declare variables as local variables in the methods. These must be initialized explicitly. These fields can be of primitive types or can also be objects of other classes.
Ex: int x;

Methods declaration:
    The method is collection of statements that are grouped together to perform an operation. The methods are declared in class and can be called with object. The signature of the method includes access specifier, access modifiers, return type of the method, method name and formal arguments.

The access specifier can be private, protected, public or default. We can have only one access specifier

Access modifiers can be static, abstract, final, synchronized etc. we can have multiple access modifiers to a method.

Ex: public static void main(String a[]){}

Creating object: as the java is dynamic we instantiate each object with "new" operator and assign that instance to a reference.

Syntax: ClassName reference=new ClassName();
Ex: Emp e=new Emp();

    Now all the features of the class are abstracted into object and they can be accessed with the object with a DOT operator called PERIOD.

In general, the data members are declared as private and the methods are declared as public. We call the method for an object and the method does process on the data of that object.

Constructor : constructor is a special method of the class that has same name of the class name and will be invoked automatically whenever the class is instantiated, i.e. allocated with memory with new operator.

Each class may have either one constructor or multiple constructors with arguments.

As these methods (constructors) are used to construct memory for the member references declared in a class, these are called constructors. And these can also be used for initializing the values of data members,
Though the constructors are methods, but not used for general process like normal methods, the constructors cannot return any value and these have no return type

We may call one constructor with in another of the same class by passing arguments to this(). And we can call base constructor with in derived constructor by passing arguments to super().

---

Questions that can be framed:
How class features are declared and how to access them?
How to define a class and access its features?
What are the features declared in a class and how to use them?

---

**Modifiers**: the private, protected, public keywords used for encapsulation are called access specifiers and the java has access modifiers like abstract, final, static , volatile, native, synchronized.

abstract : used to develop abstract classes and abstract methods.

An abstract method is one that has no definition.e. has no implementation and it is used to make sure the derived class provides its implementation.

Abstract class is one that cannot be instantiated generally and used as base in inheritance. The abstract classes tell what to do and the derived classes provide their implementations by overriding the abstract methods of abstract classes.

Abstract classes can have abstract methods and also non-abstract methods. But if a class has atleast one abstract method then the class must be abstract class.

static : any feature, either a variable or a method declared as static within a class is allocated with memory even if object is not created for that class. So the java provides a facility of using them with class name also.

The static variables or methods are generally called class features and the non-static features are called instance features.

When a class has static variables / objects declared, they are generally initialized in a

static block instead of constructor.

final : in C and C++, the keyword “const” is used for declaring constant variables. In java, the keyword “final” is used for declaring constants.

The fial keyword can be used with variables, methods and also witjh classes.

The final variable can not be modified with its values

The final methods can not be overridden in derived classes

The final classes can not be used in inheritance as base classes.

volatile : the variables that need to change their values frequently for number of times during running of program can be declared with keyword “volatile”. It is something like register variables of C.

synchronized : used in multi threading programming

native : when the code is written in C or C++ languages and that code need to be used in a java program, then we use the keyword “native” to specify that the code is from external resource and need to be executed in native environment.

---

Q :

Write about different access modifiers?

---

**Packages**:

When java is installed into system, there will be two components installed into system viz., JVM and Java API. The JVM is java virtual machine and the API is Application Programming Interface, simply java library.

The java API includes no. of pre-defined classes, grouped based on their purpose and placed into folders called packages.

A package in Java is used to group related classes. We use packages to avoid name conflicts, and to write a better maintainable code. Packages are divided into two categories.

➔ Built-in Packages

➔ User-defined Packages (create your own packages)

The main package in API is called “java” and it has no. of sub packages, where each package has no. of classes related for a specific purpose. These packages must be linked to our program (as the libraries are linked in C with #include ) with the keyword import.

Syntax of import :

import package_name.Class_Name; imports specific class into our program.

Ex: import java.util.Scanner;  imports Scanner class of util package into our p[rogram.

import Package_name.*;  imports all classes in that package

Ex: import java.util.*; imports all classes of util package into our program.

There are many system defined packages like:

lang: this package has classes to handle basic features of java LANGuage.

util: has general utilities like Scanner, Vector, ArrayList, Arrays, Stack, LinkedList etc classes.

Sql: this package has classes for database connectivity.

applet: for applet programming

io: has classes for managing input / output streams etc

to make use of the classes, we must import the corresponding package into our program.

User defined packages: we can also create our packages and organize our classes into them. To create a user defined package, we must place a “package” statement in first line of the program.

Ex: package mypack;

When we need to place a class into a package, then the program must have a package statement, the class must be declared as public class, and we must compile that class by using the option –d with javac command, specifying where to place the .class file.

Ex:

```
package mypack;

public class MyClass{

    public int add(int a, int b);
```

```
}
```

While compiling this class we must use the option –d with javac command.

Syntax: javac –d <path> FileName.java

Here the <path> specifies where to place generated class files/ where to create the package.

Ex: javac –d  c:\   MyClass.java

For using this class, we must import the package into program and make use of the class.

```
import mypack.MyClass;

class MyAddition{

        public static void main(String a[]){

                MyClass m=new MyClass();

                System.out.println(m.add(5,6) );

        }

}
```

We use the option –classpath with javac command while compiling this class.

Syntax :  javac –classpath <path> FileName.java

Here path specifies where to file class files / package

Ex:

Javac –classpath   c:\    MyAddition.java

While running this program, we use the option –cp with java command specifying the location of both class files, i.e class of the package and also current folder. Where the current folder is specified with a “.” (DOT).

Syntax : java –cp <paths> ClassName

Ex:  java –cp  .; c:\  MyAddition

Once a package is created, we can add any number of classes to it. But all the classes must be public and saved with classname.

If a class placed within a package is not public, then that class cannot be used outside of the package and they are called hidden classes.

---

Q :

Write about packages in java

Write about user defined packages and its uses

---

**Interfaces**:

An interface in java is like a pure abstract class. The interface is defined like a class but with keyword “interface” instead of “class”.

The interface methods are by default abstract. So the interfaces cannot have non-abstract methods. The interface variables are by default static and final. So the values assigned to them are not modifiable.

These are concrete by design. These provide information of what to be done in derived classes. The derived classes will have implementation specifying how it is to be done.

The interface will have method declarations only. These cannot have definitions / processes to be done. The implementing sub class of interface will have to provide the definition / how the process is to be implemented.

Syntax:

```
interface  <interface_name> {
        variables;
        method declarations;
}
```

ex:

```
interface abc{
        int x=10;
        public void disp();
}
```

In above example the variable x declared with in the interface abc is static and final. It can be accessed with interface name like abc.x and its value can not be modified.

The method disp() doesn’t have definition. It is abstract by default. It tells what to be done in derived class. If a class implements the above interface “abc”, then that class must override the method disp() and provide the definition in that class specifying how the process is to be done.

The interfaces can not be instantiated as these are abstract. These can be used as bases in inheritance. The interfaces can be used in inheritance with the keyword “implements” instead of extends.

Ex:

class xyz implements abc{
}
Now the class class xyz provides implementation for all the methods declared with in interface abc.

In java an interface can “extends” other interface. But a class implements the interface. The interface variables can be accessed either with its derived class object or with interface name as these are static and final.

Java supports implementing multiple interfaces, thus providing multiple inheritance concept fulfilled with interfaces.

Ex:

class abc implements I1, I2{
}
Where both I1 and I2 are interfaces and now the class abc must override all the methods of both I1 and I2 interfaces.

---

Questions that can be framed:

- ➔ What are interfaces and how to use them?
- ➔ How to implement multiple inheritance in java
- ➔ Write about interfaces in java
- ➔ What are interfaces and how to access their features?

# Unit III
# Exception Handling

## Introduction:

ERRORS :

There are three types of errors in java
1.Compile-time errors
2.Run time errors
3.logical errors

COMPILE TIME ERRORS:
These are errors which prevents the code from compiling because of error in the syntax such as missing a semicolon at the end of a statement or due to missing braces, class not found, etc. These errors will be detected by java compiler and displays the error onto the screen while compiling.
Ex:

```
System.out.println(“ HELLO WORLD); // ” missing etc
```

RUN TIME ERROR:
These errors are errors which occur when the program is running. Run time errors are not detected by the java compiler. It is the JVM which detects it while the program is running.
Ex:

```
int a[]=new int[5];
System.out.println(a[5]); //the 4 is the last index. It is expressed as an exception by the JVM.
```

LOGICAL ERRORS:
These errors are due to the mistakes made by the programmer. It will not be detected by a compiler nor by the JVM. Errors may be due to wrong idea or concept used by a programmer while coding or by giving wrong data given input by user.
Ex:

```
int a,b,c;
a=sc.nextInt();
b=sc.nextInt();
c=a/b;  //here if the user gives input of 0 as b value, then the JRE could not execute code and raises error by throwing an exception
```

**questions that can be framed:**
Write about errors in java
What are the types of errors in java

**Basics of Exception Handling in Java:**

The logical errors and runtime errors when raised in program, the JRE could not execute the code and terminates abruptly the function in which it could not executed code, and this case is said that the JRE has thrown an Exception.

The java provides a software mechanism called Exception Handling to let the programmer to control program execution, such that it won't terminate abruptly.

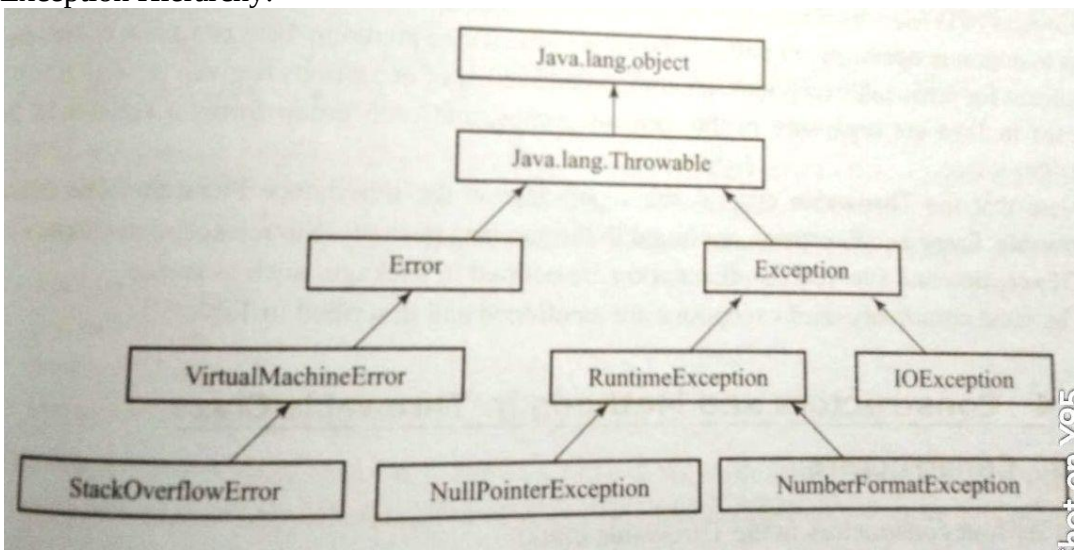
The exception handling is supported with keywords try-catch-finally, throw and throws.

Ex:

```
try{
        int x=Integer.parseInt(sc.nextLine());
        int y=Integer.parseInt(sc.nextLine());
        int z=x/y;
        System.out.println(z);
} catch(ArithmeticException ae){
        System.out.println("don't enter zeros");
}Catch(NumberFormtException ne){
        System.out.println("enter only numbers");
}
```

The finally is default block that executes in all the cases. Generally, the closing of streams, database connections etc is one with in finally block.

Exception Hierarchy:




All the classes used for exception handling are derived from the class Exception which is derived from class Throwable.

Constructors and Methods in Throwable class:

| Constructor Summary |
|---|
| Throwable ()<br>Constructs a new throwable with null as its detail message. |
| Throwable ( String message)<br>Constructs a new throwable with the specified detail message. |
| Throwable ( String message, Throwable cause)<br>Constructs a new throwable with the specified detail message and cause. |
| Throwable ( Throwable cause)<br>Constructs a new throwable with the specified cause and a detail message of (cause==null ? null : cause.toString()) (which typically contains the class and detail message of cause ). |

| | |
|---|---|
| Throwable | fillInStackTrace ()<br>Fills in the execution stack trace. |
| Throwable | getCause ()<br>Returns the cause of this throwable or null if the cause is nonexistent or unknown. |
| String | getLocalizedMessage ()<br>Creates a localized description of this throwable. |
| String | getMessage ()<br>Returns the detail message string of this throwable. |
| StackTraceElement [] | getStackTrace ()<br>Provides programmatic access to the stack trace information printed by printStackTrace() . |
| Throwable | initCause ( Throwable cause)<br>Initializes the cause of this throwable to the specified value. |
| void | printStackTrace ()<br>Prints this throwable and its backtrace to the standard error stream. |
| void | printStackTrace ( PrintStream s)<br>Prints this throwable and its backtrace to the specified print stream. |
| void | printStackTrace ( PrintWriter s)<br>Prints this throwable and its backtrace to the specified print writer. |
| void | setStackTrace ( StackTraceElement [] stackTrace)<br>Sets the stack trace elements that will be returned by getStackTrace() and printed by printStackTrace() and related methods. |
| String | toString ()<br>Returns a short description of this throwable. |

---

Q . write Throwable class
Write about constructors and methods of Throwable class
Explain basic exception handling in java with its corresponding classes

---

**Unchecked and Checked Exceptions**:

Checked exceptions: The code in java that has hardware interactivity or interactivity of other runtime environments always must be handled to achieve more perfection (Robustness). To make sure programmer to use appropriate handler, these exceptions are declared as checked / compile time exceptions. Unless these are handled, the code will not be compiled.
Runtime Exceptions: the exceptions that are thrown by JRE during running of program, based on data given input buy user or the logic implemented in program are called runtime exceptions.

The java can handle more than 600 types of pre-defined exceptions and all these are derived from java.lang.Exception class.

The different pre-defined exceptions are:

**Compile time exceptions / checked exceptions**:-
IOException : when any IO operation is performed
FileNotFoundException: when file processing code is written
SQLException: when database connectivity established
InterruptedException: when the sleep() of thread class is used
ClassNotFoundException: when a class is loaded into memory by program for further use etc situations

**Runtime exceptions / unchecked exceptions**
ArrayIndexOutOfBoundsException :when we mention the index >= length of array
StringIndexOutOfBoundsException : when we use the index >= length() of String
ArithmeticException :when division is done with ZERO
NumberFormatException : when a non-numeric value is tried to be parsed
NullPointerException : when a class reference is used prior its instantiation (allocation of memory)

---

Questions that can be framed:
What are the types of exceptions and how to handle them
Write about checked / unchecked exceptions
Write about compile time / runtime exceptions

---

**Handling Exceptions in Java**:
try-catch-finally : this block is used to handle the code, such that when an exception is thrown, the JRE customizes the process to be done with out terminating abruptly.

Syntax :

```
try{
        code expected to throw exception
}
catch(TYPEException e){
Code that need to be executed when exception is
thrown and handled
}
finally{
        default block
}
```

Here the TYPEException in catch() is replaced with appropriate exception type.

We place the code that is expected to throw exception in try block. A try must have at least one catch() or finally. But a try can have any number of catches, each handling a specific type of exception.

We place the code that is expected to throw exception in try block. A try must have at least one catch() or finally. But a try can have any number of catches, each handling a specific type of exception.

Ex:

```
try{
        int x=Integer.parseInt(sc.nextLine());
        int y=Integer.parseInt(sc.nextLine());
        int z=x/y;
        System.out.println(z);
} catch(ArithmeticException ae){
        System.out.println("don't enter zeros");
}Catch(NumberFormtException ne){
        System.out.println("enter only numbers");
}
```

The finally is default block that executes in all the cases. Generally, the closing of streams, database connections etc is one with in finally block.

Exception and Inheritance: the important thing one should remember while handling exceptions is that, while handling exceptions first child exceptions are to be handled and then the parent exceptions.

If parent exception is handled first, then because of the dynamic binding concept, as the parent reference can hold child instances, the catch with parent exception itself handles the child exceptions, which leads to raise error that the child exception is already handled.

So we must handle child exception first and then parent exceptions

---

Q. explain how the exceptions are handled in java
Explain about exception handling in java
Explain try-catch-finally blocks in exception handling

---

**Throwing User-defined Exceptions**:

The "throw" is a keyword, used to throw an exception explicitly. The "throw" can throw a pre-defined exception or any derived class from Exception including user defined exceptions.
Ex:

```
throw new ArithmeticException();
```

the throw must be placed within a try block to handle it.

User defined exceptions :
We can also develop our own exceptions by extending the java.lang.Exception class.
Ex:
UnderAgeException.java

```
public class UnderAgeException extends Exception{
	public UnderAgeException(){
		super("you are under 18. Not eligible for voting");
	}
}
-------------------------
public class Voting{
	public static void main(String args[]){
		Scanner sc=new Scanner(System.in);
		int age=sc.nextInt();
		if(age<18)
		try{
		throw new UnderAgeException();
		}catch(UnderAgeException e){
			System.out.println(e);
		}
		else
			System.out.println("Eligible for voting");
	}
}
```

The keyword "throw" is used to throw an exception explicitly. As the user defined exception is derived from Exception class, it has become throwable and can be caught like a normal pre-defined exception/
The message that must be displayed is passed as argument to Exception class constructor by calling super(), and as the toString() of Exception class returns the message as string, the Exception object can be displayed directly as System.out.println(e).

---

Questions that can be framed:
Write about user defined exceptions
Explain user defined exceptions with example
Explain how to throw a user defined exception

---

**Redirecting and Re throwing Exceptions:**

The "throws" is a keyword used along with method signature to specify that the method throws a specific exception and the programmer must provide proper handler. i.e. it is used to specify that the programmer must handle that exception with catch(), unless which the code will not be compiled.

Ex:
abc.java

```
class abc{
        public int divideData( int x, int y)  throws ArithmeticException{  //line 2
                return x/y;
        }
}
```

---

bca.java

```
class bca {
        public static void main(String args[]){
                abc a=new abc();
                try{
                System.out.println( a.add(5,0) );       //line 5
                }catch(ArithmeticException e){
                        System.out.println(e);
                }
        }
}
```

If the line at line 5 is not placed under try block, then the program will not be compiled. Thus, the "throws" at "line 2" makes sure when the add() is called, it must be tried and caught.

If the programmer doesn't have anything to do in catch(), then he may declare that the calling method also "throws" the exception, without handling.
Ex:
bca.java

```
class bca {
        public static void main(String args[]) throws ArithmeticException{
                abc a=new abc();
                System.out.println( a.add(5,0) );
        }
```

}

---

Q. what is redirecting exception
What is use of throws keyword

---

**Advantages of Exception – Handling Mechanism:**

The when exception is thrown by code in a method, it can be handled either with a try-catch block or leave it by declaring the method as it throws specific exception.
When the method is declared with "throws", the exception is not handled. For handling the code must be tried and caught. In general the programmer displays corresponding messages to user, so that the input can be given correctly into program.
But if the exception is thrown NOT by the data given input and it is because of a logical error in code, then the root cause must be identified to repair / modify the code.
For example, if "ArrayIndexOutOfBoundsException" is thrown in a location, the root cause is not in that location, it is at the location where the array is declared. To identify the root cause, so that the program can be debugged easily, we can print the stack of the exception throws and trace out the location and make correction to the code. It is done by the method printStackrace() of the Exception class.
Ex;

```
try{
        Some code
}catch(Exception e){
        e.printStackTrace();
}
```

This displays the messages of the stack that stored the root cause of current exception, by which programmer can easily debug it.

---

Questions that can be framed:
Write about debugging with exception handling.
How the exception handling can be used for debugging of program,
What are the advantages of exception handling

---

# Multithreading

An Overview of Threads :

Thread is process. When the process needs to be divided into sub processes and executed under control of programmer, that process can be made as a thread. Generally the process management is done by operating system. When process is developed as thread, the thread control is managed by programmers. As the threads won't give burden on operating system, they are called light weight components.

The thread has three properties, viz. a thread name, priority and thread group name. When we develop multiple threads, we can group them based on their purpose for easy manipulation. The threads can also be given with names to be identified uniquely in all the parts of the program. The thread priority specifies how much time the thread is to occupy the processor within given time. The priority never indicates sequence of execution of thread.

The java.lang package has a class Thread and an interface Runnable to develop threads. We can either extends Thread class or implement Runnable to develop a Thread. When the Thread is extended or Runnable is implemented, the method public void run() must be overridden, that includes the process to be executed as a thread. The run() is called explicitly by making a call to start() of Thread class. It can be blocked either by making a call to wait() or sleep().

---

Q. give an overview of threads

Explain briefly about threads

---

Creating Threads: a Thread can be created from Thread class or by implementing Runnable interface

```
class MyThread extends Thread{
    public void run(){
    }
}
```

Or

```
class MyThread implements Runnable{
```

```
    public void run(){

    }
}
```

The process that need to be executed as thread must be placed as definition of the run() method. i.e. we must override the run() method when we implement Runnable or extends Thread class.
Thread class:

Constructors:

Thread()

Thread(Thread instance)

Thread(String name)

Methods:

currentThread() : this is a static method that returns the current thread instance.

sleep() : suspends program execution for specified number of milliseconds.

setName(String) : gives a name to the thread

setPriority(int) : used to change priority of thread

run() : used to perform the action/process of the thread

start() : starts execution of thread, when start() called, the JVM calls run()

isAlive() : returns true, if the thread is alive

Exceptions in thread management:

when the sleep() / wait() is called the JRE "InterruptedException" , when a wrong argument (parameter value) is passed to any method of thread class then the "IllegalArgumentException" is thrown, and when working with multiple threads, and the threads reach a dead-lock situation, then "IllegalMonitorStateException" is thrown.

Ex:

```
class MT1 implements Runnable{
    Thread t;
    MT1()
    {
```

```
		t=new Thread(this);
	}

	public void run()
	{
		System.out.println("in MT1 :");
		for(int i=0;i<5;i++)
			System.out.print("hi ");
	}
}

class MT2
{
	public static void main(String a[]) throws Exception
	{
		System.out.println("started main thread :"+Thread.currentThread());
		MT1 m=new MT1();
		m.t.start();
		Thread.sleep(1000);
		System.out.println("\nMT1 is completed");
		System.out.println("Main  is completed");
	}
}
```

Output :

```
D:\bca>java MT2
started main thread :Thread[main,5,main]
in MT1 :
hi hi hi hi hi
MT1 is completed
Main  is completed
```

** if the main thread is not suspended with its execution with sleep(), then the main() thread completes its execution and then the user defined thread. Which leads to incorrect execution.

---

Q explain how to create threads

Explain the ways of creating threads

Explain thread execution with an example

---

Thread Life–cycle:

In Java, a thread always exists in any one of the following states. These states are:

1. New
2. Active
3. Blocked / Waiting
4. Timed Waiting
5. Terminated

New: Whenever a new thread is created, it is always in the new state. For a thread in the new state, the code has not been run yet and thus has not begun its execution.

Active: When a thread invokes the start() method, it moves from the new state to the active state. The active state contains two states within it: one is runnable, and the other is running.

- Runnable: A thread, that is ready to run is then moved to the runnable state. In the runnable state, the thread may be running or may be ready to run at any given instant of time. It is the duty of the thread scheduler to provide the thread time to run, i.e., moving the thread the running state.
  A program implementing multithreading acquires a fixed slice of time to each individual thread. Each and every thread runs for a short span of time and when that allocated time slice is over, the thread voluntarily gives up the CPU to the other thread, so that the other threads can also run for their slice of time. Whenever such a scenario occurs, all those threads that are willing to run, waiting for their turn to run, lie in the runnable state. In the runnable state, there is a queue where the threads lie.
- Running: When the thread gets the CPU, it moves from the runnable to the running state. Generally, the most common change in the state of a thread is from runnable to running and again back to runnable.

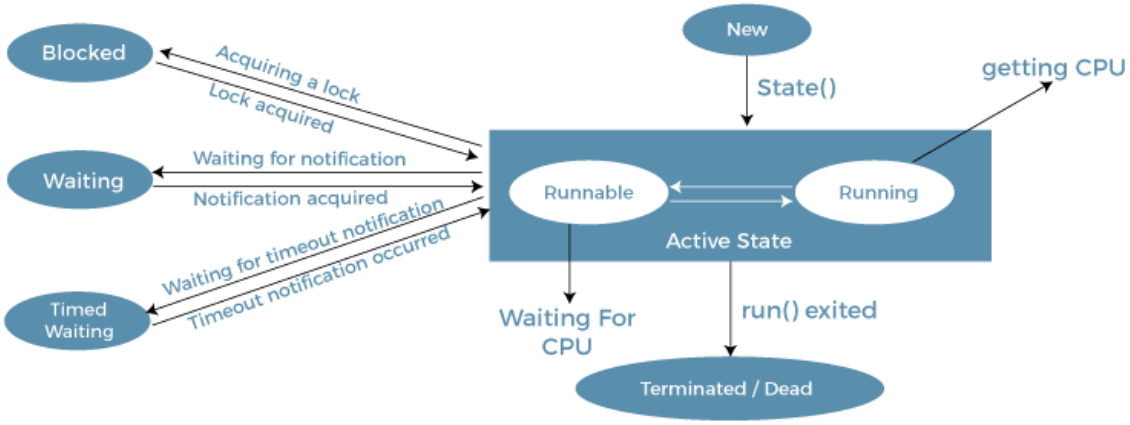
Blocked or Waiting: Whenever a thread is inactive for a span of time (not permanently) then, either the thread is in the blocked state or is in the waiting state.

Timed Waiting: Sometimes, waiting for leads to starvation. For example, a thread (its name is A) has entered the critical section of a code and is not willing to leave that critical section. In such a scenario, another thread (its name is B) has to wait forever, which leads to starvation. To avoid such scenario, a timed waiting state is given to thread B. Thus, thread lies in the waiting state for a specific span of time, and not forever. A real example of timed waiting is when we invoke the sleep() method on a specific thread. The sleep() method puts the thread in the timed wait state. After the time runs out, the thread wakes up and start its execution from when it has left earlier.

Terminated: A thread reaches the termination state because of the following reasons:

- When a thread has finished its job, then it exists or terminates normally.
- Abnormal termination: It occurs when some unusual events such as an unhandled exception or segmentation fault.

A terminated thread means the thread is no more in the system. In other words, the thread is dead, and there is no way one can respawn (active after kill) the dead thread.




Life Cycle of a Thread

---

Q. Explain thread life cycle.

---

Thread Priorities and Thread Scheduling:

Priority of a Thread (Thread Priority)

Each thread has a priority. Priorities are represented by a number between 1 and 10. In most cases, the thread scheduler schedules the threads according to their priority (known as preemptive scheduling). But it is not guaranteed because it depends on JVM specification that which scheduling it chooses. Note that not only JVM a Java programmer can also assign the priorities of a thread explicitly in a Java program.

Setter & Getter Method of Thread Priority

Let's discuss the setter and getter method of the thread priority.

public final int getPriority(): The java.lang.Thread.getPriority() method returns the priority of the given thread.

public final void setPriority(int newPriority): The java.lang.Thread.setPriority() method updates or assign the priority of the thread to newPriority. The method throws IllegalArgumentException if the value newPriority goes out of the range, which is 1 (minimum) to 10 (maximum).

The job of an operating system's thread scheduler is to determine which thread runs next. One simple implementation of the thread scheduler keeps the highest-priority thread running at all times and, if there is more than one highest-priority thread, ensures that all such threads execute for a quantum each in round-robin fashion

This means that A gets a quantum of time to run. Then B gets a quantum. Then A gets another quantum. Then B gets another quantum. This continues until one thread completes. The processor then devotes all its power to the thread that remains (unless another thread of that priority becomes ready). Next, thread C runs to completion (assuming that no higher-priority threads arrive). Threads D, E and F each execute for a quantum in round-robin fashion until they all complete execution (again assuming that no higher-priority threads arrive). This process continues until all threads run to completion.

---

Q. explain about thread priority and scheduling

---

Thread Synchronization:

When working with multiple threads, we don't prefer controlling threads with sleep() to let other threads to occupy the processor.

In this case, if one thread has not completed its execution and still occupying processor, and if other thread has completed its sleep time and tries to occupy the processor, then the processor won't respond to any of the threads and it gets hang and the JRE throws IllegalMonitorStateException. This case is called 'DEADLOCK'.

To overcome this problem of deadlock, we control threads with the methods wait() and notify() of Object class, instead of sleep() of Thread class.

But the methods called under multiple threads must be declared as 'synchronized', unless which they misbehave (execute wrongly)

Synchronization: when multiple threads try to occupy processor, i.e. when multiple threads try to do process on one shared resource, then the process goes uncontrolled. In this case, we declare the methods that are to be called by threads as synchronized, such that only one thread access the

resource at a time and the other thread is in blocked state, and when the first thread goes to blocked state, the other uses the resource.

We can either declare the methods with “synchronized” keyword, or we can write a synchronized block in current versions of java.

---

Q. explain synchronization

What is thread synchronization

---

Daemon Threads:
Daemon thread in Java is a low-priority thread that runs in the background to perform tasks such as garbage collection. Daemon thread in Java is also a service provider thread that provides services to the user thread. Its life depends on the mercy of user threads i.e. when all the user threads die, JVM terminates this thread automatically.

In simple words, we can say that it provides services to user threads for background supporting tasks. It has no role in life other than to serve user threads.

Example of Daemon Thread in Java: Garbage collection in Java (gc), finalizer, etc.
Properties of Java Daemon Thread

- They can not prevent the JVM from exiting when all the user threads finish their execution.
- JVM terminates itself when all user threads finish their execution.
- If JVM finds a running daemon thread, it terminates the thread and, after that, shutdown it. JVM does not care whether the Daemon thread is running or not.
- It is an utmost low priority thread.

Default Nature of Daemon Thread

By default, the main thread is always non-daemon but for all the remaining threads, daemon nature will be inherited from parent to child. That is, if the parent is Daemon, the child is also a Daemon and if the parent is a non-daemon, then the child is also a non-daemon.

In general, the main() method of the program is daemon thread.

---

Q. what is a daemon thread

---

Thread groups:

Java provides a convenient way to group multiple threads in a single object. In such a

way, we can suspend, resume or interrupt a group of threads by a single method call.

Java thread group is implemented by java.lang.ThreadGroup class.

A ThreadGroup represents a set of threads. A thread group can also include the other thread group. The thread group creates a tree in which every thread group except the initial thread group has a parent.

A thread is allowed to access information about its own thread group, but it cannot access the information about its thread group's parent thread group or any other thread groups.

Communication of Threads:

Inter-thread communication or Co-operation is all about allowing synchronized threads to communicate with each other.

Cooperation (Inter-thread communication) is a mechanism in which a thread is paused running in its critical section and another thread is allowed to enter (or lock) in the same critical section to be executed.It is implemented by following methods of Object class:

- wait()
- notify()
- notifyAll()

1) wait() method

The wait() method causes current thread to release the lock and wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed.

The current thread must own this object's monitor, so it must be called from the synchronized method only otherwise it will throw exception.

2) notify() method

The notify() method wakes up a single thread that is waiting on this object's monitor. If any threads are waiting on this object, one of them is chosen to be awakened. The choice is arbitrary and occurs at the discretion of the implementation.

3) notifyAll() method

Wakes up all threads that are waiting on this object's monitor.

Difference between wait and sleep?

Let's see the important differences between wait and sleep methods.

| wait() | sleep() |
|---|---|
| The wait() method releases the lock. | The sleep() method doesn't release the lock. |
| It is a method of Object class | It is a method of Thread class |
| It is the non-static method | It is the static method |
| It should be notified by notify() or notifyAll() methods | After the specified amount of time, sleep is completed. |

---

Q explain thread groups and inter thread commiunication

---

# UNIT-IV

## Files and I/O Streams:

Stream is data flow. In case of C language, the input / output are referred with the terms standard input and standard output. The same are called input stream and output stream in OOPS.

In java, we can read / write data from/ to a **file / array / console / socket.** The java.io package has number of classes for managing data streams. The java streams are classified into two types.

- ➔ Byte streams
- ➔ Char streams

Byte streams: these are also called **Low Level Streams**. These streams read / write data in ASCII format, i.e. they read 1 byte and write 1 byte data at a time. The classes used for low level streaming are identified with a suffix InputStream, OutputStream in their names. These are used to write and read data from binary files. Generally the IO operations that have hardware interactivity are managed with byte streams.
Ex: FileInputStream , FileOutputStream, ByteArrayInputStream, ByteArrayOutPutStream, PrintStream etc.

Char streams: These are called **High Level Streams**. These streams read / write data in Unicode format, i.e. these read 2 bytes and write 2 bytes at a time. The classes that are used for high level streaming of data are identified with suffix Reader / Writer in their names. These are used to read and write data from text files. Generally the IO operations that do not have hardware interactivity are managed with char streams.
Ex: FileReader, FileWriter, BufferedReader, PrintWriter etc.

For Reading data from hard disk, either FileInputStream or FileReader are preferred. The FileInputStream reads 1 byte at a time and FileReader reads 2 bytes at a time. But if the file is a binary file, then FileInputStream must be preferred.

All the input streaming classes are derived from the class InputStream and all data writing classes are derived from the class OutputStream.
The InputStream and OutputStream classes are abstract classes, so they can not be instantiated directly or they can not be used directly by creating objects of them. These provide what to be for reading and writing. All streaming classes are derived from these two and they provide how the data is to be read or write as per their

specification.

For example the InputStream class has methods….

read() :- reads a single char /key stroke from key board and returns its ASCII value as an integer.

read(byte[]) : reads data into the given byte array

close() : used to close the stream.

available() : returns no. of bytes to be read.

All byte streams override these methods to read data 1 byte at a time. And all char streams override these methods to read 2 byte data at a time.

Similarly, the OutputStream class has methods…..

write(int ): int specifying a char is written

write(byte[]): writes given byte array

All byte streams override these methods to write data 1 byte at a time. And all char streams override these methods to write 2 byte data at a time.

**All streams other than PrintStream (the System.out) throws IOException.

File Streams : when we develop a program , we concentrate on user interface process and data storage. The data storage is managed with variables, arrays, objects etc. but these are managed with in main memory. To use data in future, we must store data onto hard disk in the form of files. In java, we use file streams for data writing and reading.

We can use the classes FileInputStream / FileOutputStream or FileReader/FileWriter to read/write data.

The byte stream classes read / write 1 byte at a time and the char streams read/write 2 bytes.

** for binary files, we must use only byte streams.

Ex:

```
class MyIO1{
public static void main(String a[]) throws IOException,FileNotFoundException{
FileOutputStream  fout=new FileOutputStream("c:/MyFolder/myfile.txt");
fout.write( new String("welcome to java files").getBytes() );
fout.close();
}
}
---------------------------------
class MyIO2{
public static void main(String a[]) throws IOException,FileNotFoundException{
FileWriter fw=new FileFileWriter("c:/MyFolder/myfile.txt");
fout.write("welcome to java files" );
fout.close();
}
```

}
---

In MyIO1, as we used byte streams, we wrote data by converting string into byte array and in MyIO2, as we use FileWriter which is a char stream, we wrote data as string itself.

Instead of fout.write("welcome to java files" ), we can also write as, fout.write(new String( "welcome to java files").toCharArray() ), by converting string into char array.

File reading also very similar to above programs, but we use FileInputStream and FileReader classes.

---

Q. write about file streams

---

Filter streams: are used for data conversions between different types of streams.

For example, for reading data from console (system's keyboard), we need to use InputStream class object (System.in). But reading data with "in" object, returns the ASCII code or reads data into a byte[], which further needs to be converted into Unicode format to express it to user.

Rather, we can use a class BufferedReader object for reading data. But, as BuferedReader is a char stream object, it can not interact with hardware components directly. So we can read data by holding System.in object with BufferedReader object.

Code :

```
BufferedReader br=new BufferedReader( new InputStreamReader( System.in) );
String s=br.readLine();
```

Here the data read from System.in is in ASCII format and the readLine() method of BufferedReader class returns data in String (Unicode) format. The class InputStreamReader acts as a filter between System.in object and br object , that converts data from ASCII standards into Unicode standards.

---

Q. write about FILTER streams

---

Random Access File:

The FileInputStream is used only or reading data from file and the FileOutputStream are used for writing data to file. But we can not do both operations at a time with any of those classes.

But the RandomAccessFile provides flexible way of doing both operations on file at a time.

This class is used for reading and writing to random access file. A random access file behaves like a large array of bytes. There is a cursor implied to the array called file pointer, by moving the cursor we do the read write operations. If end-of-file is reached before the desired number of byte has been read than EOFException is thrown. It is a type of IOException.

## Constructor

| Constructor | Description |
|---|---|
| RandomAccessFile(File file, String mode) | Creates a random access file stream to read from, and optionally to write to, the file specified by the File argument. |
| RandomAccessFile(String name, String mode) | Creates a random access file stream to read from, and optionally to write to, a file with the specified name. |

## Method

| Modifier and Type | Method | Method |
|---|---|---|
| void | close() | It closes this random access file stream and releases any system resources associated with the stream. |
| FileChannel | getChannel() | It returns the unique FileChannel object associated with this file. |

| | | |
|---|---|---|
| int | readInt() | It reads a signed 32-bit integer from this file. |
| String | readUTF() | It reads in a string from this file. |
| void | seek(long pos) | It sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs. |
| void | writeDouble(double v) | It converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the file as an eight-byte quantity, high byte first. |
| void | writeFloat(float v) | It converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the file as a four-byte quantity, high byte first. |
| void | write(int b) | It writes the specified byte to this file. |
| int | read() | It reads a byte of data from this file. |
| long | length() | It returns the length of this file. |
| void | seek(long pos) | It sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs. |

---

Q. write about RandomAccecssFile

---

Serialization:

**Serialization in Java** is a mechanism of *writing the state of an object into a byte-stream*. It is mainly used in Hibernate, RMI, JPA, EJB and JMS technologies, i.e. used in network applications.

When an object is transferred over network, the object is divided into data packets, where there is a chance of loss of object related data. To maintain persistent

data in the objet when transferred over network, the object can be made as serialized object. The java.io package has an interface Serializable and the class whose object need to be transferred over network must implement this interface.

Ex:

```
import java.io.Serializable;
public class Student implements Serializable{
 int id;
 String name;
 public Student(int id, String name) {
  this.id = id;
  this.name = name;
 }
}
```

---

Q. write about serialization in java

---

---

Applets:

The java programs are of two types.

➔ Applications

➔ Applets

Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

The basic differences between the applications and applets are :

| | |
|---|---|
| Applications are Java programs that are executed within JRE. | Applets are small Java programs that are executed within a web browser. |
| Application program execution starts with main function as the JRE calls main() | Applet program execution starts with init function as the browser calls init() |
| Java application programs have the | Applets don’t have local disk and |

| | |
|---|---|
| full access to the local file system and network. | network access. |
| Applications can access all kinds of resources available on the system. | Applets can only access the browser specific services. They don't have access to the local system. |
| Applications can execute the programs from the local system. | Applets cannot execute programs from the local machine. |

The advantages of applets are:

- ➔ It works at client side so have less response time.
- ➔ Secured
- ➔ It can be executed by browsers running under many platforms, including Linux, Windows, Mac Os etc.

The applet can be developed by extending the class "**java.applet.Applet**". The applet program is placed in a web server along with html files and executed in a web browser **embedded** into a web page.

**Application programs are compiled and executed at command prompt with the commands "javac" and "java".

Ex:
MyApp.java

```
------------------------------
public class MyApp extends Applet{
        public void init(){
                applet code & design
        }
}
   //<applet  code=MyApp width=400 height=300> </applet>
------------------------------
```

The applets are compiled and tested from command prompt with the commands "javac" and "appletviewer".

For testing the applet, we must write an applet tag within the same program or in any other file.

The applet tag and its attributes look as :
<applet code=MyApp width=400 height=300> </applet>

This code can be placed with in comments in the same source file MyApp.java or in any other file.

It is compiled as "javac MyApp.java" from command prompt and tested as "appletviewer MyApp.java" or "appletviewer otherfile name with extension".

Once testing is completed, it is placed into a web-server alongwith the web-site while deploying the web site. The applet tag is specified with in a HTML file. Once the html file is loaded into browser, the applet executes automatically.

The java applets are two types.
➔ Local applets
➔ Remote applets

The applets placed within same web-site are called local applets and they can be accessed directly.

If the applet placed in some other web site and accessed in our programs, then they are called remote applets. To specify the path with code attribute of applet tag, we must specify entire URL of the remote applet.

---

Questions that can be framed:
➔ What is an applet and explain different types of applets
➔ What are the type of java programs and explain them
➔ What are the differences between applications and applets

***

Applet life cycle:

The java's applet life cycle is as follows :

**INITIALIZE -> STAT -> STOP -> DESTROY**

These are called with the methods init(), start(), stop() and destroy().

The public void init() is used to initialize the applet. It is called by browser only once, when the browser opens the web page for first time that is embedded with applet program

The public void start() is automatically invoked after init() method or when the browser is maximized

The public void stop() is called when applet is stopped or when the browser is minimized

The public void destroy() is called when the applet is closed or when the browser is closed.

As the applet class is derived from java.awt.Component class, we can override the public void paint() method in an applet. The paint() takes an argument of java.awt.Graphics class. The RE calls the paint() and it is given with empty definition in Component class. We can override it as….

```
public void paint(Graphics g){
        g.drawString("welcome to FIRST APPLET", 100,100);
}
```

The Graphics is an abstract class in java.awt package that has more than 40 non-abstract methods like drawstring(), drawOval(), drawRect(), fillOval() etc. using which we can display things on Applet and other graphical components.

When the paint() executes, it is said that the applet is in **Display State**.

When an applet program is compiled, the javac generates .class file of the applet program. This .class file OR the byte code file of applet program is called an **executable applet**.

---

Questions that can be framed:

- ➔ Write about different states of applet
- ➔ Write about applet life cycle
- ➔ Write how to create executable applet
- ➔ Write about the display state of applet

**This part is not in your syllabus…. But it is for your general information.**

HTML : it is **Hyper Text Markup Language**, the web designing language used to develop web pages. The HTML has no special editor. The OS editor like notepad for windows is used to develop web pages.

These programs are typed in notepad and executed in browser.

The HTML is a TAG driven language. A tag is used to tie up features to the text. A tag in HTML is represented with < >.

The HTML tags are two types, Independent tags and Paired tags. The paired tags have closing tag also. A closing tag is like a normal tag prefixed with a / with in < >.

Examples :

<b> : for bold text
<u> : underlined text
<i> : italics

 : for center alignment
<br> : line break
<p> : paragraph
<hr> : horizontal row / line
<html> : denotes starting and ending of html program
<head> : to place header information like title, keywords if any assigned
<body> : document area code.

The <html> , <head> and <body> tags are optional from the version 5 of internet explorer browser. In previous versions these were must tags.

General outlook of html program:

```
<html>
    <head>
        Header information
    </head>

    <body>
        Document area code
    </body>
</html>
```

Once program is typed in notepad, it must be saved with extension .html (for example abc.html). it is executed in a browser by simply double clicking with mouse.

Many tags have attributes for customization of features.

<body> tag attr:
bgcolor = used to specify background color of document area
text = to specify text color

<hr> tag attr:
Color= specify line color
Width : how much width the line occupies in document area

Size : thickness of the line

**Till here it is not part of your syllabus**

**APPLET TAG**:

The tag <applet> is used to embed an applet program into a web page. It has a closing tags also. The attributes of <applet> tag are :

| | | |
|---|---|---|
| Align | URL | *Deprecated* − Defines the text alignment around the applet |
| Alt | URL | Alternate text to be displayed in case browser does not support applet |
| archive | URL | Applet path when it is stored in a Java Archive ie. jar file |
| Code | URL | A URL that points to the class of the applet |
| codebase | URL | Indicates the base URL of the applet if the code attribute is relative |
| height | pixels | Height to display the applet |
| name | name | Defines a unique name for the applet |
| object | name | Specifies the resource that contains a serialized representation of the applet's state. |
| Title | test | Additional information to be displayed in tool tip of the mouse |
| Width | pixels | Width to display the applet. |

The ALIGN attribute can be specified with values “LEFT / RIGHT / CENTER / JUSTIFIED”

We can pass parameters to an applet with the tag <param> with in applet opening / closing tags.

The tag PARAM has two attributes viz., name and value. The name attribute is used to specify name for the value and the value attribute is used to specify the parameter value.

This parameter can be fetched into applet with the method getParameter() of Applet class.

Ex:

```
import java.applet.*;
import java.awt.*;

public class MyApp extends Applet
{

	public void init(){
	}

	public void paint(Graphics g)
	{
		String s= getParameter("msg");
		g.drawString(s,100,100);
	}
}

/*<applet code=MyApp width=400 height=300>
<param name="msg" value="HELLO">
</applet>
*/
```

***** but the java doesn’t support applet programming from the**

**version JDK 9 onwards. And the browsers also won't support applets**

---

Q. write about applet tag

---

## Database Handling Using JDBC:

JDBC or Java Database Connectivity is a Java API to connect and execute the query with the database. It is a specification from Sun microsystems that provides a standard abstraction(API or Protocol) for java applications to communicate with various databases. It provides the language with java database connectivity standards. It is used to write programs required to access databases. JDBC, along with the database driver, can access databases and spreadsheets. The enterprise data stored in a relational database(RDB) can be accessed with the help of JDBC APIs.

**Definition of JDBC(Java Database Connectivity)**

JDBC is an API(Application programming interface) used in java programming to interact with databases. *The classes and interfaces of JDBC* allow the *application to send* requests *made by users to the specified database.*

**Purpose of JDBC**

Enterprise applications created using the JAVA EE technology need to interact with databases to store application-specific information. So, interacting with a database requires efficient database connectivity, which can be achieved by using the ODBC(Open database connectivity) driver. This driver is used with JDBC to interact or communicate with various kinds of databases such as Oracle, MS Access, Mysql, and SQL server database.

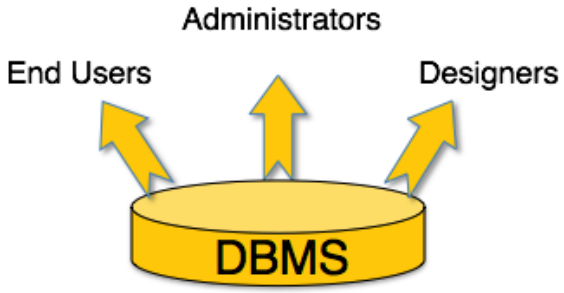
## An Overview of DBMS:

**Database** is a collection of related data and data is a collection of facts and figures that can be processed to produce information.

Mostly data represents recordable facts. Data aids in producing information, which is based on facts. For example, if we have data about marks obtained by all students, we can then conclude about toppers and average marks.

A **database management system** stores data in such a way that it becomes easier to retrieve, manipulate, and produce information.

Traditionally, data was organized in file formats. DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management




A DB product when installed into system, the DB user interface , DB Engine and a Physical DB is installed into system. The DB Engine is the actual runtime of the DB. It includes number of components like data descriptions, a language called SQL to interact with DB, ODBC (Open DB Connectivity) , meta data etc.

The SQL a language used for interacting with DB and the SQL is classified into 2 types dynamic and embedded. The queries passed from DB UI to DBE are called dynamic SQL and if the same SQL query is placed in a program and the executed through a program, then it is called embedded SQL.

The ODBC is the software mechanism through which the data comes out of the DB that is fetched into a java program using JDBC.

A programmer to work with DB must know CRUD operations. CRUD stands for Create data, Read data, Update data and Delete data.

Creating data is done with an insert query.

If we have a table STUD( sid, sname and fee), then we load data as….

Insert into stud values(101,'lakshmi',45000);

Reading data is done with select query:

Select sid,sname,fee from stud

Update is to modify data:

Update stud set fee=38000 where sid=101

Delete is to remove data from table:

Delete from stud where sid=101;

---

Q. write an overview on DBMS

---

JDBC Architecture:

The JDBC is java database connectivity a software mechanism used to connect to data. It is generally called JDBC driver. It is used to connect to a DB, through the ODBC of the DB.

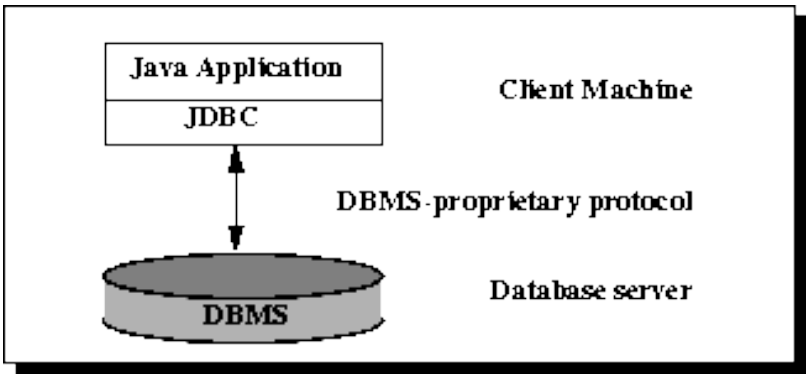
When we develop an application, we concentrate on 3 things viz., user interface, process and data storage. As data is vital in the application, if all three components are managed with one application, the data is logically associated with logic of the program and if we modify logic, it affects the data. To overcome this problem, the application can be divided into 2 parts, in which one part is managed with java program and the other part with a DBMS like oracle.

The java program manages user interface and business logic and the oracle manages the data. Here the application runs on 2 layers viz., java layer and oracle layer otherwise program layer and DB layer. So it is called 2tiered architecture.

It works as client-server application. When user uses this application, he can see only the java program, so it is called front end and the DB is

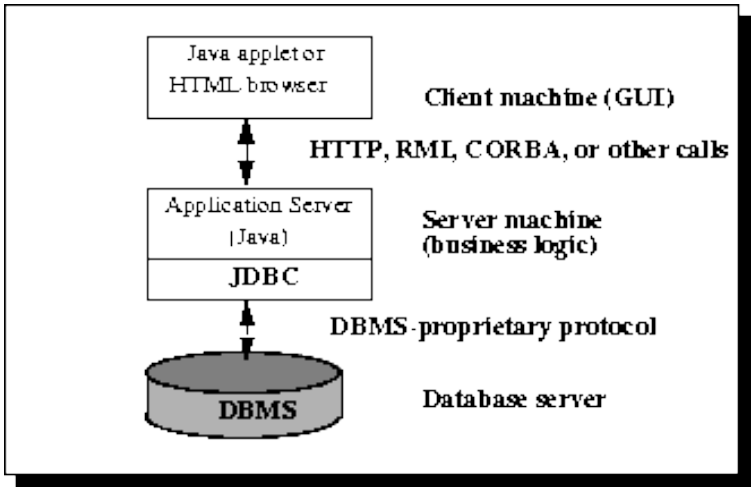
never worked by user directly as it stays at back of the application, so it is called backend.

A 2-tiered application looks as :




In above 2-tiered applications, the user interface and the logic are bound with each other strongly. Where if the user interface is modified it may affect logic or the user interface need to be updated in all systems to make use of updations. It is practically impossible to update front end in all the systems. And to make use of data, every system must have the front end installed. To overcome this problem, a 3-tiered architecture is designed , in which the Front End is designed with html and managed by web-servers and web-browsers. Middle tier (layer) is managed with java servlets, JSPs etc and the backend is managed with oracle like DB.

Then it is called an enterprise application. Its architecture looks as :



---

Q. write about JDBC architecture

Write about 2-tiered & 3 tiered architecture

---

Working with JDBC:

First we need to identify the drivers to be used for connecting java with DB. There are different types of drivers

JDBC drivers are client-side adapters (installed on the client machine, not on the server) that convert requests from Java programs to a protocol that the DBMS can understand. There are 4 types of JDBC drivers:

1. Type-1 driver or JDBC-ODBC bridge driver
2. Type-2 driver or Native-API driver
3. Type-3 driver or Network Protocol driver
4. Type-4 driver or Thin driver

The oracle thin driver is managed with a class "OracleDriver" of "oracle.dbc.driver" package.

In case of oracle 10g, it is available in the path of oracle installation as a jar file.

For ex:

C:\oraclexe\app\oracle\product\10.2.0\server\jdbc\lib\ojdbc14.jar

This need to be set in classpath for making use of it.

Once classpath is set, then we can load the above thin driver into memory for establishing connection. It can be done with the class "java.lang.Class" and its static method forName().

Code:

Class.forName("oracle.jdbc.driver.OracleDriver");

In database terminology it is said that the driver is **registered**.

The different classes and interface of java.sql package are used for

fetching data from DB into java program viz.,

DriverManager : it is a class, used to establish connection to the DB whose URL is specified using above loaded drivers, and returns the connection as an instance of java.sql.Connection interface.

The DB URL for oracle using thin drivers is :

**“jdbc:oracle:thin:@localhost:1521:xe”**

The DriverManager has a static method getConnection(), that establish connection.

Code :

Connection con=

DriverManager.getConenction(“jdbc:oracle:thin:@localhost:1521:xe”,”userid”,”password”);

Now…. The DriverManager’s getConnection() establish connection to oracle DB with name “xe” running on localhost with port number 1521 with given user name and password and returns the connection and that is held by con.

Connection : this interface is instantiated by the getConnection() of DriverManager and has methods to create Statement , PreparedStatement objects. These two objects are used to convert String query into SQL standard query and execute on DB.

Statement : this interface has methods execute() and executeUpdate() etc to execute SQL queries. This object is given with query while executing the query. So each time the query is compiled.

Ex:

Statement st=con.createStatement();

ResultSet rs= st.executeQuery(“select * from emp”);

st.executeUpdate(“update emp set sal=3000 where eid=101”);

PreparedStatemenet : this interface also has same methods execute() and executeUpdate() as of in Statement interface. This object is given with query while getting created. Query is not given while executing. So query is compiled once and can be executed any number of times.

The PreparedStatement has a facility of giving with “place holders” for values not known, and later the place holders modified with values. The place holder is represented with a “?”.

Ex1:

```
PreparedStatement pst1=con.prepareStatement(“select * from emp where eid=?”);

pst1.setInt(1, Integer.parseInt( tf.getText() ) );

ResultSet rs1=pst1.executeQuery();
```

Ex2:

```
PreparedStatement pst2=con.prepareStatement(“update emp set ename=?, sal=?  where eid=?”);

pst2.setString(1, tf.getText() );

pst2.setInt(2, Integer.parseInt( tf_sal.getText() ) );

pst2.setInt(3, Integer.parseInt( tf_num.getText() ) );

pst1.executeUpdate();
```

*irrespective of Statement or PreparedStatement used, the executeQuery() returns no. of rows data as instance of ResultSet interface. And the executeUpdate() returns an int representing no. of rows affected by query.

CallableStatement : used to execute built in stored procedures of the SQL. i.e. PL/SQL part queries like procedures, functions etc are execute by this CallableStatement object, to increase the speed of execution of the query.

ResultSet : it is an interface that is used to move across the data fetched by executeQuery() of either Statement or PreparedStatement objects.

It has methods first(), previous(), last() and next() to move across each row of data fetched. The methods getInt(), getString() etc are used to fetch data of each column in current row.

Ex:

System.out.println( rs.getInt(1) +" "+ rs.getString(2)+" "+ rs.getInt(3));

---

Q. explain working with JDBC

Write about classes and interfaces used for JDBC connectivity

Explain how to get data from a DB into a java program.

---

Unit V

Servlets

Introduction:

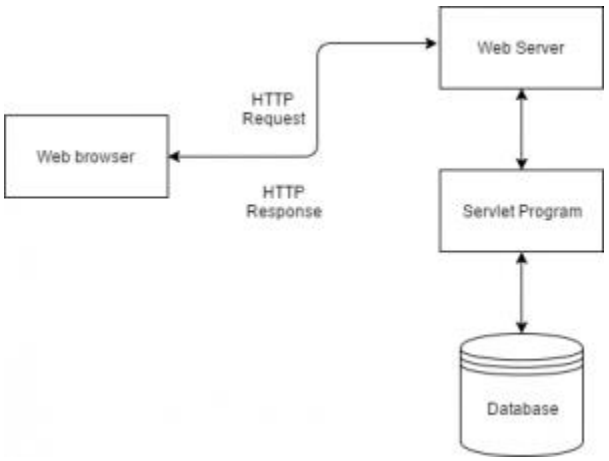
As we all are aware of the need of creating dynamic web pages i.e the ones which have the capability to change the site contents according to the time or are able to generate the contents according to the request received by the client. If we like coding in Java then the way is Java Servlet. But before we move forward with our topic let's first understand the need for server-side extensions.

Servlets are the Java programs that run on the Java-enabled web server or application server. They are used to handle the request obtained from the webserver, process the request, produce the response, then send a response back to the webserver.

Properties of Servlets are as follows:

- Servlets work on the server-side.
- Servlets are capable of handling complex requests obtained from the webserver.

Servlet Architecture is can be depicted from the image itself as provided below as follows:




How to run servlets:

The Servlets are not executed directly like a simple java program. They are executable only on a web server.

The J2EE when installed into system, it has a pre-defined web service that can be started manually and the servlet programs are placed into a location called servlet

container in the J2EE service.

But in practical life as the servlets run on web servers, there are many web servers that provide servlet api.

Taking example of tomcat web server, the tomcat has its own folder architecture to get placed with web pages, xml file and servlet classes.

We can create a project folder and place all web pages , xml file and servlet classes is its specific folders, generae a war file (war=Web ARchieve) and deploy the war file onto web server using the web server tools.

Or also, We create a project folder in tomcat's "webapps" folder, create a sub folder with name "WEB-INF" within project folder, create a subfolder "classes" within WEB-INF folder and place all servlet .class files into this classes folder.

When user opens the website through a browser and submit a request from a dynamic web page by clicking the "submit" button, then the request is submitted to the specific servlet whose URL is specified in the web page.

Thus a servlet program is executed on the web server by sending a request.

---

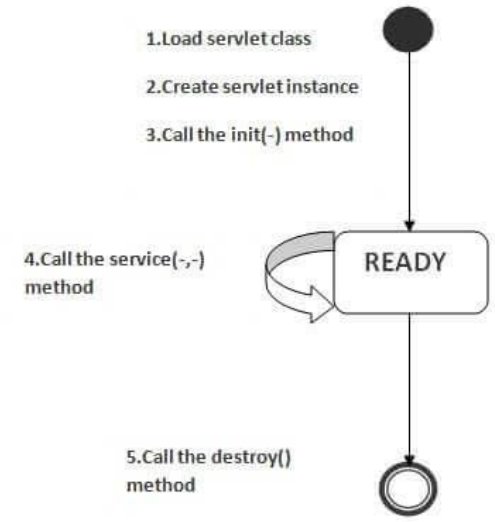
Q. explain how to run servlets
Give an introduction to servers and how to run them
How servlets are used on a web server

---

The Life–cycle of the servlet:
The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.

4. service method is invoked.
5. destroy method is invoked.




1) Servlet class is loaded
The class loader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.
2) Servlet instance is created
The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

3) init method is invoked
The web container calls the init method only once after creating the servlet instance. The in initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface. Syntax given below:

public void init() throws ServletException

4) service method is invoked
The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below:
public void service(ServletRequest request, ServletResponse response) throws Ser vletException, IOException

5) destroy method is invoked

The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:

public void destroy()

---

Q. explain the servlet Life cycle

---

servlet API:

The javax.servlet and javax.servlet.http packages represent interfaces and classes for servlet api.

The javax.servlet package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.

The javax.servlet.http package contains interfaces and classes that are responsible for http requests only.

There are many interfaces in javax.servlet package. They are as follows:

1. Servlet : base interface to all servlets
2. ServletRequest : used to handle request received from client
3. ServletResponse : used to handle responses to be sent to client using streams
4. RequestDispatcher etc

Classes in javax.servlet package

There are many classes in javax.servlet package. They are as follows:

1. GenericServlet : is an abstract class that has init(), service() methods to be overridden.
2. ServletInputStream
3. ServletOutputStream
4. ServletException and so on

Interfaces in javax.servlet.http package

There are many interfaces in javax.servlet.http package. They are as follows:

1. HttpServletRequest : used to handle http specific requests
2. HttpServletResponse : used to handle http specific responses
3. HttpSession : used to manage sessions between the requests and responses

Classes in javax.servlet.http package

There are many classes in javax.servlet.http package. They are as follows:

1. HttpServlet : it is derived from GenericServlet and has methods doGet() and doPost() to be overridden by programmer.
2. Cookie : it is also used for session management.

** when using tomcat server for running the servlets, till the tomcat version8.0 version, the API is named as javax.servlet and javax.servlet.http. but from the version tomcat9.0 onwards, the API is changed to Jakarta.servlet and Jakarta.servlet.http.

So to run servlets in current versions of tomcat server, we import jakarta.servlet.* and import jakarta.servlet.http.*;

---

Q. explain the servlet API

---

Multitier Applications using JDBC from a servlet:

Servlets are the server-side components that enhance the capability of Java-enabled servers and allow a variety of network-based services. For example, servlets with the database communication can facilitate solutions for distributed applications.

Typical three-tier architecture for database applications is shown in the Figure. The first tier (client) contains a browser (such as Netscape or Microsoft IE). The second tier contains the Java-enabled web server and the JDBC driver (for example, JDBC-ODBC bridge driver). The database is contained in the third tier. Servlets allow dynamic HTML pages to communicate with the client and database through JDBC


Writing JDBC/Servlet

Consider the example of the TestDataBaseCon servlet in Program which illustrates the use of the servlet framework in JDBC/Servlet communication. This servlet makes a connection to the database and returns data in HTML form.

Program Making a database connection using the init() method.

```
public class TestDataBaseCon extends GenericServlet
{
     Connection con = null;
     public void init() throws ServletException
     {
          try{
            class.forName("oracle.jdbc.driver.OracleDriver");
con=DriverManager.getConnection(“jdbc:oracle:thin:@localhost:1521:xe”,
“system”,”1234”);
        }
        catch(ClassNotFoundException e)
        {
            System.out.println("problem in loading the class:");
         }
         catch(SQLException e)
         {
              System.out.println("Sql error");
          }
}

     public void service(ServletRequest req, ServletResponse res){

               //do the process that is to be executed with this servlet
     }
     public void destroy()
     {
     // Close the connection and allow it to be garbage collected
```

```
            con.close();
            con=  null;
        }


    }
```

Servlets are built by extending a basic Servlet class and overriding some of the methods to deal with incoming client requests. TestDataBaseCon extends the GenericServlet class that provides most of the capabilities needed to serve client requests. After the servlet is loaded, three main methods are involved in the life-cycle of a servlet.
The init() used to initialize the DB connection, and the service() is used to provide the service to be given by that servlet and the destroy() is used to close the connection.

---

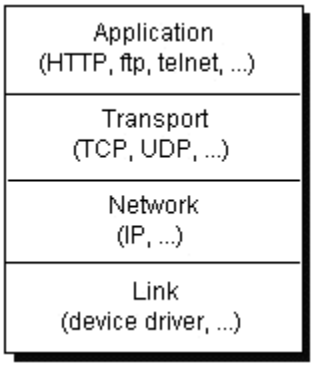
Q. explain multi tier application with DB connection in a servlet.

---

## Networking and Remote Method Invocation

Introduction to Networking:

Computers running on the Internet communicate to each other using either the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP), as this diagram illustrates:




When you write Java programs that communicate over the network, you are programming at the application layer. Typically, you don't need to concern yourself with the TCP and UDP layers. Instead, you can use the classes in

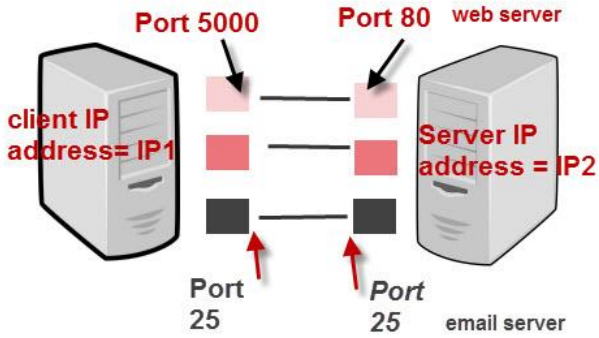
the java.net package. These classes provide system-independent network communication.
a system to be in networks.... should have some standards called protocols. the basic protocols are :
IP : is internet protocol taht specifies that each system must be identified with a unique number called IP address.
an ip address is a 4 byte number where each byte is separated with a DOT.
ex :

192.168.0.1

TCP: specifies that, when data transferred over network, it must be divided into small pieces called data packets and then transferred. at the destination all DPs are merged into one file and expressed to user.
in olden days, the IP and TCP were 2 different protocols. but now a days, it is combined into one and called TCP/IP suit.
** the TCP is a connection oriented protocol.

Understanding Ports:

An IP address alone is not sufficient for running network applications, as a computer can run multiple applications and/or services.
Just as the IP address identifies the computer, The network port identifies the application or service running on the computer.




IP Address + Port number = Socket

## TCP/IP Ports And Sockets

A port number uses 16 bits and so can therefore have a value from 0 to 65535 decimal
Port numbers are divided into ranges as follows:
Port numbers 0-1023 – Well known ports. These are allocated to server services by the Internet Assigned Numbers Authority (IANA). e.g Web servers normally use port 80 and SMTP servers use port 25 (see diagram above).

Ports 1024-49151- Registered Port -These can be registered for services with the IANA and should be treated as semi-reserved. User written programs should not use these ports.
Ports 49152-65535– These are used by client programs and you are free to use these in client programs. When a Web browser connects to a web server the browser will allocate itself a port in this range.

---

Q. write about basic protocols and port numbers required for networking

---

Networking Classes in JDK :
There are number of classes in java.net package used for network applications. The basic classes used are Socket and ServerSocket. These classes are used to transfer data between systems as per TCP standards.
The system on which ServerSocket object runs is called a server system and the client will have a Socket object running.
The server system will have a ServerSocket object and a Socket object running on it and the client will have only Socket object. The ServerSocket listens and the socket objects are used to transfer data over network as per TCP by creating streams generated from socket object.
ServerSocket: this can listen.
const :
    ServerSocket()
    ServerSocket(int port_no)
methods :
accept() : creates a Socket instance (that works as a P.A. to this SS)

Socket: this class is used for creating a Socket object, that can transfer and receive data with the help of streams developed from it.
const :
    Socket();
    Socket(String ip/sys_name, int port_no);
methods :
    getInputStream()
    getOutputStream()

CODE : - MySS1.java code that need to be executed on server system.

```
import java.net.*;
import java.io.*;

class MySS1
{
	public static void main(String a[])throws Exception
	{
		ServerSocket ss=new ServerSocket(8765);
		Socket skt=ss.accept();

		InputStream is=skt.getInputStream();
		OutputStream os=skt.getOutputStream();

		byte b[];
		String data;
BufferedReader br=new BufferedReader( new InputStreamReader(System.in));

		while(true)
		{
			b=new byte[100];
			is.read(b);
			data=new String(b);
			data=data.trim();
			System.out.println("CLIENT:>"+data);

			System.out.print("SERVER:>");
			data=br.readLine();
			os.write(data.getBytes() );

		}
	}
}
```

CODE : - MySkt1.java code that need to be executed on client machine.

```
import java.net.*;
import java.io.*;

class MySkt1
{
	public static void main(String a[]) throws Exception
	{
		Socket skt=new Socket("rajanipc",8765);
		OutputStream os=skt.getOutputStream();
		InputStream is=skt.getInputStream();
		byte b[];
		String data;
		BufferedReader br=new BufferedReader( new
InputStreamReader(System.in));

		while(true)
		{
			System.out.print("CLIENT:>");
			data=br.readLine();
			os.write(data.getBytes() );

			b=new byte[100];
			is.read(b);
			data=new String(b);
			data=data.trim();
			System.out.println("SERVER:>"+data);

		}
	}
}
```

---

Q. write about networking classes and how to use them?

---

Introduction to RMI:

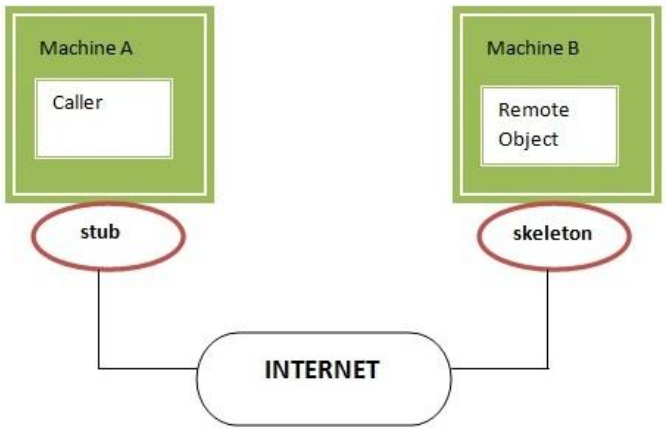
RMI stands for Remote Method Invocation. It is a mechanism that allows an object residing in one system (JVM) to access/invoke an object running on another JVM.

RMI is used to build distributed applications; it provides remote communication between Java programs. It is provided in the package java.rmi.

RMI Architecture:

In an RMI application, we write two programs, a server program (resides on the server) and a client program (resides on the client).

- Inside the server program, a remote object is created and reference of that object is made available for the client (using the registry).
- The client program requests the remote objects on the server and tries to invoke its methods.




RMI uses stub and skeleton object for communication with the remote object. The STUB and SKEL classes are generated from server objects with rmi compiler that is activated with command “rmic”. The SKEL class is placed on server system and the STUB class is placed on client system. The skel-stub establish communication from client to server and vice-versa.

The method of request and data send from client to server through the STUB and SKEL is called marshalling and the process of results sent from SKEL to STUB is called un-marshalling. The RMI programs establish communication on RMI protocol which is managed by a built in program of JDK called “rmiregistry”.

The different interfaces, classes used for RMI application are :

Remote (Interface) :

The interface java.rmi.Remote is a marker interface that defines no methods:

public interface Remote {}

In RMI, a remote interface is an interface that declares a set of methods that may be invoked from a remote Java virtual machine. As the pre-defined interface is a market interface, we extend this Remote interface with a user interface.

UnicastRemoteObject (Class) : The UnicastRemoteObject class defines a non-replicated remote object whose references are valid only while the server process is alive. The UnicastRemoteObject class provides support for point-to-point active object references (invocations, parameters, and results) using TCP streams.
Objects that require remote behavior should extend RemoteObject, typically via UnicastRemoteObject

Naming (class) : this class has methods for binding names to server objects running on remote systems that can be looked up from a client.

Methods : bind("string name", server obj)
rebind("string name", server obj)
lookup("string name")

RemoteException : the classes that run on rmi protocol always throws RemoteException.

---

Q. write about classes and interfaces used in RMI applications.
Give an introduction to RMI and its related classes.

---

Implementing Remote class and interface:

MyInt.java

```
import java.rmi.*;
public interface MyInt extends Remote
{
public int add(int a, int b) throws RemoteException;
}
```

MyServer.java

```
import java.rmi.*;
import java.rmi.server.*;
```

```
public class MyServer extends UnicastRemoteObject implements MyInt
{
	public MyServer() throws RemoteException
	{}

	public int add(int a, int b) throws RemoteException
	{
		return a+b;
	}

}
```

ServerRun.java

```
import java.rmi.*;

class ServerRun
{
	public static void main(String a[]) throws Exception
	{
		Naming.rebind("ms",new MyServer());
	}
}
```

MyClient.java

```
import java.rmi.*;

class MyClient
{
	public static void main(String a[]) throws Exception
	{
	MyInt mi= (MyInt)Naming.lookup("ms");

	System.out.println(mi.add(5,6));
	}
}
```

compile the MyInt.java program, it generates MyInt.class
compile MyServer.java program, it generates MyServer.class

compile the MyServer with “rmic” that generates SKEL & STUB classes.
NOTE: from the JDK1.5 version onwards, the skel is auto generated when a request is submitted. We get only stub class created physically now.

Compile server running class and client program.

Copy user interface, server program, server running program byte code files onto server machine.
Copy user interface, client program and stub class byte code files onto client machine.

Use the command “Start rmiregistry” at command prompt that maintains rmi protocol on server machine.
Run server program on server machine
Run client program on client machine.

Security: always run a security policy file to provide high security to rmi applications.

policy

```
// In this example, for simplicity, we will use a policy file that
// gives global permission to anyone from anywhere. Do not use this
// policy file in a production environment.

grant {
	permission java.net.SocketPermission "*", "accept,resolve";
};
```

---

Q. how to implement remote interfaces and classes?
How an rmi program is executed?

---