Unit-1

Q) Expalin about features of java?

Java is a widely used object-oriented programming language and software platform that runs on billions of devices . Java is system independent language other reasons for to popularity of this language is:

1.simple:-

java is a simple programming language for the following reasons

a)the difficult concepts of c and c++ have been eliminated in java.

for example the concept of "pointers" in c language is difficult for both learners and programmers .so pointer concept completely eliminated in java.

b) java is developed the same syntax of c and c++ in some concepts ,so the programmer who knows c and c++ language java is simple.

c) java contains rich set of "API "

API is collection of packages,packages is a collection of predefined class,class is a collection of data members and methods

d) java contain automatic memory management

2) Portable:

Portability is one of the core features of java which enables the java programs to run on any computer or operating system.

If a programme given same result on every machine ,then that programme is called portable.

3) interpreted language:-

After compiling the java programme ".class" (byte code)file is generated if we take any other language only interpreter or compiler to execute a programme , but in java using both compiler and interpreter for the execution so java is interpreter language.

4 robust:-

Robust means strong java language is strong for the following two reasons

a)java contains predefined "concept" exception handling for converting system to error messages into user friendly messages .run time errors or called exception .in java any exception occur(happened) no harm will happen

b)automatic memory management

5.high performance:-

The problem with interpreter inside the JVM is that is slow because of this java programme execution is slow .to overcome this drawback

along with the interpreter java soft people introduce JIT(just in time compiler)t o speed the java program execution . so JVM uses both interpreter and JIT compiler to execute a java programme ,so java is high performance language.

6) Secure:

Java is said to be more secure programming language because it does not have pointers concept , java provides a feature "applet" which can be embedded into a web application. The applet in java does not allow access to other parts of the computer.

Security problems like ever draping and tempering and virus threats can be eliminated or minimized by using java on internet

7.distributed:-

Information is distributed on various computers on a network using java. we can write programme and distribute it to the clients . this is possible because java can handle the protocals like TCP/IP and UDP (universal development protocal)

8.object oriented:-

java is said to be a pure object-oriented programming language. In java, everything is an object . It supports all the features of the object-oriented programming paradigm . any programming language is said to be object oriented programming language,that language satisfy following 8 properties

1.class 2.object

3.abstraction 4.encapsulation

5.inheritence 6.polymorphism

7.dynamic binding 8.message passing

In java multiple inheritance is not supported at classes level but supported in interfaces level.

9.multithreading:-

In java programme if we write a group of statememts then the statements are executed by JVM one by one .this execution is called a thread , because JVM uses a thread to execute statements . in every java programme contain atleast two threads to execute a java program . so java multithreaded programming language . In a java program multiple threads are executed at the same time.

10.dynamic:-

Before the development of java only static text used to be displayed in the browser but in java using applet programme dynamic text will be displayed on browser.so java is a dynamic.

Java is said to be dynamic because java contain dynamic memory allocation and deallocation (objects and garbage collector).

Q) explain about JVM ?

JVM (Java Virtual Machine) Architecture:

JVM means java Virtual Machine and it is a program . JVM provides runtime environment in which java bytecode can be executed.

JVM are available for many hardware and software platforms (i.e. JVM is platform dependent).

What it does:

The JVM performs following operation:

Loads code

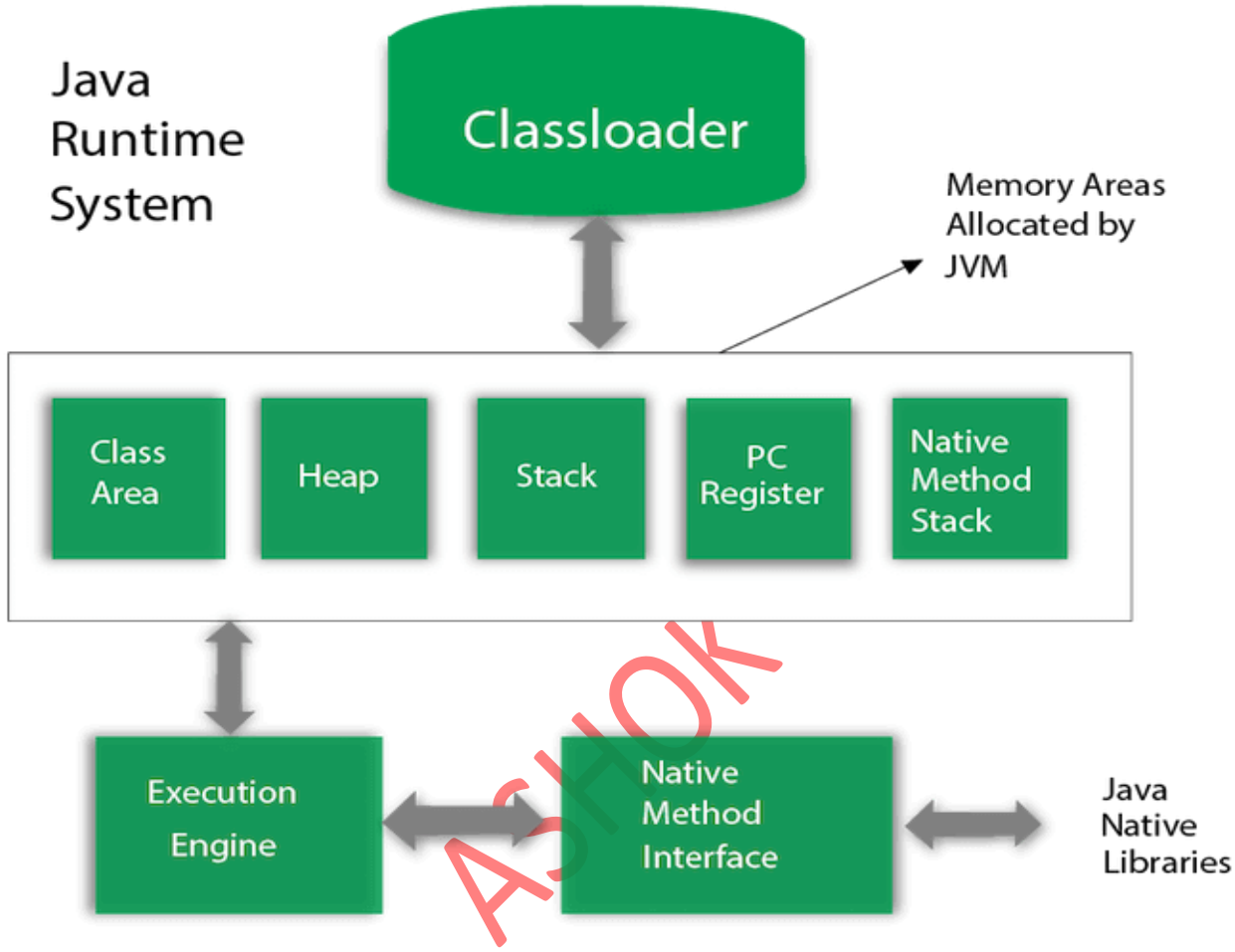
Verifies code

Executes code

Provides runtime environment

JVM Architecture

Let's understand the internal architecture of JVM. It contains classloader, memory area, execution engine etc.



JVM(Java Virtual Machine) acts as a run-time engine to run Java applications. JVM is the one that actually calls the main method present in a java code. JVM is a part of JRE(Java Runtime Environment).

Java applications are called WORA (Write Once Run Anywhere). This means a programmer can develop Java code on one system and can expect it to run on any other Java-enabled system without any adjustment. This is all possible because of JVM.

When we compile a *.java* file, *.class* files(contains byte-code) generated by the Java compiler. This *.class* file goes into various steps when we run it. These steps together describe the whole JVM.

JVM Memory :

Method area: In the method area, all class level information like class name, immediate parent class name, methods and variables information etc. are stored, including static variables.

Heap area: Information of all objects is stored in the heap area. There is also one Heap Area per JVM. It is also a shared resource.

Stack area : For every thread, JVM creates one run-time stack which is stored here.

PC Registers: Store address of current execution instruction of a thread. Obviously, each thread has separate PC Registers.

Native method stacks : For every thread, a separate native stack is created. It stores native method information.

Q)explain about parts of JAVA?

Parts of JAVA:

1) Java SE (Java Standard Edition):

Java SE means java standard edition . it is used to develop stand alone applications . It is a Java programming platform. It includes Java programming APIs such as java.lang, java.io, java.net, java.util,

java.sql, java.math etc. It includes core topics like OOPs, String, Exception handling, Multithreading, I/O Stream, Networking, AWT, Swing, Collection, etc.

2) Java EE (Java Enterprise Edition)

Java EE menas Java Enterprise Edition.It is an enterprise platform that is mainly used to develop web and enterprise applications. It is built on top of the Java SE platform. It includes topics like Servlet, JSP, Web Services, EJB, JPA, etc.

3) Java ME (Java Micro Edition)

Java ME means java mobile edition . It is a micro platform that is dedicated to mobile applications.

Q) explain Naming conventions in java?

Java Naming Convention

Java naming convention is a rule to follow some rules for given names for class, package, variable, constant, method, etc.

These conventions are suggested by several Java communities such as Sun Microsystems and Netscape.

All the classes, interfaces, packages, methods and fields of Java programming language are given according to the Java naming convention.

Naming Conventions of the Different Identifiers:

The following table shows the popular conventions used for the different identifiers.

Identifiers Type	Naming Rules	Examples
Class	It should start with the uppercase letter. It should be a noun such as Color, Button, System, Thread, etc. Use appropriate words, instead of acronyms.	System NumberFormateException
Interface	It should start with the uppercase letter. It should be an adjective such as Runnable, Remote, ActionListener.	Runnable
Method	It should start with lowercase letter. It should be a verb such as main(), print(), println(). If the name contains multiple	Print() nextInt()

	words, start it with a lowercase letter followed by an uppercase letter such as actionPerformed().	
Variable	It should start with a lowercase letter such as id, name. It should not start with the special characters like & (ampersand), $ (dollar), _ (underscore). If the name contains multiple words, start it with the lowercase letter followed by an uppercase letter such as firstName, lastName. Avoid using one-character variables such as x, y, z.	firstName lastName
Package	It should be a lowercase letter such as java, lang. If the name contains multiple words, it should be separated by dots (.) such as java.util, java.lang.	Java.lang Java.util
Constant	It should be in uppercase letters	MIN_PRIORITY

such as RED, YELLOW. MAX_PRIORITY

If the name contains multiple words, it should be separated by an underscore(_) such as MAX_PRIORITY.

It may contain digits but not as the first letter.

Q).explain about constants?

Constant is fixed value .the value of constant can not be change in execution time

Integer constant:

An integer constant contain members without decimal point values

Ex: 10,20,30

Integer constant rules:

- Integer constant should contain atleast one digit
- Integer constant should not contain decimal point
- The sign of of integer constant is either positive or negative
- By default sign is positive
- Range is 4 bytes in java

Real constant:

Real constant is used for storing the decimal point value

Ex: 0.002,0.764

Real constant rules:

- **A real constant should contain atleast one digit**
- **A real constant should contain a decimal point**
- **The sigh of real constant positive or negative**
- **By default sign is positive**
- **Range is 4 bytes in java**

Note: exponential constant follow the all rules of floating point constant there are used for storing the exponential values like e-32 ,e-12

Character :-

An alphabets or digits or special symbols enclosed or keep within a single quotation called character,character are four types

1)alphabetic character:

Alphabets(a to z and A to Z) enclosed within a single quotation(' ') is called alphabetic character

Ex: char x='a';

Char y='x';

2)Numeric character:

Digits(0 to 9) enclosed within a single quotation is called numeric character

Ex: char a='2';

3)special character:

Special symbols are enclosed within single quotation is called special symbol character

Ex: char x='&';

String character:

Collection of characters is called string is enclosed with in a double quotation

Ex: string x="ashok";

In java string is given as a class

In java every class is a data type

Q) expalin about data types in java?

- **Datatypes indicates which type of data stored into a variable**
- **Datatypes specify the size and type of values that can be stored**
- **Every variable in java should contain datatype**
- **Datatype indicates how much amount of memory space created in the main memory of the computer(ram) to store value in to a variable**

In java primitive datatypes are devided inti 4 categories

1.integer datatype

2 real datatype

3.character datatype

4.boolean datatype

Integer data type:

These datatypes represent integer numbers that means number without any decimal point.

Ex: 10,20,-90 etc

Integer datatypes are contain following datatypes

Byte	**1 byte**	**-128 to 127**
Short	**2 bytes**	**-32768 to 32767**
Int	**4 bytes**	**-2147483648 to 2147483647**
Long	**8 bytes**	**-9223372036854775808 to 9223372036854775807**

Float datatype:

- **Float category data types are used for representing the data in the form of scale and precision i.e ,these category datatypes are used for representing float values**
  - **Ex: 10.45,2.56**

- **This category contains to datatypes they are given the following table**
- **Whenever we take any decimal constant directly in java programme it is by default treated as highest datatype in float category i.e double**

s.no	Datatype	Size(bytes)	range	No of decimal places
1 2	Float Double	4 8	-2147483648 to 2147483647 $-9.223*10^{18}$ to $9.223*10^{17}$	8 16

Character datatypes:

A character is an identifier which is enclosed within a single quotation

Ex: 'a','5','x'

In java to represent character data we use a datatype called character the datatypes takes 2 bytes since it follows UNI code character set.

Java is available in 18 international language ang it is following UNI code character set

UNICODE character set is one which contain all the characters which are available in 18 international languages and it contains 65536 characters.

Data type	Size(bytes)	range
Char	2	-32768 to 32767

Boolean datatype:

- **Boolean category datatype is used for representing logical values i.e true or false values**
- **To represent logical values we use a 'keyword" called Boolean**
- **This datatype takes zero bytes of memory space**

Q) Explain about structure of java ?

Ans.

Structure of java program:

Sun micro system has prescribed the following standard structure for the java programmer to develop their application.

Syntax:-

Comment section;

```
Import packages;

Class <class name>

 {

        Data  members;

        User defined methods;

        Public static void main(string args[])

        {

                Block of statement;

            }

 }
```


- **Comments : the java comments are statements that are not executed by compiler and interpreter . comment are three types .**

  1) **Single line comments**
  2) **Multi line comments**
  3) **Documentation comments**

1) Single line comments :- the single line comments is used to comment only one line.

2) Multi line comments :- the multiline comment is used to comment multiple lines of code.

Syntax : /* this is multi line comment

Java program for printing msg*/

In the above syntax :

- **Package: a package is a collection of classes, interfaces and sub packages. A sub packages contains collections of classes, interfaces and sub packages etc.**
- **"class" is a keyword used to develop a user defined data type.**
- **"class name" is any java valid variable name. this is user defined name.**
- **Class code starts with a opening"{"and ends with"}**
- **"data members" will be selected based on the class name. these variables are instance or static.**
- **"user defined methods" a new used or performing some specific operations .every user defined method must be defined with in the class, and these methods are static or instance.**

Public static void main(string args[]) ::

- **Every java program starts execution from main) only . main() not return any value and hence the return type must be "void"**
- **Since main() is calling only once in program execution .this type of methods are called "static" method**
- **Main() must be accessed by every java program and hence whose access modifier must be "public"**

- **Main() should be taken array strings are parameters ,main() is written in java as public static void main(string args[])**
- **Main() contain the collection of statements**
- **The file name is given as same name of the class name with extension ”java”**

Java compilation : javac filename. Java

Java execution : java filename

- **Public – to call by JVM from any where**
- **Static – without object JVM call this method**
- **Void –main()cannot return any values**
- **Main –name of the method which is executed by JVM**
- **String args[]- command line arguments**
- **Example program for printing a message in java :**


```
Class ashok
{
        Public static void main (string args[])
        {
            System out println(“hai……”);
        }
}
```

Out put : hai……

Q) explain JAVA tokens?

Java Tokens:

A Java program is a collection of tokens, comments and white spaces. There are five types of tokens included in java language. They are:

1)key keywords (reserved words)

2)Identifiers

3)Literals

4)Operators

5)Seperators

1)Keywords:

Java keywords are also known as reserved words (or) predefined words. Keywords are particular words which acts as a key to a code. These are predefined words by Java so it cannot be used as a variable or object name. in jva contain bove 50 keywrds

List of Java Keywords

Class , public , static , void , Byte , short , int , long , float , double , char , Boolean , if , else , switch , while , for , ...

2)Identifiers:

In programming languages, identifiers are used for identification purposes. In Java, an identifier can be a class name, method name, variable name, or label. For example :


```
public class Test
{
   public static void main(String[] args)
   {
      int a = 20;
   }
}
```

In the above java code, we have 5 identifiers namely (these 5 are names in a java program) :

Test : class name.

main : method name.

String : predefined class name.

args : variable name.

a : variable name.

Rules for defining Java Identifiers:

There are certain rules for defining a valid java identifier. These rules must be followed, otherwise we get compile-time error. These rules are also valid for other languages like C,C++.

Identifier should start with alphabet(a to z or A to Z) or underscore(_) or dolar($) . Identifiers should not start with digits([0-9]).

For example "123geeks" is a not a valid java identifier.

Java identifiers are case-sensitive.

key Words(reserved words) can't be used as an identifier.

For example "int while = 20;" is an invalid statement as while is a reserved word. There are 53 reserved words in Java.

The only allowed characters for identifiers are all alphanumeric characters([A-Z],[a-z],[0-9]), '$'(dollar sign) and '_' (underscore).

For example "geek@" is not a valid java identifier as it contain '@' special character.

There is no limit on the length of the identifier but it is advisable to use an optimum length of 4 – 15 letters only.

Examples of valid identifiers :

MyVariable	**MYVARIABLE**	**myvariable**
X	**I**	**x1**
i1	**_myvariable**	**$myvariable**
sum_of_array	**geeks123**	

Examples of invalid identifiers :

My Variable // contains a space

123geeks // Begins with a digit

a+c // plus sign is not an alphanumeric character

variable-2 // hyphen is not an alphanumeric character

sum_&_difference // ampersand is not an alphanumeric character

3)literls:

Literal : Any constant value which can be assigned to the variable is called as literal/constant.

// Here 100 is a constant/literal.

int x = 100;

different types of literals are I) integer literal

II) float literal

III) character literal

IV) string literal

V) Boolean literal

I)Integer literals :

For Integer literal are without decimal point values , we can specify integer literals in 4 ways:-

a)Decimal literals (Base 10) : In this form the allowed digits are 0-9.

int x = 101;

b)Octal literals (Base 8) : In this form the allowed digits are 0-7.

// The octal number should be prefix with 0.

int x = 0146;

c)Hexa-decimal literals (Base 16) : In this form the allowed digits are 0-9 and characters are a-f. We can use both uppercase and lowercase characters. As we know that java is a case-sensitive programming language but here java is not case-sensitive.

// The hexa-decimal number should be prefix

// with 0X or 0x.

-int x = 0X123Face;

d)Binary literals : From 1.7 version onward we can specify literals value even in binary form also, allowed digits are 0 and 1. Literals value should be prefixed with 0b or 0B.

int x = 0b1111;

II)Floating-Point literal :

For Floating-point literals are decimal point values , we can specify literals in only decimal form and we can't specify in octal and Hexa-decimal forms.

Decimal literals(Base 10) : In this form the allowed digits are 0-9.

double d = 123.456;

III)Char literal:

We can specify literal to char data type as single character within single quote.

char ch = 'a';

IV)String literal:

Any sequence of characters within double quotes is treated as String literals.

String s = "Hello";

v)boolean literals :

Only two values are allowed for Boolean literals i.e. true and false.

boolean b = true;

4)operators:

Operators are used to perform operations on variables and values.

In the example below, we use the + operator to add together two values:

```
int x = 100 + 50;
```


Java divides the operators into the following groups:

Arithmetic operators

Assignment operators

Comparison operators

Logical operators

Bitwise operators

Increment operator

Decrement operator

Special operators(misslenious operators)

(see the explanation in previous topic)

5)Separators:

Separators help define the structure of a program. The separators are parentheses, ( ), braces, { }, the period, ., and the semicolon, ;. The table lists the six Java separators (nine if you count opening and closing separators as two). Following are the some characters which are generally used as the separators in Java.

Separator	Name	Use
()	Parenthesis	This holds the list of parameters in method definition. Also used in control statements & type casting.
{}	Braces	This is used to define the block/scope of code, class, methods.
[]	Brackets	It is used in array declaration.
.	Period	It is also used to separate variable or method from its object or instance. It is used to separate the package name from sub-package name & class name.
,	Comma	It is also used to separate the consecutive variables of same type while declaration . It

		is used to separate the consecutive parameters in the method definition..
;	Semicolon	It is used to terminate the statement in Java.

Q)explain java Operators ?

Operators are used to perform operations on variables and values.

In the example below, we use the + operator to add together two values:

```
int x = 100 + 50;
```


Java divides the operators into the following groups:

Arithmetic operators

Assignment operators

Comparison operators

Logical operators

Bitwise operators

1)Arithmetic Operators :

Arithmetic operators are used to perform common mathematical operations. these are 5.

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x – y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y

2)Relational operators (Java Comparison Operators) :

Comparison operators are used to compare two values: these are 6.

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

3)Java Assignment Operators :

Assignment operators are used to assign values to variables.

In the example below, we use the assignment operator (=) to assign the value 10 to a variable called x:

Example

```
int x = 10;
```

The addition assignment operator (+=) adds a value to a variable:

Example

```
int x = 10;

x += 5;
```

A list of all assignment operators:

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x – 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3

4)Java Logical Operators :

Logical operators are used to determine the logic between variables or values:

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

5) Java Bitwise Operators

Bitwise operators are used to perform binary logic with the bits of an integer or long integer.

Operator	Description	Example	Same as	Result	Decimal
&	AND - Sets each bit to 1 if both bits are 1	5 & 1	0101 & 0001	0001	1
|	OR - Sets each bit to 1 if any of the two bits is	5 | 1	0101 | 0001	0101	5

	1				
^	XOR - Sets each bit to 1 if only one of the two bits is 1	5 ^ 1	0101 ^ 0001	0100	4
~	NOT - Inverts all the bits	~ 5	~0101	1010	10
<<	Zero-fill left shift - Shift left by pushing zeroes in from the right and letting the leftmost bits fall off	9 << 1	1001 << 1	0100	4
>>	Signed right shift - Shift right by pushing copies of the leftmost bit in from the left and letting the rightmost bits fall off	9 >> 1	1001 >> 1	10010	18

6)Increment and decrement operator:

In programming (Java, C, C++, JavaScript etc.), the increment operator ++ increases the value of a variable by 1. Similarly, the decrement operator -- decreases the value of a variable by 1.

++ and -- operator as prefix and postfix

If you use the ++ operator as prefix like: ++var. The value of var is incremented by 1 then, it returns the value.

If you use the ++ operator as postfix like: var++. The original value of var is returned first then, var is incremented by 1.

The -- operator works in a similar way like the ++ operator except it decreases the value by 1.

Example :

```
class Operator {
   public static void main(String[] args) {
      int var1 = 5, var2 = 5;


      // var1 is displayed
      // Then, var1 is increased to 6.
      System.out.println(var1++);
```

```
        // var2 is increased to 6

        // Then, var2 is displayed

        System.out.println(++var2);

    }

}
```

7)Conditional Operator (? :) :

Conditional operator is also known as the ternary operator. This operator consists of three operands and is used to evaluate Boolean expressions. The goal of the operator is to decide, which value should be assigned to the variable. The operator is written as −

variable x = (expression) ? value if true : value if false

example :

```
class Test {


  public static void main(String args[]) {

    int a, b;

    a = 10;

    b = (a == 1) ? 20: 30;
```

```
      System.out.println( "Value of b is : " +  b );


      b = (a == 10) ? 20: 30;
      System.out.println( "Value of b is : " + b );
   }
}
```

This will produce the following result −

Output

Value of b is : 30

Value of b is : 20

8)Miscellaneous Operators :

There are few other operators supported by Java Language.

a)instanceof Operator

This operator is used only for object reference variables. The operator checks whether the object is of a particular type (class type or interface type). instanceof operator is written as −

(Object reference variable) instanceof (class/interface type)

If the object referred by the variable on the left side of the operator passes the IS-A check for the class/interface type on the right side, then the result will be true. Following is an example

Example

```
public class Test {


   public static void main(String args[]) {


      String name = "James";


      // following will return true since name is type of String

      boolean result = name instanceof String;

      System.out.println( result );

   }

}
```

This will produce the following result −

Output

```
true
```

b) Java new Keyword:

The Java new keyword is used to create an instance of the class.

It is used to create the object.

It allocates the memory at runtime.

All objects occupy memory in the heap area.

Syntax

```
NewExample obj=new NewExample();
```

c) dot (.) operator :

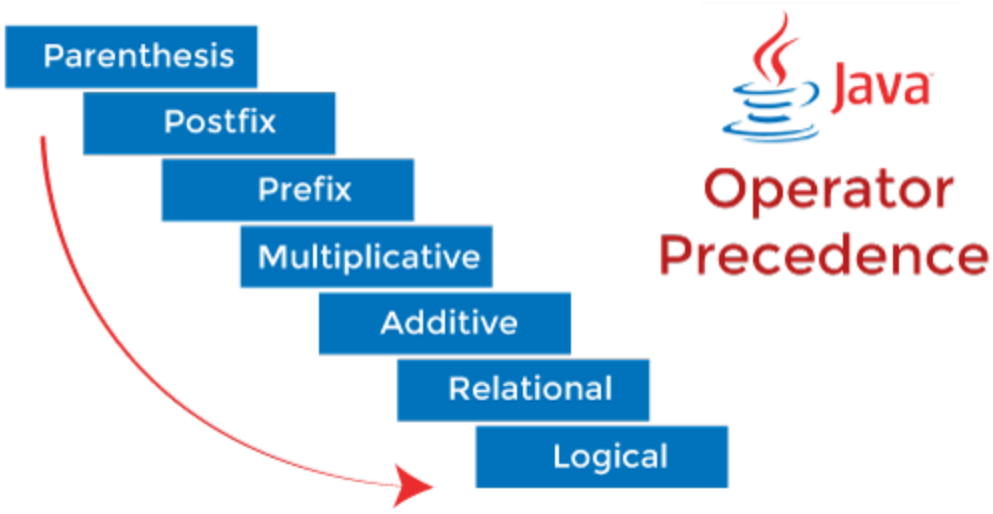
The dot (.) operator is used to select members of a class or object instance:

Q) explain operator precedence and priority ?

Java Operator Precedence

The operator precedence represents how two expressions are bind together. In an expression.

While solving an expression two things must be kept in mind the first is a precedence and the second is associativity.



java Operator Precedence Table

The following table describes the precedence and associativity of operators used in Java.

Precedence	Operator	Type	Associativity
15	()	Parentheses	Left to Right
	[]	Array subscript	
	.	Member selection	
14	++	Unary post-increment	Right to left
	--	Unary post-decrement	
13	++	Unary pre-increment	Right to left
	--	Unary pre-decrement	
	+	Unary plus	
	-	Unary minus	

	!	Unary logical negation	
	~	Unary bitwise complement	
	(type)	Unary type cast	
	*	Multiplication	
12	/	Division	Left to right
	%	Modulus	
	+	Addition	
11	-	Subtraction	Left to right
		Bitwise left shift	
	<<	Bitwise right shift with sign	
10	>>	extension	Left to right
	>>>	Bitwise right shift with zero extension	
	<	Relational less than	
	<=	Relational less than or equal	
9	>	Relational greater than	Left to right
	>=	Relational greater than or equal	
	instanceof	Type comparison (objects only)	
	==	Relational is equal to	
8	!=	Relational is not equal to	Left to right
7	&	Bitwise AND	Left to right

6	^	Bitwise exclusive OR		Left to right
5	|	Bitwise inclusive OR		Left to right
4	&&	Logical AND		Left to right
3	||	Logical OR		Left to right
2	? :	Ternary conditional		Right to left
1	=	Assignment		Right to left
	+=	Addition	assignment	
	-=	Subtraction	assignment	
	*=	Multiplication	assignment	
	/=	Division	assignment	
	%=	Modulus assignment		

Note: Larger number higher the precedence.

Q) explain control statements in java ?

Control statements

Control statements are 3 types

1)Conditional (or) selection (or) Decision making statements

a)simple if

b) if-else

c) else if ladder

d) nested if

e) switch

2) iterative (or) looping statements

a) while loop

b) for loop

c) do-while loop

3) jump statements

a) break

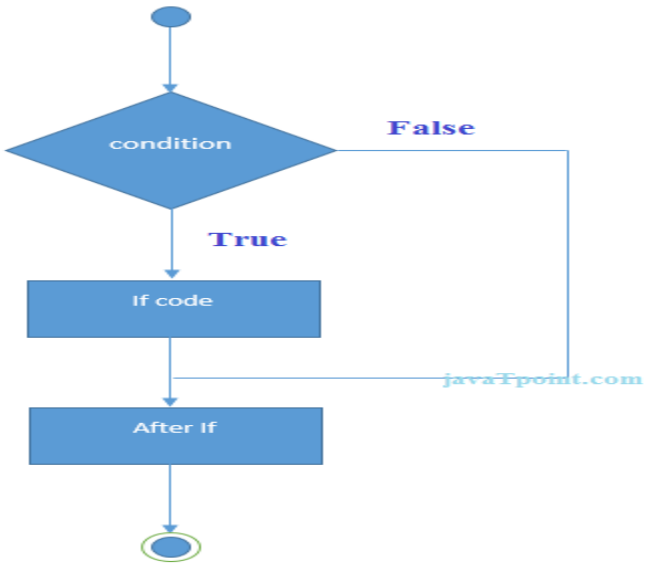
b) continue

c) goto

a)If Statement :

The if statement is used to check some given condition and perform some operations depending upon the correctness of that condition . The syntax of the if statement is given below.

```
if(expression)
 {
  //code to be executed
 }
```



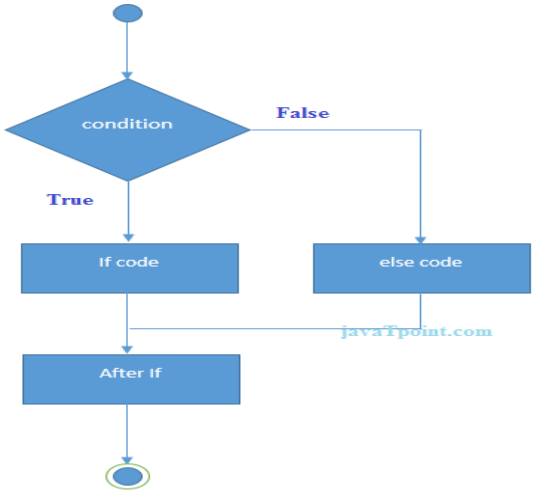
Let's see a simple example of C language if statement.

b)If-else Statement:

The if-else statement is used to perform two operations for a single condition. The if-else statement is an extension to the if statement using which, we can perform two different operations, i.e., one is for the correctness of that condition, and the other is for the incorrectness of the condition . Here, we must notice that if and else block cannot be executed simiulteneously . The syntax of the if-else statement is given below.

```
if(expression)
 {
        //code to be executed if condition is true
 }
else
{
        //code to be executed if condition is false
```

}



3)If else-if ladder Statement :

The if-else-if ladder statement is an extension to the if-else statement. It is used in the scenario where there are multiple cases to be performed for different conditions. In if-else-if ladder statement, if a condition is true then the statements defined in the if block will be executed, otherwise if some other condition is true then the statements defined in the else-if block will be executed, at the last if none of the condition is true then the statements defined in the else block will be executed. There are multiple else-if blocks possible. It is similar to the switch case statement where the default is executed instead of else block if none of the cases is matched.

```
if(condition1)
{
```

```
        //code to be executed if condition1 is true
    }
  else if(condition2)
  {
        //code to be executed if condition2 is true
  }
 else if(condition3)
 {
     //code to be executed if condition3 is true
  }
    ... ……................
     ……..................
  else
  {
      //code to be executed if all the conditions are false
  }
```


Flowchart of else-if ladder statement in C

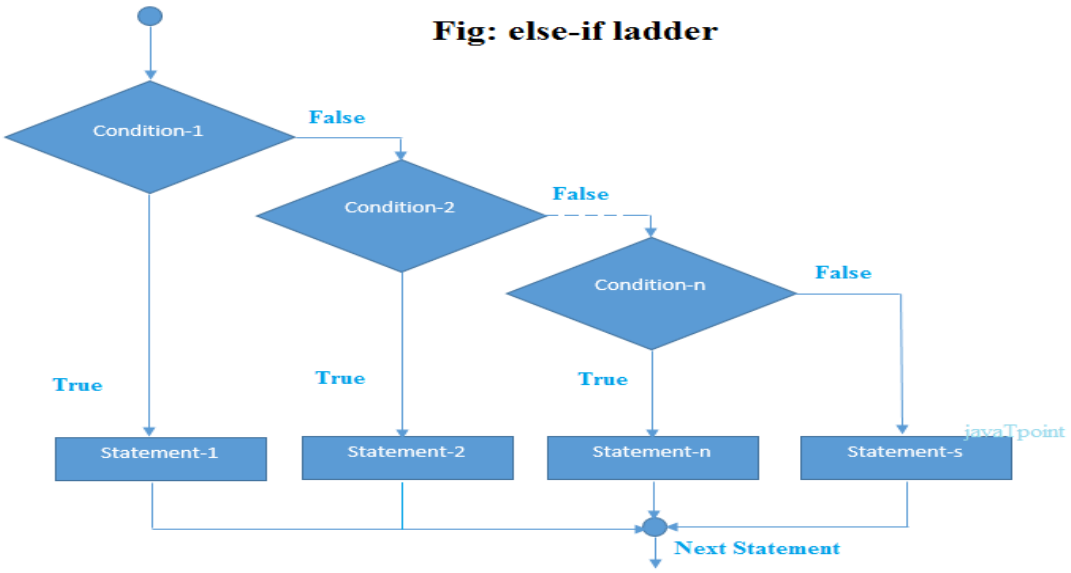


Fig: else-if ladder

d)Nested If..else statement :

When an if else statement is present inside the body of another "if" or "else" then this is called nested if else.

Syntax of Nested if else statement:

```
if(condition)
 {
      //Nested if else inside the body of "if"
      if(condition2)
       {
            //Statements inside the body of nested "if"
```

```
        }
        else
        {
            //Statements inside the body of nested "else"
        }
    }
    else
    {
       //Statements inside the body of "else"
    }
```

e) C Switch Statement:

The switch statement in C is an alternate to if-else-if ladder statement which allows us to execute multiple operations for the different possibles values of a single variable called switch variable . Here, We can define various statements in the multiple cases for the different values of a single variable.

The syntax of switch statement in c language is given below:

```
switch(expression)
{
case value1:
//code to be executed;
 break;
case value2:
 //code to be executed;
 break;
.....
```

```
default:
 code to be executed if all cases are not matched;
}
```

Rules for switch statement in C language

1) The *switch expression* must be of an integer or character type.

2) The *case value* must be an integer or character constant.

3) The *case value* can be used only inside the switch statement.

4) The *break statement* in switch case is not must. It is optional. If there is no break statement found in the case, all the cases will be executed present after the matched case. It is known as *fall through* the state of C switch statement.

Let's try to understand it by the examples. We are assuming that there are following variables.

```
1. int x,y,z;
2. char a,b;
3. float f;
```

Flowchart of switch statement in C

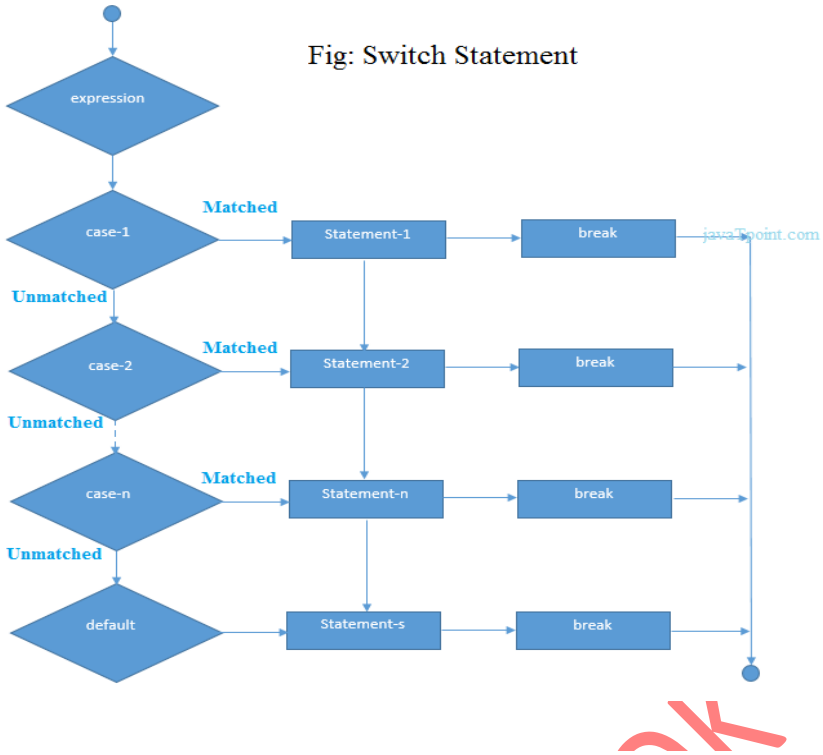


Fig: Switch Statement

java Loops

The looping can be defined as repeating the same process multiple times until a specific condition satisfies. There are three types of loops used in the C language.

Advantage of loops in C

1) It provides code reusability.

2) Using loops, we do not need to write the same code again and again.

3) Using loops, we can traverse over the elements of data structures (array or linked lists).

Types of C Loops:

There are three types of loops in C language that is given below:

1. **while**
2. **for**
3. **do while**

while loop :

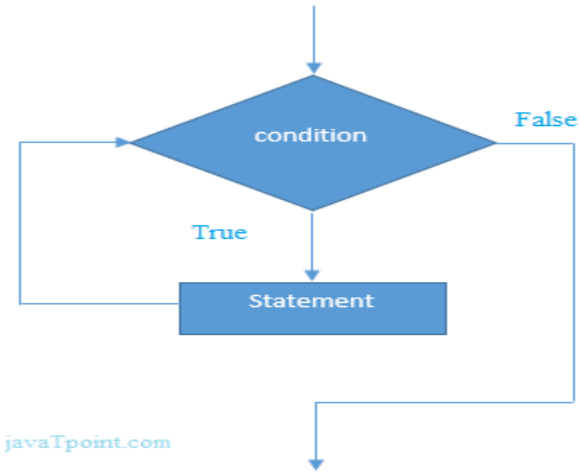
While loop is also known as a pre-tested loop. In general, a while loop allows a part of the code to be executed multiple times depending upon a given boolean condition . it is called as entry controlled loop.

Syntax of while loop in C language

The syntax of while loop in c language is given below:

```
while(condition)
{
//code to be executed
}
```

Flowchart of while loop in C



- **while loop will repeatedly execute until the given condition fails.**
- **In while loop, the condition expression is compulsory.**
- **Running a while loop without a body is possible.**
- **We can have more than one conditional expression in while loop.**
- **If the loop body contains only one statement, then the braces are optional.**

b) for loop :

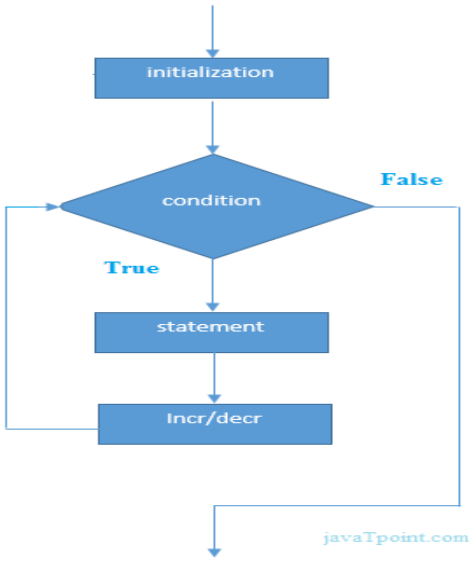
The for loop in C language is used to iterate the statements or a part of the program several times.

Syntax of for loop in C

The syntax of for loop in c language is given below:

```
for(Expression 1; Expression 2; Expression 3)
  {
     //code to be executed
  }
```

Flowchart of for loop in C



Properties of Expression 1

- The expression represents the initialization of the loop variable.
- We can initialize more than one variable in Expression 1.
- Expression 1 is optional.
- In C, we can not declare the variables in Expression 1. However, It can be an exception in some compilers.

Properties of Expression 2

- Expression 2 is a conditional expression. It checks for a specific condition to be satisfied. If it is not, the loop is terminated.

- **Expression 2 can have more than one condition. However, the loop will iterate until the last condition becomes false. Other conditions will be treated as statements.**
- **Expression 2 is optional.**
- **Expression 2 can perform the task of expression 1 and expression 3. That is, we can initialize the variable as well as update the loop variable in expression 2 itself.**
- **We can pass zero or non-zero value in expression 2. However, in C, any non-zero value is true, and zero is false by default.**

Properties of while loop

- **A conditional expression is used to check the condition. The statements defined inside the** Properties of Expression 3
- **Expression 3 is used to update the loop variable.**
- **We can update more than one variable at the same time.**
- **Expression 3 is optional.**

Loop body

The braces {} are used to define the scope of the loop. However, if the loop contains only one statement, then we don't need to use braces. A loop without a body is possible. The braces work as a block separator, i.e., the value variable declared inside for loop is valid only for that block and not outside. Consider the following example.

c)do while loop :

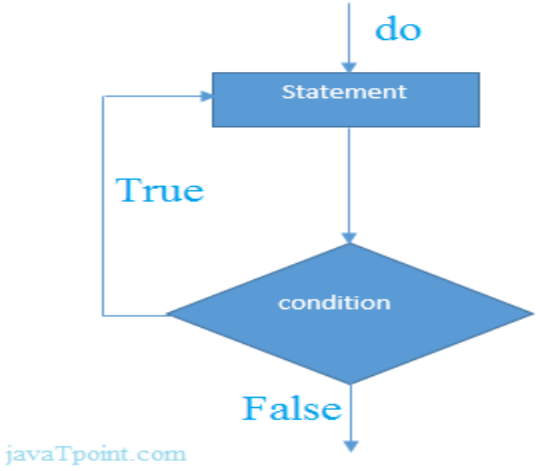
The do while loop is a post tested loop. Using the do-while loop, we can repeat the execution of several parts of the statements. The do-while loop is mainly used in the case where we need to execute the loop at least once. The do-while loop is mostly used in menu-driven programs where the termination condition depends upon the end user.

do while loop syntax

The syntax of the C language do-while loop is given below:

```
do{
//code to be executed
}while(condition);
```

Flowchart of do while loop



3)JUMP STATEMENTs

java break statement:

The break is a keyword in C which is used to bring the program control out of the loop. The break statement is used inside loops or switch statement. The break statement breaks the loop one by one, i.e., in the case of nested loops, it breaks the inner loop first and then proceeds to outer loops. The break statement in C can be used in the following two scenarios:

1. **With switch case**
2. **With loop**

Syntax:

```
1. //loop or switch case
2. break;
```

Flowchart of break in c

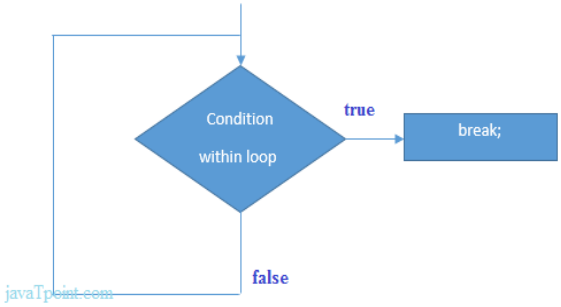


Figure: Flowchart of break statement

Continue:

The continue statement in C language is used to bring the program control to the beginning of the loop. The continue statement skips some lines

of code inside the loop and continues with the next iteration. It is mainly used for a condition so that we can skip some code for a particular condition.

Syntax:

```
//loop statements
continue;
```

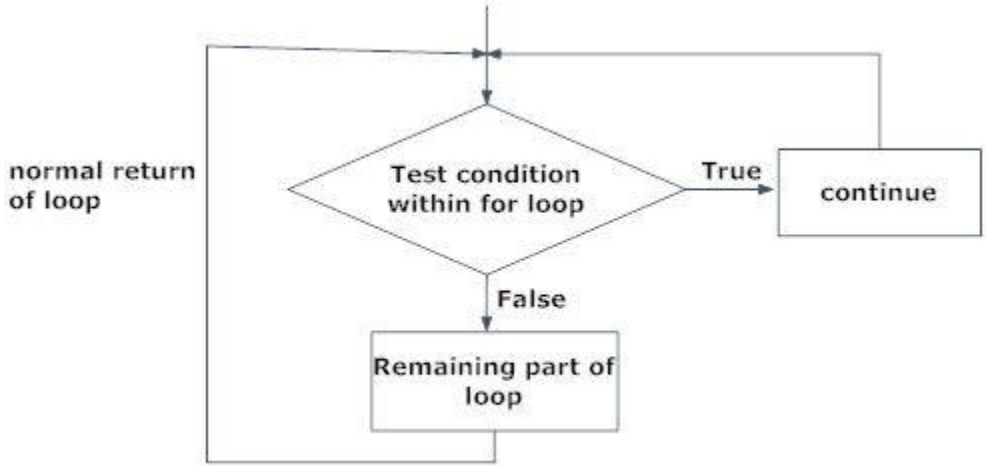


Fig: Flowchart of continue statement

return keyword in Java:

in Java, return is a reserved keyword i.e, we can't use it as an identifier. It is used to exit from a method, with or without a value. Usage of return keyword as there exist two ways as listed below as follows:

- **Case 1: Methods returning a value**
- **Case 2: Methods not returning a value**

Q) Explain input and output statements in java?

Input statements: (reading input)

scanner class (accepting input from the keyboard)

Scanner is a class in java.util package used for obtaining the input of the primitive types like int, double, etc. and strings. It is the easiest way to read input in a Java program

The Scanner class is used to get user input, and it is found in the java.util package.

To use the Scanner class, create an object of the class and use any of the available methods found in the Scanner class documentation. In our example, we will use the nextLine() method, which is used to read Strings:

input Types

In the example above, we used the nextLine() method, which is used to read Strings. To read other types, look at the table below:

Method	Description
nextBoolean()	Reads a boolean value from the user
nextByte()	Reads a byte value from the user

nextDouble()	**Reads a double value from the user**
nextFloat()	**Reads a float value from the user**
nextInt()	**Reads a int value from the user**
NextLine()	**Reads a String value from the user**
nextLong()	**Reads a long value from the user**
nextShort()	**Reads a short value from the user**

Output statements : (displaying output)

System.out :This is the standard output stream that is used to produce the result of a program on an output device like the computer screen.

Here is a list of the various print functions that we use to output statements:

- **print(): This method in Java is used to display a text on the console. This text is passed as the parameter to this method in the form of String. This method prints the text on the console and the cursor remains at the end of the text at the console. The next printing takes place from just here.**

PrintIn() :This method in Java is also used to display a text on the console. It prints the text on the console and the cursor moves to the start of the next line at the console. The next printing takes place from the next line.

Q. expalin about arrays?

An array represents a group of elements of same data type .it can store a group of elements .so,we can store a group of int values or a group of float values or a group of strings in the array,but we can not store some int values ,some float values in the arrat.in c and c++ by default ,arrays are created on static memory unless,pointers are used to create them.in java ,arrays are created on dynamic memory allocated at runtime by JVM.

TYPES OF ARRAYS:

Arrays are generally categorized into 2 parts as described here ;

1.single dimensional array or 1D array

2.multi dimensional array or 2D ,3D array

1)Single dimensional arrays :-

A one dimensional array or single dimensional array represents a row or column of elements .for example the marks obtained by a 1D array because these marks can be written as a row are as a column

declaring a single dimensional array

the syntax for declaring an array is

syn:- data type array variable name ();

or

data type [] array/variable name

ex: int a[];

or

int[]a;

here 'int' represents integers data type elements store the arrays .and variable'a'represents name of the array .and a square brases'([])' represents single dimensional arrays

<u>initilization of 1D array:</u>

at a time of declaring an array given values of that array is called array initialization

int a[]={10,20,30,40,50}

here index starts with zero and ends with size-1

the above example the single dimensional array elements are stored

10	20	30	40	50
a[0]	a[1]	a[2]	a[3]	a[4]

here a[0]=10

a[1]=20

a[2]=30

a[3]=40

a[4]=50

creating 1D array:-

creating a one dimensional array is by declaring the array first and create memory for using new operator

```
int a[];
a[]=new int[10];
```

these two statements also can be written by combining them into a single statement as

```
int a[]=new int[10];
```

here JVM create memory for storing to '10'integer elements into the array

displaying the single dimensional or 1D array:-

the following example is for displaying five elements into an array

```
class array2
{
    Public static void main(string args[])
    {
        Int a[]={10,20,30,40,50},i;
```

```
            For(i=0;i<a.length;i++)
            {
                System.out println(a[i]);
            }
        }
}
```

Output: 10

20

30

40

50

2)Multi dimensional arrays(2D,3D,.....)

Multi dimensional arrays represents 2D,3D .....arrays which are combinations for several earlier types of arrays

For example ,a two dimensional array is a combination of two one dimensional array similarly,a three dimensional array is a combination of 2 or more two dimensional array

Two dimensional arrays:-

A two dimensional array represents several rows and columns of data.for example ,the marks obtained by a group of students in five different subjects can be represented by a 2D array.

If you write the marks of three students as

50,60,55,67,70

62,65,70,70,81

72,66,77,80,69

Declaring two dimensional array:

The syntax for declaring an array is

Syn: datatype /array variable name;

Ex: int a[][]={10,20,30,40,50,60,70,80,90}

Here int represents integer datatype element store into the arrays ,and variable 'a' represents name of the array.and a double square braces "[][]" represent a two dimensional array

Initialization of 2D array :

At a time of declaring an array given values to that array is called 2D initialization

Int a[][]={10,20,30,40,50,60,70,80,90}

Or

Int a[][]={10,20,30},{40,50,60},{70,80,90}

Index of double dimensional array(2D) array:

In this rows and columns index starts with zero ends with size-1

In the below example the double dimensional array (2D) are stored

Int a [3][3]={10,20,30,40,50,60,70,80,90}

a 0 1 2

10	20	30
40	50	60
70	80	90

a[0][0]=10A

a[0][1]=20

a[0][2]=30

a[1][0]=40

a[1][1]=50

a[1][2]=60

a[2][0]=70

a[2][1]=80

a[2][2]=90

creating 2D array:

creating a two dimensional array is by declaring the array first and create a memory for using “new” operator .

syntax: datatype array name[][]=new datatype[row size][column size];

ex:

```
int a[][];
a[][]=new int[5][5]
```

or

```
int a[][]=new int[5][5];
```

here JVM create memory for storing row wise ,column wise integer elements into an array displaying double dimensional array(2D)

the following example for displaying 25 elements into an array

```
class hai
{
    Public static void main(string args[])
    {
        Int a[3][3]={10,20,30,40,50,60,70,80,90},i,j;
      for (i=0;i<=2;i++)
      {
```

```
                    For(j=0;j<=2;j++)
                    {
                        System.out.println(a[i][j]);
                    }
                System.out.println("/n");
            }
        }
}
```

Output:

```
10   20   30
40   50   60
70   80   90
```


Q) explain length of array?

Array length Attribute:

In Java, the array length is the number of elements that an array can holds. Every array has an in-built length property whose value is the size of the array. Size implies the total number of elements that an array can contain. The length property can be invoked by using the dot (.) operator followed by the array name.

int[] arr=new int[5];

int arrayLength=arr.length

In the above code snippet, arr is an array of type int that can hold 5 elements. The arrayLength is a variable that stores the length of an array.

Q) expalin about command line arguments?

Command line arguments are parameters that are supplied to the programme at the time of execution the programme.in java the main() contain one argument.that is string array type

Public static void main(string arga[])

Here args [] is a one dimensional array of string type so it can store a group of strings the users should pass the values from out side ,at the time of running at the programme command prompt as

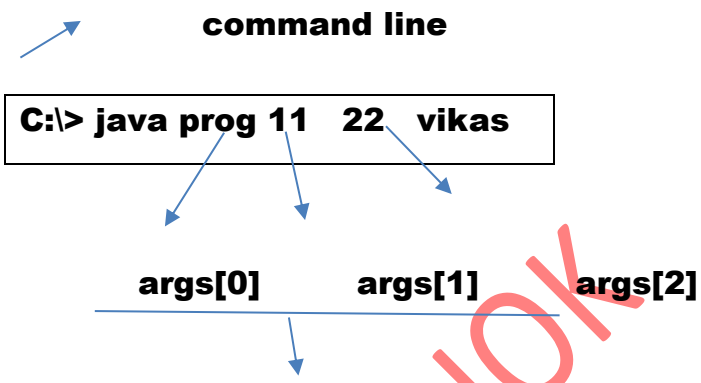
C:\>java prog 11 22 vikas

Here args[0]=11

args[1]=22

args[2]=vikas

the three values passed to main method at the time of running of are 11 ,22 and vikas these three values are automatically stored in the main() methods args [] is a string type array thus 11 is stored as a string in args[0],22 stored as string in args[1] and vikas is stored in args[2]



command line arguments

the following methods are used for convert string data type into fundamental data type

Data type	Wrapper class	Convert string data type into fundamental data type
Byte Short Int Long Float Double	Byte Short Integer Long Float Double	Public static byte parse Byte(string) Public static short parseShort(string) Public static int parseInt(string)

Char boolean	Character boolean	Public static long parseLong(string) Public static float parseFlost(string) Public static double parseDouble(string) Public static chat parseChar (string) Public static Boolean parseBoolean (string)

The following programme is enter to command line arguments as integer values and sum of two integer values

Example:

```
  Class ashok
{
        Public static void main (string args[])
        {
            String a=args[0];
            String b=args[1];
            Int x=integer.parseInt(a);
            Int y=integer.parseInt(b);
```

```
            Int z=x+y;

            System.out.println(z)

        }

} Output:  30
```

Unit-2

STRINGS

Q) what is string ? how to create a string?

String is a collection of characters enclosed with in double quotation mark .

Ex:-"rajam", "ashok"

- **String represents an array of characters ,where the last character will be '/0' (called null character) This last character being '/0' represents the end of the String . but This is not valid in java .a String is an object of String class .**
- **In java String is given as a predefined class. every predefined class is treated as user defined data type .**

Creating Strings:-

There are three ways to create Strings in java

a) **We can create a String just by assigning a group of characters to a String type variable**

```
String s; //declare String type variable

S="Hello"; //assign a group of characters to it
```

Preceding two statements can be combine and written as

String s=”Hello “

In This, case JVM creates an object and stores the String “Hello” in that object.

b) We can create an object to the String class by allocating memory using new operator .This is just like creating an object to any class like given here

String s=new String(“Hello”);

Here ,we are doing two things . first we are creating object using new operator .then we are storing the Strings ”Hello “ into the object.

c) The third way of creating the Strings is by converting character arrays into Strings. let us take a character type array arr [] with some characters , as :

char arr[]={‘c’ ,‘h’ ,‘a’, ’l’, ’r’, ’s’};

now create a String object ,by passing the array name to it , as ;

String s=new String (arr);

Q) explain methoeds of String ?

String class methods:-

1. **public int length ():- This method is used for return number of character in the String**

Ex : String a=”Hello”;

int x = a .length ();

System .out.println(x);

Output: 5

2.public String concat():- This method is used for adding two Strings.

String concat(String)

Ex : String a=”Hello”;

String b=”Hai”;

String x=a.concat(b);

System.out.println(x);

Output: HelloHai

3.Public String toUpperCase(): This method is used for convert the given String into upper case characters .

String a=”Hello”;

String x= a.toUpperCase();

System.out.println(x);

Output : HELLO

4.public String toLowerCase():- This method using for convert the given String into lowercase characters

Ex: String a=”Hello” ;

String x=a.tolowercase();

System.out.println(x);

Output : hello

5.public char charAt(int) : This method returns a particular character depending upon the Given index value.

Ex:String a=”degree collage”;

Char x=a.charAt(6);

System.out.println(x);

Output: e

6. public String replace(char c1,char c2):- This method is used for replace the character c2 instead of character c1 in the given String .

Ex : String a=”degree collage”;

String x=a.replace(‘c’ ,‘x’);

System.out.println(x);

Output: dxgrxx collxgx

7.public String trim():- This method is used for eliminating the space in before and after the String .

Ex: String s=” degree”;

```
String s=s. trim ();

System.out.println(x);
```

Output :degree

8. public boolean equals(String):-This method is used for compare two Strings with case sensitively .if two strings are equal then returns true otherwise returns false .

Ex:

```
String s1="Hello";

String s2="Hai";

boolean x=s1.equals(s2);

System.out.println(x);
```

Output:

false

9 . public boolean equalsIgnoreCase(string):-

This method is used for compare two Strings with case insensitively .if two strings are equals then the result true otherwise false

Ex:

```
String s1="Hello";
```

```
String s2=”hai”;

boolean x=s1.equalignore(s2);

System.out. println(x);
```

Output :

false

9 . public String substring(int):-

This method is returns some part of the string in the given string based on given index value

```
ex:String s=”sltn degree collage” ;

String x=s.substring(5);

System.out.println(x);
```

Output:degree collage

10.public boolian startswith (String):

This method check if the given string starts with the specified string (or) not .

```
Ex: string s=”sltn degree college”;

boolean x=s.startsWith(“sltn”);

System.out.println(x);
```

Output:true

12. public boolean endsWith(string):-

This method check if the given string ends with the specified string (or) not .

ex:-string s=”sltn degree college”;

boolean x=s.endswith(“sltn”);

System.out.println(x);

Output : false

Q) explain Java String comparision ?

Java String compare:

We can compare String in Java on the basis of content and reference.

here are three ways to compare String in Java:

1. **By Using equals() Method**
2. **By Using == Operator**

1) By Using equals() Method:

The String class equals() method compares the original content of the string. It compares values of string for equality. String class provides the following two methods:

- **public boolean equals(Object another) compares the string to the specified object. returns true if both strings are equal . otherwise returns false.**
- **public boolean equalsIgnoreCase(String another): compares the string to another string, ignoring case. returns true if both strings are equal . otherwise returns false.**
-

```
class Teststringcomparison1
{
public static void main(String args[])
{
String s1="Sachin";
String s2="Sachin";
System.out.println(s1.equals(s2));//true

}
 }
```

Output:

true

example2:

```
class Teststringcomparison2

{
```

```
 public static void main(String args[])

{

  String s1="Sachin";

  String s2="SACHIN";

 System.out.println(s1.equals(s2));//false

  System.out.println(s1.equalsIgnoreCase(s2));//true

 }

}
```


Output:

```
false
true
```

2) By Using == operator:

The == operator compares references not values.

```
class Teststringcomparison3

{

 public static void main(String args[])

{
```

```
String s1="Sachin";

String s2="Sachin";

String s3=new String("Sachin");

System.out.println(s1==s2);//true (because both refer to same inst
ance)

System.out.println(s1==s3);//false(because s3 refers to instance cr
eated in nonpool)

 }

}
```


Output:

```
true
false
```

Suppose s1 and s2 are two String objects. If:

- **s1 == s2 : The method returns 0.**
- **s1 > s2 : The method returns a positive value.**
- **s1 < s2 : The method returns a negative value.**

Q) immutable String in Java?

immutable String in Java:

String references are used to store various attributes like username, password, etc. In Java, String objects are immutable. Immutable simply means unmodifiable or unchangeable.

Once String object is created its data or state can't be changed but a new String object is created.

Let's try to understand the concept of immutability by the example given below:

Testimmutablestring.java

```
 class Testimmutablestring
{
  public static void main(String args[])
{
    String s="Sachin";
    s.concat(" Tendulkar");//concat() method appends the string at the end
    System.out.println(s);//will print Sachin because strings are immutable objects
```

```
    }

}
```

Output:

Sachin

Q) what is the difference between procedure oriented and object oriented programming language?

procedural Oriented Programming	Object-Oriented Programming
In procedural programming, the program is divided into small parts called *functions*.	In object-oriented programming, the program is divided into small parts called *objects*.
Procedural programming follows a *top-down approach*.	Object-oriented programming follows a *bottom-up approach*.
There is no access specifier in procedural programming.	Object-oriented programming has access specifiers like private, public, protected, etc.
Adding new data and functions is not easy.	Adding new data and function is easy.
In procedural programming, overloading is not possible.	Overloading is possible in object-oriented programming.
In object-oriented programming, inheritance is used.	the concept of data hiding and

procedural Oriented Programming	Object-Oriented Programming
In procedural programming, the function is more important than the data.	In object-oriented programming, data is more important than function.
Procedural programming is used for designing medium-sized programs.	Object-oriented programming is used for designing large and complex programs.
Code reusability absent in procedural programming,	Code reusability present in object-oriented programming.
Examples: C, FORTRAN, Pascal, Basic, etc.	Examples: C++, Java, Python, C#, etc.

Q) explain about oops principals?

Object means real world entity.Any language is said to be object oriented programming language,it has to satisfy the following 8 principals.

1.class

2. object

3.data abstraction

4.data encapsulation

5.inheritance

6.polymorphism

7.dynamic binding

8.message passing

1) Class

class is a user defined datatype.

class is a collection of data members and methods without class there is no java programme,that means each and every java programme must starts with a concept is called class

a class is a blue print it is used foe create number of objects.

Ex: fruit is a class then apple ,mango are objects of fruits.

Syntax:- class classname

```
{
        Data members;
        Methods;
}
```

in the above syntax class is a keyword.

class is keyword used for developing a java programme.

class name is any java variable name

every class name in java is treated has a user defined data type.

data members(variables) of a class will be selected based on the class name of user requirement.

in oops we have two types of methods 1.member methodes 2. Non member methodes

a member methode is one which is defined inside the class.a non member methode is one which is defined out side of the class.java only support member methods.

when ever we defined a class there is no memory space is created for the data members of the class the memory space created for the data members of the class,when ever an object created to the class

Ex: class classname

```
{
    Public static void main (string args[])
      {
        Int a=10,b=20;
        System.out.println(a);
        System.out.println(b);
    }
}
```

Output: 10

20

2)Object:

enter oop methodology has been derived from the single root concept is called object

an object is instance of a class

an object is anything that really exist in the world everything that we see physically comes into the object for example a table a ball,car.....

without class depending an object is not possible.an object is created from a class

The difference between class and object is class is a model for creating object

And does not exit reality.an object is any thing that's really exist.both the class and object contain data members and methods.

Creating an object :

If you want create an object in java using "new" operator .new operator is dynamic memory allocation operator.without class creating an object is not possible

Syntax: class objectname=new classname();

Or

Classname objectname: //object declaration

Objectname=new classname(); //object initialization

Ex: hai h=new hai();

Or

Hai h;

H=new hai();

Here hai is a class name

Ex: example programme for find some of two numbers using object

```
Class ashok
{
        Int a=10,b=20;
            Void sum()
        {
             System.out.println("sum of two numbers:"+(a+b));
        }
}
Class ashok1
{
        Public static void main(string args[])
```

```
        {
            Ashok a=new ashok();
            a.sum();
        }
}
```

Output: sum of two numbers:30

3)Abstraction:

Abstraction is hiding the internal details and showing only esential functionality. In the abstraction concept, we do not show the actual implemention to the end user, instead we provide only esential things. For example, if we want to drive a car, we does not need to know about the internal functionality like how wheel system works? how brake system works? how music system works? etc.

4)Encapsulation:

Encapsulation is the process of combining data and code into a single unit (object / class). In OOP, every object is associated with its data and code. In programming, data is defined as variables and code is defined as methods. The java programming language uses the class concept to implement encapsulation.

5)Inheritance:

inheritence is a mechanism used for inherit(acquiring)the data members and methods from one class to another class using "extends" keyword

inheritence is a ,mainly called as reusability and extendability .

which class is given data members and methods to another class is called "base class" or "super class"or "parent class"

which class getting data members and methods from another class is called "derived class" or "sub class "or "child class"

Syntax for inherit one class features into another class

```
Class classname2 extends classname1
{
    Data members;
    Methods;
}
```

Here classname2 is a derived class or sub class or child class

Classname1 is a base class or super class parent class

polymorphism:

Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So polymorphism means many forms.

The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

Real-life Illustration: Polymorphism

A person at the same time can have different characteristics. Like a man at the same time is a father, a husband, an employee. So the same person possesses different behavior in different situations. This is called polymorphism.

Polymorphism is considered one of the important features of Object-Oriented Programming. Polymorphism allows us to perform a single action in different ways.

In Java polymorphism is mainly divided into two types:

- **Compile-time Polymorphism**
- **Runtime Polymorphism**

7) Dynamic binding:

When type of the object is determined at run-time, it is known as dynamic binding.

CONSTRUCTORS

Q) Explain about constructers ?

Ans: constructers :-

- **Constructers name is the same name of the class name and does not contain any return type even "void "also.**
- **It is syntactically similar to a method .**

Properties of constructers :-

- **Constructer name is same name of the class name and constructer does not return any value even void also.**
- **Constructer are called automatically whenever an object Is created to the class .**
- **Constructer should not be static .**
- **Constructer should not be private.**

Constructers are two types:-1) Default constructer .

2)Parameterized constructer.

1)Default constructer:-

Here the constructer name Is the same name of the class and does not contain any perameters is default constructer.

Syn:- class class name

```
                {
               ..............
               ..............
        Class name() ->default constructer.

        {
        ..........................................................
        ..........................................................
         }
}
```

```
Ex:- import java.util.*;
     Class hello
{
  Scanner s=new scanner(System.in);
System.out.println("enter  2 numbers");
 int a=s.nextint();
 int b=s.nextint();
Hello()
  {
     System.out.println("sum of a,b are:"+(a+b));
   }
```

```
}

Class hello1

{

 Public static void main(String args[])

  {

   Hello h=new hello();

  }

}
```

Compilation: java hello1.java

Execution: java hello1

Enter 2 numbers : 10 20

Output: sum of a,b are:30

Access specifiers in java

Q)explain access specifiers in java?

Four Types of Access Modifiers

- **Private: We can access the private modifier only within the same class and not from outside the class.**
- **Default: We can access the default modifier only within the same package and not from outside the package. And also, if we do not specify any access modifier it will automatically consider it as default.**
- **Protected: We can access the protected modifier within the same package and also from outside the package with the help of the child class. If we do not make the child class, we cannot access it from outside the package. So inheritance is a must for accessing it from outside the package.**
- **Public: We can access the public modifier from anywhere. We can access public modifiers from within the class as well as from outside the class and also within the package and outside the package.**

now let us understand the scope of these access modifiers with the help of a table:

Accessibility	Private	Default	Protected	Public

Same Package	Same Class	Yes	Yes	Yes	Yes
Without Inheritance	No	Yes	Yes	Yes	
With Inheritance	No	Yes	Yes	Yes	
Different Package	Without Inheritance	No	No	No	Yes
With Inheritance	No	No	Yes	Yes	

Q) explain methoeds in java?

Methoeds in java:

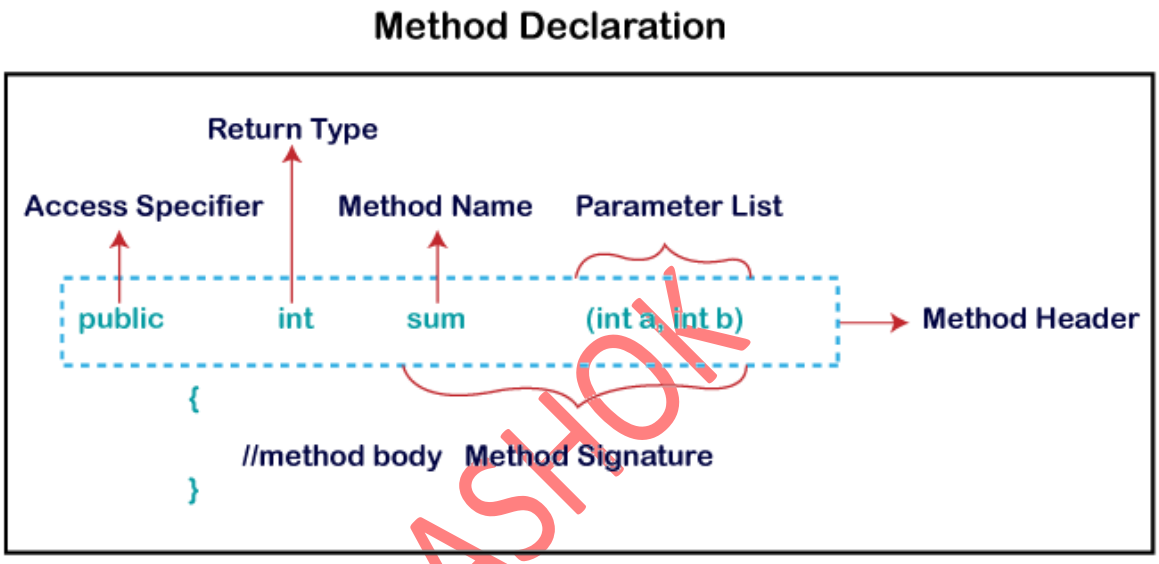
A method is a block of code or collection of statements or a set of code grouped together to perform a certain task or operation. It is used to achieve the reusability of code. We write a method once and use it many times.

A method is a block of code which only runs when it is called.

You can pass data, known as parameters, into a method.

Method Declaration:

The method declaration provides information about method attributes, such as visibility(access modifier), return-type, name, and arguments. It has six components that are known as method header, as we have shown in the following figure.



Create a Method:

A method must be declared within a class. It is defined with the name of the method, followed by parentheses (). Java provides some pre-defined methods, such as System.out.println(), but you can also create your own methods to perform certain actions:

Example

Create a method inside Main:

```
public class Main {
  void myMethod()
 {
    // code to be executed
  }
}
```

Example Explained

- **myMethod() is the name of the method.**
- **void means that this method does not have a return value. You will learn more about return values later in this chapter.**

Call a Method:

To call a method in Java, write the method's name followed by two parentheses () and a semicolon;

In the following example, myMethod() is used to print a text (the action), when it is called:

```
public class Main
{
  static void myMethod()
  {
```

```
    System.out.println("I just got executed!");
  }

  public static void main(String[] args)

  {
    myMethod();
  }
}
```

A method can also be called multiple times:

```
public class Main {
  static void myMethod() {
    System.out.println("I just got executed!");
  }

  public static void main(String[] args) {
    myMethod();
    myMethod();
    myMethod();
  }
}
```


Java Method Parameters:

Parameters and Arguments

Information can be passed to methods as parameter. Parameters act as variables inside the method.

Parameters are specified after the method name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma.

The following example has a method that takes a String called fname as parameter. When the method is called, we pass along a first name, which is used inside the method to print the full name:

Example:

```
public class Main
{
  static void myMethod(String fname)

  {
    System.out.println(fname);
  }

  public static void main(String[] args)
    {
    myMethod("ashok");
    myMethod("balu");
    myMethod("ravi");
  }
```

}

Types of Method:

There are two types of methods in Java:

- **Predefined Method**
- **User-defined Method**

Predefined Method

In Java, predefined methods are the method that is already defined in the Java class libraries is known as predefined methods. It is also known as the standard library method or built-in method. We can directly use these methods just by calling them in the program at any point. Some pre-defined methods are length(), equals(), compareTo(), sqrt(), etc. When we call any of the predefined methods in our program, a series of codes related to the corresponding method runs in the background that is already stored in the library.

Each and every predefined method is defined inside a class. Such as print() method is defined in the java.io.PrintStream class. It prints the statement that we write inside the method. For example, print("Java"), it prints Java on the console.

Let's see an example of the predefined method.

Demo.java

```
public class Demo
{
public static void main(String[] args)
{
// using the max() method of Math class
System.out.print("The maximum number is: " + Math.max(9,7));
}
}
```

Output:

```
The maximum number is: 9
```


In the above example, we have used three predefined methods main(), print(), and max(). We have used these methods directly without declaration because they are predefined. The print() method is a method of PrintStream class that prints the result on the console. The max() method is a method of the Math class that returns the greater of two numbers.

User-defined Method

The method written by the user or programmer is known as a user-defined method. These methods are modified according to the requirement.

How to Create a User-defined Method:

Let's create a user defined method that checks the number is even or odd. First, we will define the method.

```
//user defined method
public static void findEvenOdd(int num)
{
//method body
if(num%2==0)
System.out.println(num+" is even");
else
System.out.println(num+" is odd");
}
```


We have defined the above method named findevenodd(). It has a parameter num of type int. The method does not return any value

that's why we have used void. The method body contains the steps to check the number is even or odd. If the number is even, it prints the number is even, else prints the number is odd.

Static Method:

A method that has static keyword is known as static method. In other words, a method that belongs to a class rather than an instance(object) of a class is known as a static method. We can also create a static method by using the keyword static before the method name.

The main advantage of a static method is that we can call it without creating an object. It can access static data members and also change the value of it . It is invoked by using the class name. The best example of a static method is the main() method.

Instance Method:

The method of the class is known as an instance method. It is a non-static method defined in the class. Before calling or invoking the instance method, it is necessary to create an object of its class.

There are two types of instance method:

- **Accessor Method**
- **Mutator Method**

Accessor Method: The method(s) that reads the instance variable(s) is known as the accessor method. We can easily identify it because the method is prefixed with the word get. It is also known as getters.

Example

```
1. public int getId()
2. {
3. return Id;
4. }
```

Mutator Method: The method(s) read the instance variable(s) and also modify the values. We can easily identify it because the method is prefixed with the word set. It is also known as setters.

Example

```
1. public void setRoll(int roll)
2. {
3. this.roll = roll;
4. }
```

Abstract Method:

The method that does not has method body is known as abstract method. In other words, without an implementation is known as abstract method. It always declares in the abstract class. It means the class itself must be abstract if it has abstract method. To create an abstract method, we use the keyword abstract.

Syntax

1. **abstract void method_name();**

Factory method

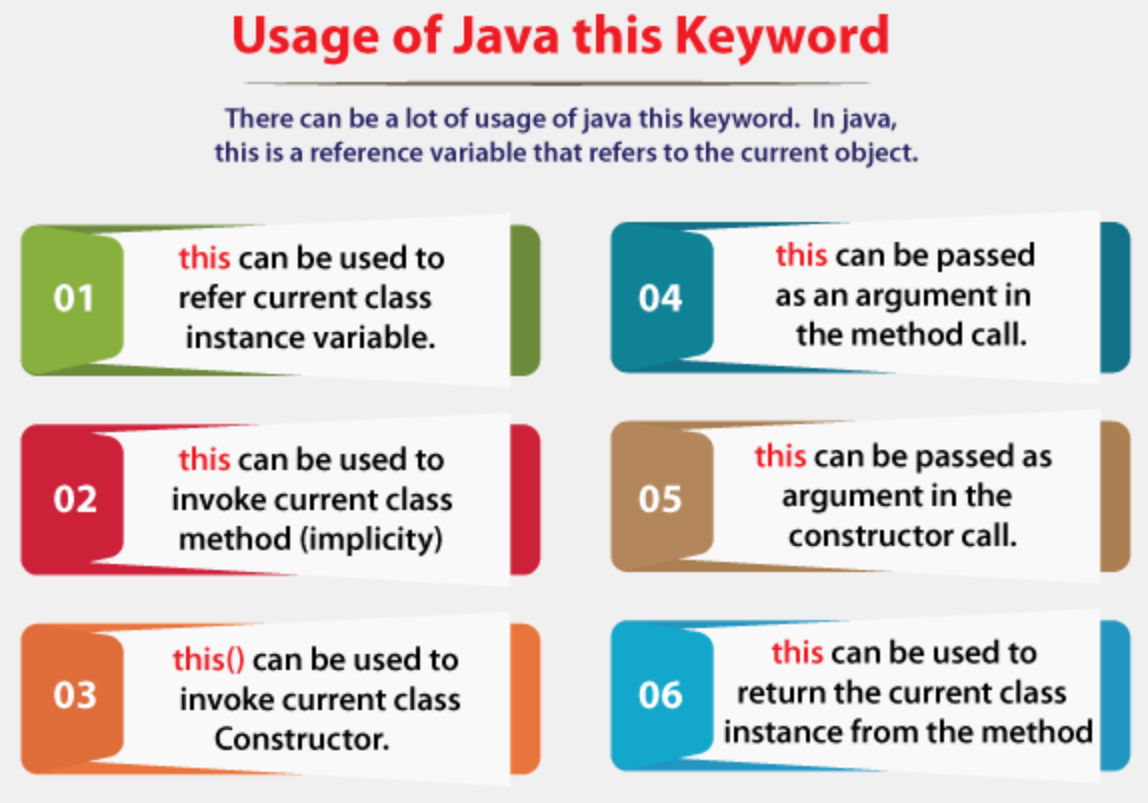
It is a method that returns an object to the class to which it belongs. All static methods are factory methods. For example, NumberFormat obj = NumberFormat.getNumberInstance();

this keyword

q) explain “this” keyword in Java?

There can be a lot of usage of Java this keyword. In Java, this is a reference variable that refers to the current object.

Suggestion: If you are beginner to java, lookup only three usages of this keyword.



1) this: to refer current class instance variable:

The this keyword can be used to refer current class instance variable. If there is ambiguity between the instance variables and parameters, this keyword resolves the problem of ambiguity.

```
class This1

{

    int a=10;

   void disp(int a)

   {
```

```
        System.out.println(a);

       System.out.println(this.a);

   }

}

Class Hello

{

    Public static void main(String args[])

    {

        This1 t=new This();

        t.disp();

   }

}
```


b) this() : to invoke current class constructor:

The this() constructor call can be used to invoke the current class constructor. It is used to reuse the constructor. In other words, it is used for constructor chaining.

```
    class A
```

```
{
  A()
  {
    System.out.println("hello a");
  }
  A(int x)
  {
    this();
     System.out.println(x);
  }
}

class TestThis5
{
public static void main(String args[])
  {
    A a=new A(10);
```

```
        }

    }
```

Output:

```
hello a
10
```


Super Keyword

Q)Explain "Super" Keyword in Java?

The super keyword in Java is a reference variable which is used to refer immediate parent class object.

Usage of Java super Keyword:

1. **super can be used to refer immediate parent class instance variable.**
2. **super can be used to invoke immediate parent class method.**

3. super() can be used to invoke immediate parent class constructor.

1) super is used to refer immediate parent class instance variable.

e can use super keyword to access the data member or field of parent class. It is used if parent class and child class have same fields.

```
class Animal
{
String color="white";
}
class Dog extends Animal
{
String color="black";
void printColor()
{
  System.out.println(color);//prints color of Dog class
  System.out.println(super.color);//prints color of Animal class
}
```

```
    }

 class TestSuper1
{
   public static void main(String args[])
{
   Dog d=new Dog();
 d.printColor();

 }

}
```

black

white

super is used to invoke parent class constructor:

The super keyword can also be used to invoke the parent class constructor. Let's see a simple example:

```
    class Animal

    {

    Animal()

    {

       System.out.println("animal is created");}
```

```
}

class Dog extends Animal{

Dog()

{

super();

System.out.println("dog is created");

}

}

class TestSuper3{

public static void main(String args[]){

Dog d=new Dog();

}}
```

Output:

```
animal is created
dog is created
```

Inhertince

Inheritance is a mechanism used for inherit (acquiring)the data members and methoda from one class to another class using'extends'keyword

Or

Inheritance is process of a deriving a new class fron an existing class

The main purpose of inheritance is reusability.

Base class : which class given features to another class is called as base class or super class or parent class

Derived class : which class is getting the features from another class is called derived class or sub class or child class

"extends" extends is keyword used for getting the features from one class to another class

Syntax foe inheriting the features from base class to derived class:

```
Class<class name2> extends <class name 1>
{
        Variable declaration/initialization;
        Method definition ;

}
```

Here class name1----base class/parent class/super class

Class name 2-----derived class/child class/sub class

"extends" is a keyword is used for inheriting the data members and methods from base class to derived class

Types of inheritance:

Based on taking the features from base class to the derived class ,in java we have 5 types of inheritance.these are

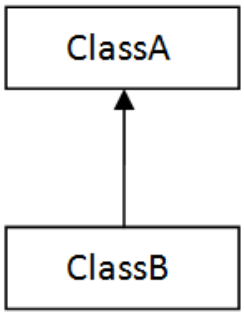
1.single inheritance

2.multilevel inheritance

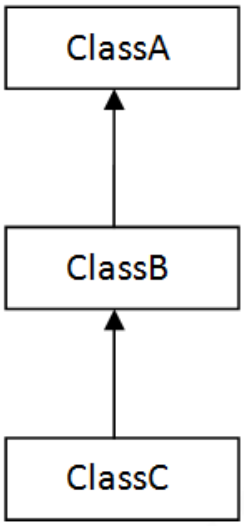
3.hierarchal inheritance

4.multiple inheritance

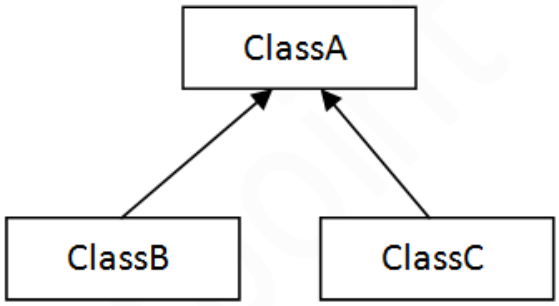
5.hybrid inheritance



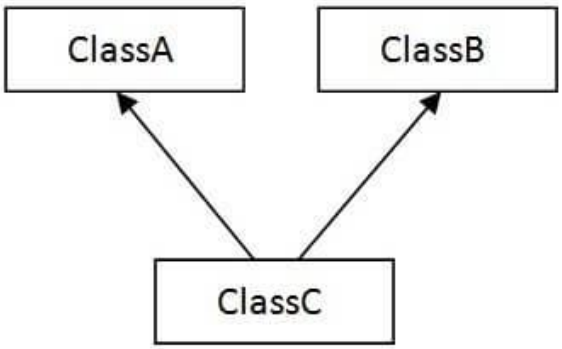
1) Single



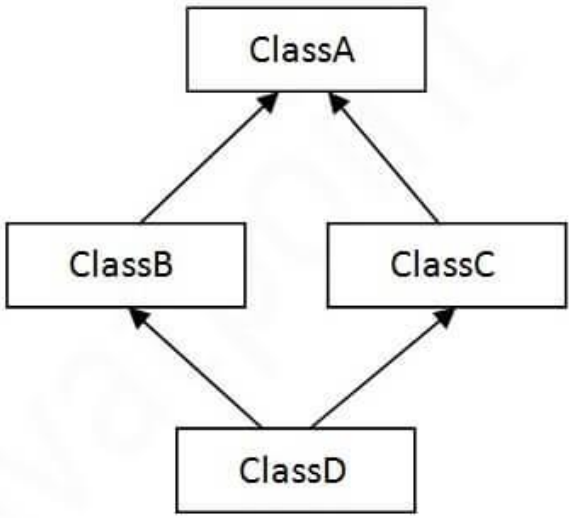
2) Multilevel



3) Hierarchical



4) Multiple



5) Hybrid

1.single inheritance:

Single inheritance is one which there exists single base class and single derived class

Syn:-

```
Class A
{
    Date members ;
     Methods;
}
Class B extends classA
{
   Data members;
   Methods;
```

```
}
```

In the above syntax

Class A is a parent/super /base class,

Class B is a child/sub /derived class

Example programme for single inheritance:

```
Class A
{
    Void sum()
    {
        System.out.println("sum is :"+(a+b));
    }
}
Class B extends class A
{
    Void sub()
    {
        System.out.println("sub is :"+(a-b));
    }
}
Class inhr1
{
   Public static void main(string args[])
```

```
    {
        B b=new();
        b.sum();
        b.sub();
    }
}
```

Output: sum is:30

Sub is : -10

2.multilevel inheritance:

Multilevel inheritance is one which there exists single base class,single derived class and n number of intermediate base classes.

Syntax:

```
Class A
{
        Data members;
        Methods;
}
Class B extends  A
{
        Data members;
        Methods;
}
```

```
Class C extends  B
 {
       Date members ;
        Methods;
  }
```

Here class B is one ,in one contest it acts as base class and in another contest acts as derived class.

EXAMPLE PROGRAMME FOR MULTIPLE INHERITANCE:

```
   Class A
   {
         Int a=1=,b=20;
        Void sum()
        {
             System.out.println("sum is:"+(a+b));
         }
}
Class B extends  A
{
 Void sub()
{
      System .out println("sub is:"+(a-b));
```

```
}
}
Class C extends  B
{
    Void mul()
            System.out.println("mul is:"+(a*b));
     }
}
Class inhr 2
{
    Public static void main (String args[])
     {
        C c=new C();
        c.sum();
       c. sub();
        c.mul();
   }
}
```

Out put: sum is : 30

Sub is : -10

Mul is :200

3.hierarchal inheritance:

Hierarchal inheritance is one in which there exists single base class multiple no of derived class .

Syntax:

```
Class A
{
 Data members ;
  Methods;
}
Class B extends A
{
 Data members;
  Methods;
}
Class C extends A
{
 Data members;
 Methods;
}
```


4.multiple inheritance:

Multiple inheritance is one in which there exists n number of base classes and single derived classes

Multiple inheritance or not supported by java through classes but it is supported by java through the concept of interfaces

```
Class A
{
   ::::::::::::
}
Class B
{
   ::::::::::::::
}
Class C extends A,B
{
   ::::::::::::
}
```


5.hybrid inheritance:

When deriving a sub class of an multiple super classes them there is a chance of some data members one methods an existing in all or some of the super classes it is combination of two or more inheritance are called hybrid inheritance

unit-3

polymorphism

Q) what is polymorphism ? explain types of polymorphism ?

Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So polymorphism means many forms.

The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

Real-life Illustration:

Polymorphism

A person at the same time can have different characteristics. Like a man at the same time is a father, a husband, an employee. So the same person possesses different behavior in different situations. This is called polymorphism.

Polymorphism is considered one of the important features of Object-Oriented Programming. Polymorphism allows us to perform a single action in different ways.

In Java polymorphism is mainly divided into two types:

- **Compile-time Polymorphism (Method overloading)**
- **Runtime Polymorphism (Method overriding)**

Compile-time Polymorphism (Method Overloading):

When there are multiple methoeds with the same name but different parameters then these methoeds are said to be overloaded. methoeds can be overloaded by change in the number of arguments or/and a change in the type of arguments..

).

Method overloading is one of the ways through which java supports polymorphism . method overloading can be done by changing member of argument or by changing the data type of arguments . If 2 or more method have same name and same parameter list but differs in return type are not said to be over loaded method


```
Class calculate
{
        Void sum(int a,int b)
        {
            System.out.println("sum is:"+(a+b));
        }
        Void sum (float a,float b)
```

```
        {
            System.out.println("sum is:"+(a+b));
        }
}
Class calculate1
{
    Public static void main (string args[])
        {
            Calculate cal=new calculate();
            Cal.sum(8,5);
             Cal.sum(4.6f,3.8f);
        }
}
```

Output: sum is :13

Sum is :8.4

Example of Java Runtime Polymorphism(methoed overring)

In this example, we are creating two classes Bike and Splendor. Splendor class extends Bike class and overrides its run() method. We are calling the run method by the reference variable of Parent class. Since it refers to the subclass object and subclass method overrides the Parent class method, the subclass method is invoked at runtime.

Since method invocation is determined by the JVM not compiler, it is known as runtime polymorphism.

```
class Bike

{

  void run()

  {

    System.out.println("running");

    }

}

class Splendor extends Bike

 {

    void run()
```

```
        {
            System.out.println("running safely with 60km");
         }

  public static void main(String args[])
 {
    Bike b = new Splendor();//upcasting
    b.run();
  }
}
```

running safely with 60km.

Dynamic binding:

When type of the object is determined at run-time, it is known as dynamic binding.

```
class Bike
{
  void run(){System.out.println("running");}
```

```
}

class Splendor extends Bike{

  void run(){System.out.println("running safely with 60km");}


  public static void main(String args[]){

    Bike b = new Splendor();//upcasting

    b.run();

  }

}
```

running safely with 60km.

Final keyword:

final **keyword is used in different contexts. First of all, *final* is a non-access modifier applicable only to a variable, a method, or a class. The following are different contexts where final is used.**

Final Variables

When a variable is declared with the *final keyword*, its value can't be modified, essentially, a constant. This also means that you must initialize a final variable.

```
inal int THRESHOLD = 5;
static final double PI = 3.141592653589793;
// Final static variable PI
```

Final classes

When a class is declared with *final* keyword, it is called a final class. A final class cannot be extended(inherited).

```
Final class classname
{
   ::::::::::::::::::::;
    ::::::::::::::::::;
}
```


Final Methods

When a method is declared with *final* keyword, it is called a final method. A final method cannot be overridden. The Object class does this—a number of its methods are final.

```
Class Hello
```

```
{
    Final void disp()
   {
          ::::::::::::::::::::::;;
    }
}
```

Q) explain typecasing ?

Type Casting in Java

In Java, type casting is a method or process that converts a data type into another data type in both ways manually and automatically. The automatic conversion is done by the compiler and manual conversion performed by the programmer.

Type casting:

Convert a value from one data type to another data type is known as type casting.

Types of Type Casting

There are two types of type casting:

- **Widening Type Casting(implicit type convertion)**
- **Narrowing Type Casting(explicit type convertion)**

Widening Type Casting

Converting a lower data type into a higher one is called widening type casting. It is also known as implicit conversion or casting down. It is done automatically. It is safe because there is no chance to lose data. It takes place when:

- **both data types must be compatible with each other.**
- **The target type must be larger than the source type.**

1. **byte -> short -> char -> int -> long -> float -> double**

```
public class WideningTypeCastingExample
 {
 public static void main(String[] args)
 {
 int x = 7;
 //automatically converts the integer type into long type
 long y = x;
 //automatically converts the long type into float type
 float z = y;
 System.out.println("Before conversion, int value "+x);
 System.out.println("After conversion, long value "+y);
 System.out.println("After conversion, float value "+z);
```

```
}
```

Narrowing Type Casting:

Converting a higher data type into a lower one is called narrowing type casting. It is also known as explicit conversion or casting up. It is done manually by the programmer. If we do not perform casting then the compiler reports a compile-time error.

double -> float -> long -> int -> char -> short -> byte

```
public class NarrowingTypeCastingExample
{
public static void main(String args[])
{
double d = 166.66;
//converting double data type into long data type
long l = (long)d;
//converting long data type into int data type
int i = (int)l;
System.out.println("Before conversion: "+d);
```

```
//fractional part lost

System.out.println("After conversion into long type: "+l);

//fractional part lost

System.out.println("After conversion into int type: "+i);

}

}
```

Abstract classes

Q) expalin abstract classes?

Classes are two types 1.concrete class

2.abstract class

Concrete class:-

Which class contain fully defined methods is called concrete class

Syntax: class<class name>

```
{

   defined methods;

}
```

Ex: class hai

```
{
    Void disp()
      {
        ---------
        ---------
    }
    Void disp1()
    {
        ---------
        ----------
    }
}
```


We can create an object to the concrete directly .

```
hai n=new hai();        //valid
```

2.Abstract class:-

If a class contain zero or more undefined (abstract)method is called abstract class

A class that is declared which abstract keyword is known as abstract class in java

Syntax: abstract class<class name>

{

Abstract methods;

}

Abstract method:-

A method that is declared is abstract and does not have implementation is known as abstract method

Syntax: abstract returntype methode-name;

Ex: abstract void sum();

Example:

Abstract class hai

{

Abstract Void disp();

}

We can not create an object to the abstract class directly but we can create an object to the abstract class indirectly

Hai h=new Hai(); //invalid

Abstract derived class:-

If abstract base class contain n number of abstract methods .then the derived class extends abstract base class,in this time the deriverd class define all n number of abstract methods in base class then the derived is concrete class.if the derived class is not defined atleast one abstract method in n number abstract methods then the derived class is abstract derived class.not possible to given final keyword before abstract class name.that means abstract class can not be made as final classes.

We can create an object to the abstract class indirectly

Syntax : abstract classname objectname=new derived classname ();

Programme:

```
Abstract class hai
{
Abstract void sum();
}
Class hai1 extends hai
{
Int a=10,b=20;
Void sum()
```

```
        {
            System.out.println("sum is:"+(a+b));
        }
    }
Class hai2 extends hai
{
        Float a=10.35f,b=20.45f;
        Void sum()
        {
            System.out.println("sum is:"+(a+b));
        }
}
Class hello
{
        Public static void main(string args[])
        {
            hai h=new hai1();
            h.sum();
```

```
        hai h1=new hai2();

        h1.sum();

    }

}
```

Compalition:java hello.java

Execution: java hello

Out put: sum is:30

Sum is 30.80

Interfaces

Q) expalin about interfaces?

- **An interface is one which contains only purely undefined methods .**
- **Interface are basically used to develop user defined datatype**
- **With respect to interfaces we can achive the concept of multiple inheritance**
- **Syntax for defining an inter face**

  Interface<interface name>

  {

```
Variable initialization;
Method declaration;
}
```

Ex:

```
Interface hai
{
    Int a=10,b=20;
    Void disp();
    Void disp1();
}
```

- **Interface is keyword which is used for defining a interface ."interface name"represent a java valid variable name**
- **Variable initialization represents the type of date members which we use as a part of interface**
- **Method declaration represents the type of methods which we use apart of interface**
- **What ever the variables we write in the interface they are by default belongs to public static final *** data members;**

  ***** represent data type**
- **Hence ,by default all the methods of interface belong to public abstract methods**
- **When ever we compile an interface ,we get <interface name>.class as an intermediate file ,if no error are present in interface**

- **"implements" is keyword used for getting the features from interface to a class**

  Syntax 1:

  Syntax for re using the features of interface to class:

  Class<class name>implements <intf 1><intf 2>----<intf n>

  {

  Variable declaration;

  Method declaration;

  }

  In the above syntax class name represents name of the class which is inheriting the features from n number of interfaces"implements" is a keyword which is used to inherit the features of interface to a derived class

- **"extends" is keyword used for getting the features from one class to another class or one interface to another interface**

  Syntax2 :

  Syntax inheriting n number of interfaces to another interface

  Interface<intf 1> extends <intf 2> .....<intf n>

  {

  Method definition;

  }

  For example:

  Interface11

  {

```
    Int a=10;
    Void f1();
  }
Interface12 extends interface11
{
     Int b=20;
     Void f2();
}
```

If one interface is taking the features of another interface then that inheritance is known as interface inheritance

Syntax 3:

```
    [abstract] class <derived class name > extends <base class name> implements <intf 1><int f 2>...<intf n>
    {
        Variable declaration ;
        Method definition;
}
```

When ever we used both extends and implements keywords a part of java programme we must always write extends keyword first and later we must use implements keyword

1)one class can extend only one class

2)one class can implement n number of interfaces

3)one interface can extends more than on interfaces

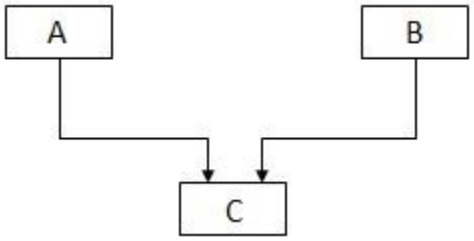
4)interface can not implements are extends a class .since defined things can not be made as undefined things

The implementation of inter face can take various forms

- **A java class can only extends one base class**
- **Multiple inheritance is not allowed in classes level**
- **Interfaces are not classes however the interface can extend more than one base interface**

Multiple inheritance using interfaces:

Multiple inheritance is inheriting properties of two or more parent classes to one child class.As given in the below diagram class A and class B is being inherited by the child class C.



In java multiple inheritance is not supported at classes level but supported in interfaces level.

Interface is a collection of abstract methods(non-defined methods). A class implements an interface, hence inheriting the abstract methods of the interface. An interface may also contain constants, Variables, default methods, static blocks, methods etc.
the syntax of interface is given below :-

// interface

```
interface Parent
{
  public void walk(); // interface method (does not have a body)
  public void run(); // interface method (does not have a body)
}
```

as we have studied above to inherit a class in a class we use *extends* keyword similarily to inherit an interface in a interface we use *extends* keyword but when we are inheriting the interface in the class we use *implements* keyword . hence , to perform multiple interfaces in a class we will be using a keyword *'implements'*. Below is an example of where multiple inheritance is achieved with the help of interfaces.

Multiple Inheritance Using Interface Example Program:

```
interface vehicleone
{
        int  speed=90;
        public void distance();
}

interface vehicletwo
{
        int distance=100;
```

```
        public void speed();
}

class Vehicle  implements vehicleone,vehicletwo
{
        public void distance()
    {
                int  distance=speed*100;
                System.out.println("distance travelled is "+distance);
      }
        public void speed()
    {
                int speed=distance/100;
      }
}
class MultipleInheritanceUsingInterface
{
        public static void main(String args[])
     {
                System.out.println("Vehicle");
                obj.distance();
                obj.speed();
        }
}
```

Sample Output

Output is:

distance travelled is 9000

packages

In java, a package is a container of classes, interfaces, and sub-packages. We may think of it as a folder in a file directory.

We use the packages to avoid naming conflicts and to organize project-related classes, interfaces, and sub-packages into a bundle.

In java, the packages have divided into two types.

- **Built-in Packages**
- **User-defined Packages**

Built-in Packages

The built-in packages are the packages from java API. The Java API is a library of pre-defined classes, interfaces, and sub-packages. The built-in packages were included in the JDK.

There are many built-in packages in java, few of them are as java, lang, io, util, awt, javax, swing, net, sql, etc.

We need to import the built-in packages to use them in our program. To import a package, we use the import statement.

User-defined Packages

The user-defined packages are the packages created by the user. User is free to create their own packages.

Definig a Package in java

We use the package keyword to create or define a package in java programming language.

Syntax

```
package packageName;
```

Let's consider the following code to create a user-defined package myPackage.

Example

```
package myPackage;


public class DefiningPackage {

        public static void main(String[] args) {

                System.out.println("This class belongs to myPackage.");
```

Now, save the above code in a file DefiningPackage.java, and compile it using the following command.

javac -d . DefiningPackage.java

The above command creates a directory with the package name myPackage, and the DefiningPackage.class is saved into it.

Run the program use the following command.

java myPackage.DefiningPackage

Q) explain jar files?

A JAR (Java Archive) is a package file format typically used to aggregate many Java class files and associated metadata and resources (text, images, etc.) into one file to distribute application software or libraries on the Java platform.

In simple words, a JAR file is a file that contains a compressed version of .class files, audio files, image files, or directories. We can imagine a .jar file as a zipped file(.zip) that is created by using WinZip software.

1.1 Create a JAR file

In order to create a .jar file, we can use *jar cf command* in the following ways as discussed below:

Syntax:

jar cf jarfilename inputfiles

Here, cf represents to create the file. For example , assuming our package pack is available in C:\directory , to convert it into a jar file into the pack.jar , we can give the command as:

C:\> jar cf pack.jar pack

C:\> jar cf pack.jar pack

Exception handling

Q) Explain types of errors?

Error refers to an illegal operation performed by the user which results in the abnormal working of the program. Errors are 3 tyeps

a)compile time error

b)runtime error

c)logical error

Java Compile Time Errors

Compile-time errors occur when syntactical problems occur in a java program due to incorrect use of Java syntax. Since all syntax errors are detected by *Java compiler*, therefore, these errors are also known as compile time errors in Java.

These compile time errors are generated may be missing semicolons, missing brackets, misspelled keywords, use of undeclared variables , missing double-quote in Strings, and so on.

Java compiler finds syntax errors in a program,

Example:

```
public class CompileTimeErrorEx
{
public static void main(String[] args)
```

```
{
  System.out.println("a") // Syntax error. Semicolon missing.
 }
}
```

Runtime Errors in Java:

Runtime errors occur when a program is successfully compiled creating the ".class file " but does not run properly. It is detected at run time (i.e. during the execution of the program). ***JVM*** is responsible to detect runtime errors while the program is running.

Java compiler has no technique to detect runtime errors during compilation because a compiler does not have all of the runtime information available to it.

These runtime errors are usually known as exceptions.

The most common runtime errors are as follows:

1. Dividing an integer by zero.
2. Accessing an element that is out of range of the array.
3. Passing an argument that is not in a valid range or valid value for a method.
4. String to use a negative size for an array.
5. Attempting to convert an invalid string into a number.
6. and many more.

Example:

```
public class DivisionByZeroError
{
public static void main(String[] args)
{
 int a = 20, b = 5, c = 5;
 int z = a/(b-c); // Division by zero.

 System.out.println("Result: " +z);
  }
}
```


Logical Errors in Java Program:

Logical errors in Java are the most critical errors in a program and they are difficult to detect. These errors occur when the programmer uses incorrect logic or wrong formula in the coding.

The program will be compiled and executed successfully but does not return the expected output.

Logical errors are not detected either by Java compiler or JVM (Java runtime system). The programmer is entirely responsible for them.

Exception : Runtime errors are called exceptions.

Q)what is exception handling ? how to handling the excptions?

exception handling

Java Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc. Exception is an unwanted or unexpected event, which occurs during the execution of a program, i.e. at run time.

The Exception Handling in Java is one of the powerful *mechanism to handle the runtime errors* so that the normal flow of the application can be maintained.

Using exception handling we can covert system error messages into user friendly messages.

Exception handling can be perform using 3 major keywords

1)try block

2)catch block

3)finally block

Try block in Java:

- try block contains set of statements where an exception can occur.

- A try block is always followed by a catch block or finally block.
- if exception occurs, the rest of the statements in the try block are skipped and the flow immediately jumps to the corresponding catch block.
- We can write "n" number of catch blocks to the single try block
- Try block with in another try block is called as inner (or) nested try block

Note: A try block must be followed by catch blocks or finally block or both.

Syntax of try block with catch block

```
Try
{
   //statements that may cause an exception
}
catch(Exception e)
{
  //statements that will execute when exception occurs
}
```


Syntax of try block with finally block

```
Try
{
   //statements that may cause an exception
}
Finally
```

```
{



  //statements that execute whether the exception occurs or not
}
```

Syntax of try-catch-finally in Java

```
Try
{
   //statements that may cause an exception
}
catch(Exception e)
{
  //statements that will execute if exception occurs
}
Finally
{
  //statements that execute whether the exception occurs or not
}
```


Catch block in Java:

- **A catch block is where you handle the exceptions, this block must immediately placed after a try block.**

- **Catch block is used for displaying user friendly messages.**
- **A single try block can have several catch blocks associated with it. You can catch different exceptions in different catch blocks.**
- **A single try block can have several catch blocks associated with it . but at atime only one catch block will be execute.**
- **When an exception occurs in try block, the corresponding catch block that executes.**
- **For example if an arithmetic exception occurs in try block then the statements enclosed in catch block for arithmetic exception executes.**

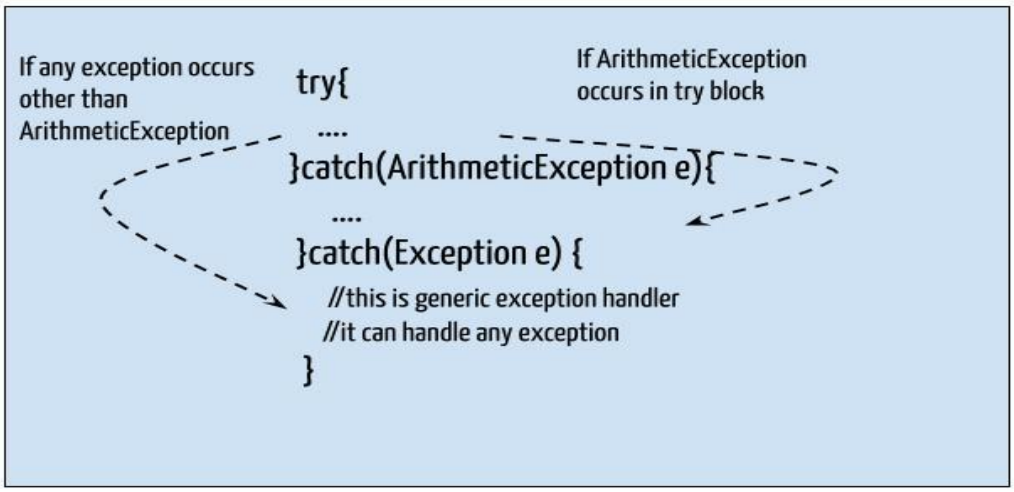
Syntax:


```
catch(exceptionclass1 obj)
{
//error handling code
  }
catch(exceptionclass2 obj)
{
//error handling code
}
```

```
        catch(exceptionclass3 obj)

        {

    //error handling code
     }

        ::::::::::::::;;;;;;:::::::::::::::

        ::::::::::::::::::::::::::::::::::

        catch(exceptionclass-n obj)

        {

        //error handling code

        }
```


Syntax of try catch in java

```
try
{
    //statements that may cause an exception
}
catch (exception(type) e(object))
{
    //error handling code
}
```



Java finally block:

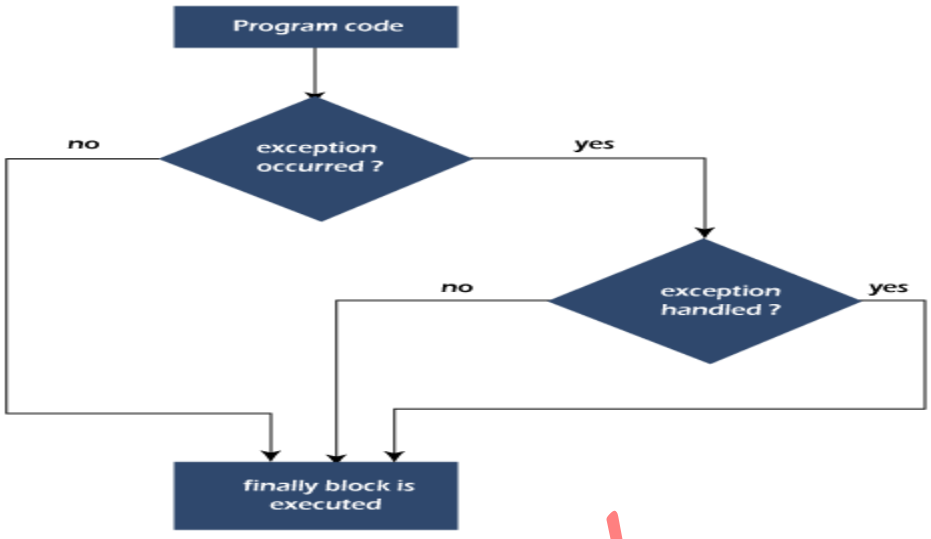
- **Java finally block is a block used to execute important code such as closing the connection, etc.**
- **Java finally block is always executed whether an exception is handled or not. Therefore, it contains all the necessary statements that need to be printed regardless of the exception occurs or not.**
- **The finally block follows the try-catch block.**
- **Writing finally block is optional**

Syntax:

```
finally
{
        Block of code
```

```
}
```

Flowchart of finally block



Throw and throws in Java

The throws keyword:

Whenever an exception occurs in a method you need to handle it by wrapping the code that caused exception within the try-catch block or, you can throw/postpone it using to the calling method using the throws keyword. Then you need to handle the exception at the calling method.

Syntax of Java throws

```
1. return_type method_name() throws exception_class_name
2. {
3. //method code
```

```
4. }
```

The throw keyword:

You can throw a user-defined exception or, a predefined exception explicitly using the *throw* keyword.

There are two types of exceptions user-defined and predefined each exception is represented by a class and which inherits the Throwable class.

To throw an exception explicitly you need to instantiate the class of it and throw its object using the throw keyword.

Syntax:

```
throw new exception_class("error message");
```

example:

Let's see the example of throw IOException.

```
throw new IOException("sorry device error");
```

Example

Throw an exception if age is below 18 (print "Access denied"). If age is 18 or older, print "Access granted":

```
public class Main {
  static void checkAge(int age) {
    if (age < 18) {
      throw new ArithmeticException("Access denied - You must be at
least 18 years old.");
    }
    else {
      System.out.println("Access granted - You are old enough!");
    }
  }

  public static void main(String[] args) {
    checkAge(15); // Set age to 15 (which is below 18...)
  }
}
```


Output:

```
xception in thread "main" java.lang.ArithmeticException: Access
denied - You must be at least 18 years old.
        at Main.checkAge(Main.java:4)
        at Main.main(Main.java:12)
```

Q) what is exception ? explain types of exceptions?

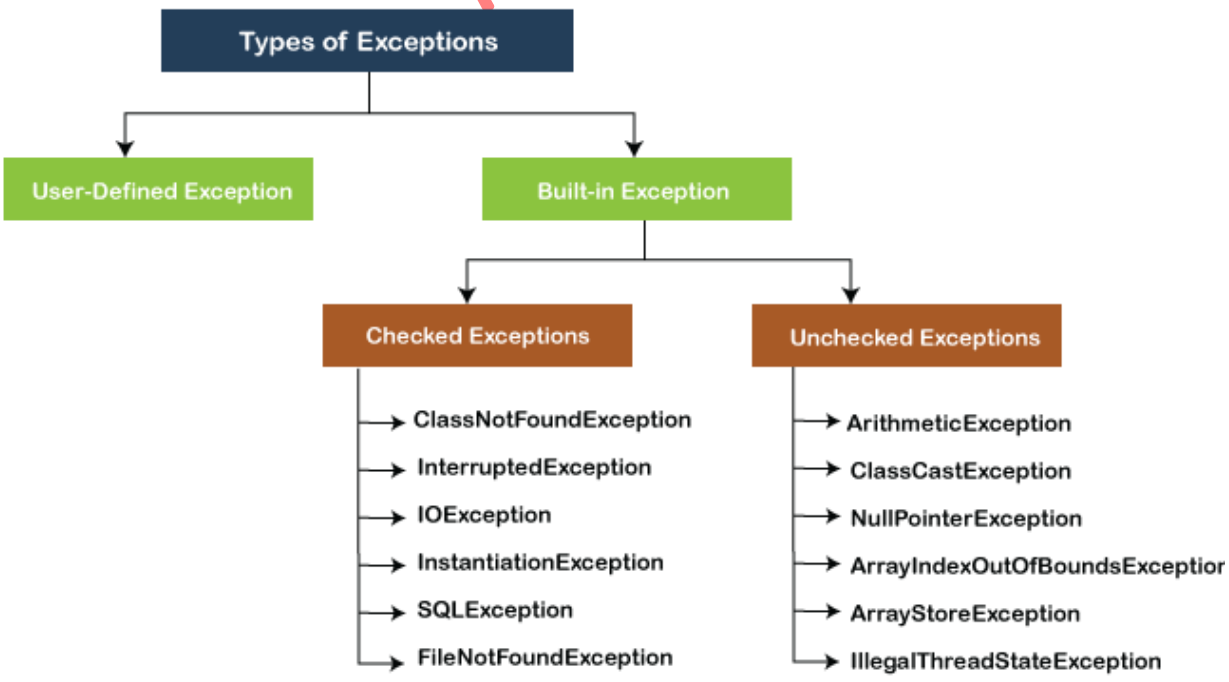
Exception : Runtime errors are called exceptions.

Types of Exception in Java:

In Java, exception is an event that occurs during the execution of a program and disrupts the normal flow of the program's instructions. Bugs or errors that we don't want and restrict our program's normal execution of code are referred to as exceptions. In this section, we will focus on the types of exceptions in Java and the differences between the two.

Exceptions can be categorized into two ways:

1. **Built-in Exceptions**
   - **Checked Exception**
   - **Unchecked Exception**
2. **User-Defined Exceptions**



Built-in Exception(predefined exceptions)

Exceptions that are already available in Java libraries are referred to as built-in exception. It can be categorized into two broad categories, i.e., checked exceptions and unchecked exception.

Checked Exception

Checked exceptions are called compile-time exceptions because these exceptions are checked at compile-time by the compiler. The compiler ensures whether the programmer handles the exception or not. The programmer should have to handle the exception; otherwise, the system has shown a compilation error.

Unchecked Exceptions

The unchecked exceptions are just opposite to the checked exceptions. The compiler will not check these exceptions at compile time. In simple words, if a program throws an unchecked exception, Usually, it occurs when the user provides bad data during the interaction with the program.

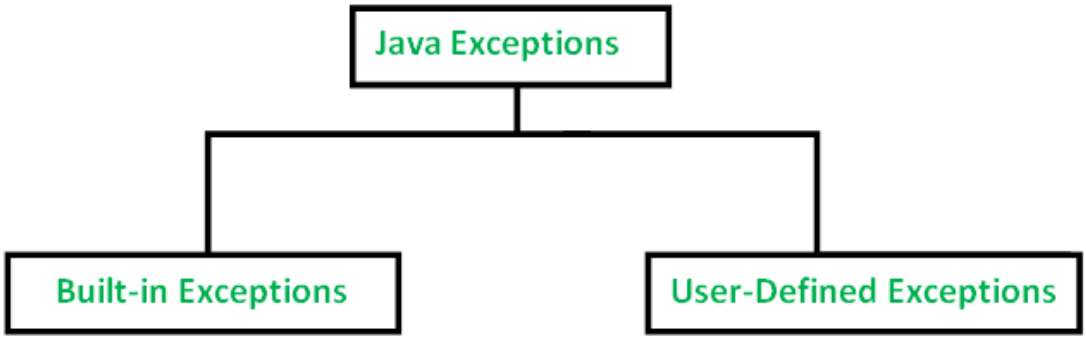
An exception is an unexpected event that occurs during program execution. It affects the flow of the program instructions which can cause the program to terminate abnormally.

An exception can occur for many reasons. Some of them are:

- **Invalid user input**

- **Device failure**
- **Loss of network connection**
- **Physical limitations (out of disk memory)**
- **Code errors**
- **Opening an unavailable file**

java defines several types of exceptions that relate to its various class libraries. Java also allows users to define their own exceptions.



Output

```
ArithmeticException:Division by Zero
```

#2) ArrayIndexOutOfBoundsException:

ArrayIndexOutOfBoundsException is thrown when an array element is accessed using an illegal index.

#3) ClassNotFoundException: If the class definition is not found then the ClassNotFoundException is raised.

#4) FileNotFoundException: FileNotFoundException is given when the file does not exist or does not open.

#5) IOException: IOException is thrown when the input-output operation fails or is interrupted.

#6) InterruptedException: Whenever a thread is doing processing or sleeping or waiting, then it is interrupted by throwing InterruptedException.

#7) NoSuchFieldException: If a class does not contain a specified field or variable, then it throws NoSuchFieldException.

#8) NoSuchMethodException: When the method being accessed is not found, then NoSuchMethodException is raised.

#9) NullPointerException: NullPointerException is raised when a null object is referred. This is the most important and most common exception in Java.

#10) NumberFormatException: This exception is raised when a method could not convert a string into a numeric format.

#11) RuntimeException: Any exception that occurs at runtime is a RuntimeException.

#12) StringIndexOutOfBoundsException: The StringIndexOutOfBoundsException is thrown by String class and indicates that the index is beyond the size of the String object or is negative.

User-Defined Exceptions:

Sometimes, the built-in exceptions in Java are not able to describe a certain situation. In such cases, the user can also create exceptions which are called 'user-defined Exceptions'.

The following steps are followed for the creation of a user-defined Exception.

- **The user should create an exception class as a subclass of the Exception class. Since all the exceptions are subclasses of the Exception class, the user should also make his class a subclass of it. This is done as:**

class MyException extends Exception

- **We can write a default constructor in his own exception class.**

MyException(){}

- **We can also create a parameterized constructor with a string as a parameter.**
  We can use this to store exception details. We can call the

superclass(Exception) constructor from this and send the string there.

```
MyException(String str)
{
  super(str);
}
```

- To raise an exception of a user-defined type, we need to create an object to his exception class and throw it using the throw clause, as:

```
MyException me = new MyException("Exception details");
throw me;
```

unit-4

Multithreading

Multithreading in Java is a process of executing multiple threads simultaneously.

A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

Multitasking

Multitasking is a process of executing multiple tasks simultaneously. We use multitasking to utilize the CPU. Multitasking can be achieved in two ways:

- **Process-based Multitasking (Multiprocessing)**
- **Thread-based Multitasking (Multithreading)**

Process based multitasking: executing several tasks simultaneously where each task is separate independent process such type of multitasking is called process based multitasking.

This type of multitasking is best suitable at "OS level"

Example:

While typing a java program in the editor we can able to listen mp3 audio songs at the same time we can download a file from the net all these tasks are independent of each other and executing simultaneously and hence it is process based multitasking.

Thread based multitasking(multithreading):

Executing several tasks simultaneously where each task is a separate independent part of the same program , is called thread based multitasking . and each independent part is called a "Thread".

This type of multitasking is best suitable for "promatic level".

When compared with "C++" , developing multithreading example is very easy in java because java in buil support for multithreading through a rich API(Thread , Runnable , Threadgroup..)

Q) how to create and running a thread ?

We can create a thread by two ways 1) create a Thread extending "Thread" class

2) using runnable runnable interface

create a Thread by Extending a Thread Class:

step-1:

The way to create a thread is to create a new class that extends Thread class using the following steps. This approach provides more flexibility in handling multiple threads ,

```
Class MyClass extends Thread

{

    ::::::::::::::::::::::::::

    ::::::::::::::::::::::::::

}
```

Step 2:

You will need to override run( ) method available in Thread class. This method provides an entry point for the thread and you will put your complete business logic inside this method. Following is a simple syntax of run() method −

```
public void run( )
```

example:

```
class MyClass extends Thread
 {
      Public void run()
      {
        :::::::::::::::::
```

```
        ::::::::::::::::::
    }
}
```

Step 3:

Once Thread object is created, you can start it by calling start() method, which executes a call to run( ) method. Following is a simple syntax of start() method −

```
void start( );
```

Example

Here is the preceding program rewritten to extend the Thread –

```
class MyClass extends Thread
 {
        public void run()
         {
             for(int i=0;i<10;i++)
            {
                System.out.println("child thread");
            }
```

```
        }
}
Class Hello
{
        public static void main(String args[])
         {
              MyClass t=new MyClass();
             t.start();
           for(int i=0;i<10;i++)
          {
               System.out.println("main thread");
          }
        }
}
```


Output:

Unexpected output:

b)Create a Thread by Implementing a Runnable Interface:

If your class is intended to be executed as a thread then you can achieve this by implementing a Runnable interface. You will need to follow three basic steps −

Step-1

Our class implements to runnable interface

```
Class MyClass implements Runnable
 {
        ::::::::::::::::::::::::::::::::::
 }
```

Step 2

you need to implement a run() method provided by a Runnable interface. This method provides an entry point for the thread and you will put your complete business logic inside this method. Following is a simple syntax of the run() method −

```
public void run( )
```

Step 3

As a third step, you will instantiate a Thread object using the following constructor −

```
Thread(Runnable threadObj);
```

Where, *threadObj* is an instance of a class that implements the Runnable interface and threadName is the name given to the new thread.

Step 4

Once a Thread object is created, you can start it by calling start() method, which executes a call to run( ) method. Following is a simple syntax of start() method −

```
void start();
```

Q) explain Life Cycle of a thread ?

Life Cycle of a Thread

A thread goes through various stages in its life cycle. For example, a thread is born, started, runs, and then dies. The following diagram shows the complete life cycle of a thread.

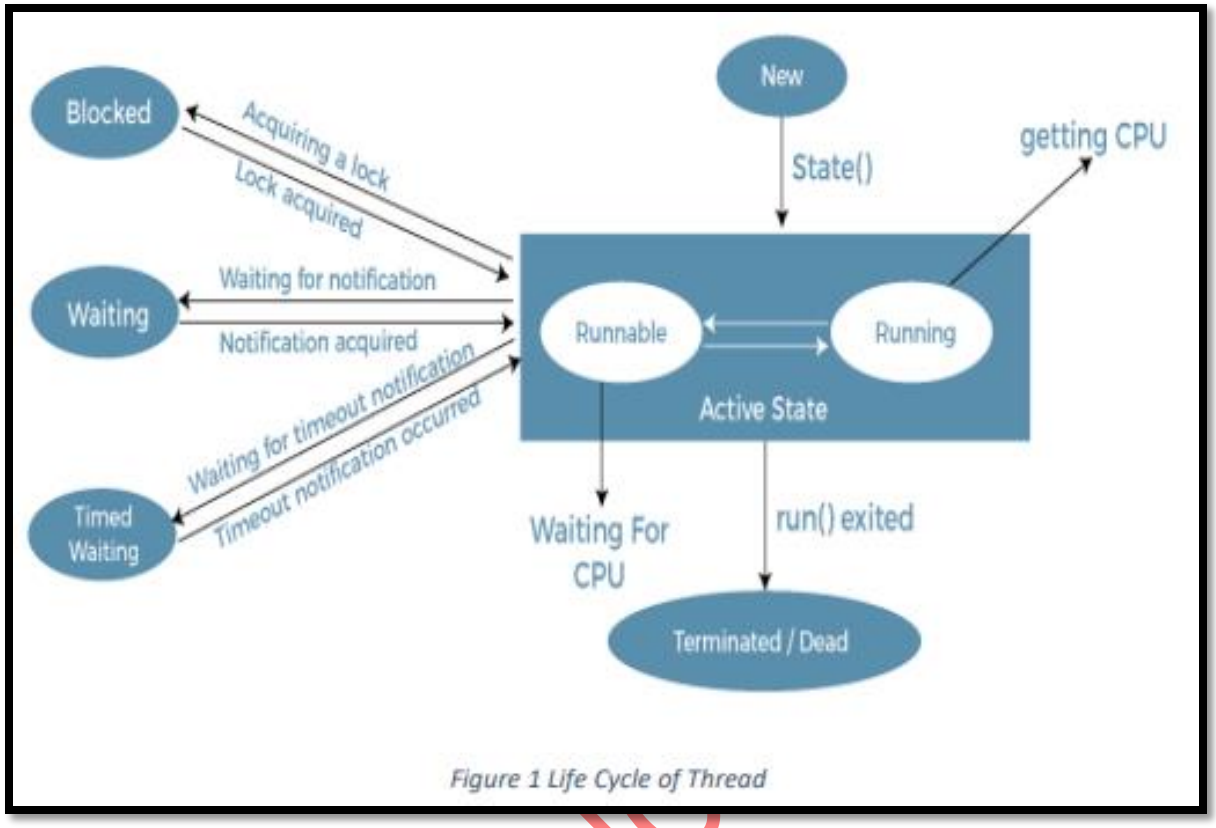


Figure 1 Life Cycle of Thread

Following are the stages of the life cycle −

- **New state −A *NEW Thread* (or a Born *Thread*) is a thread that's been created but not yet started. It remains in this state until we start it using the *start()* method. It is also referred to as a born thread.**
- **Runnable − When we've created a new thread and called the *start()* method on that, it's moved from *NEW* to *RUNNABLE* state. Threads in this state are either running or ready to run, but they're waiting for resource allocation from the system.**
- **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. In this our class run() methoed code is excuted.

- **Waiting state: A runnable thread can enter the timed waiting state for a specified interval of time. A thread in this state transitions back to the runnable state when that time interval expires or when the event it is waiting for occurs . According to JavaDocs, any thread can enter this state by calling any one of the following three methods:**

a. ***object.wait()***
b. ***thread.join()*** **or**
c. ***hread.sleep(long millis)***
d. ***wait(int timeout)*** **or** ***wait(int timeout)***

- **Terminated (Dead) − This is the state of a dead thread. It's in the *TERMINATED* state when it has either finished execution or was terminated abnormally. This state is also called dead state.**

Q) explain methoeds of Thread class?

Methoeds of Thread class:

1. public void start()

starts thread to begin execution, JVM calls run method of this thread.

2. public void run()

run method is used to perform operations by thread.

3. public final void setName(String name)

Changes the name of the thread.

name – new name for this thread.

4. public final String getName()

Returns this thread's name.

5. public final void setPriority(int newPriority)

Changes the priority of the thread.

IllegalArgumentException – If the priority is not in the range MIN_PRIORITY to MAX_PRIORITY.

6. public final int getPriority()

Returns thread's priority.

7. public final boolean isAlive()

returns true if thread is alive.

8. public long getId()

Returns the ID of the thread(A long number generated at the time of thread creation).

9. public static void sleep(long millis)

Causes the currently executing thread to sleep for the specified number of milliseconds.

10. public static void yield()

Causes the currently executing thread object to temporarily pause and allow other threads to execute.

11. public static Thread currentThread()

Returns a reference to the currently running thread.

12. public final void stop()

As the name suggests, this method is used to stop the currently running thread. Remember, once the thread execution is stopped, it cannot be restarted.

13. public final void join()

The current thread invokes this method on a second thread, causing the current thread to block until the second thread terminates.

14. public final void join(long millisec)

The current thread invokes this method on a second thread, causing the current thread to block until the second thread terminates or the specified number of milliseconds passes.

Thread priorities

• Every thread in java has some priority it may be default priority generated by JVM or explicitly provided by the programmer

• The valid range of thread priorities is 1 to 10(but not 0 to 10)where 1 is least priority and 10 is highest priority

• thread class defines the following constants to represent some standard priorities

1)thread.MIN_PRIORITY----------1

2)thread .MAX_PRIORITY---------10

3)thread.NORM_PRIORITY---------5

Program:

Ex1:

```
Class MyThread extends Thread
{
        Public static void main (string args[])
        {
                System.out.println(MIN_PRIORITY);
                System.out.println(NORM_PRIORITY);
                System.out.println(MAX_PRIORITY);
        }
}
```


Output:

1

5

10

• there are no constants like Thread.LOW_PRIORITY , Thread. HIGH_PRIORITY

• thread scheduler uses these priorities while allocating CPU

• the thread which is having highest priority will get chance for first execution

• if 2 threads having the same priority then we can not except exact execution order it depends on thread schedular whose behavior is vendor dependent

• we can get and the set the priority of a thread by using the following methods

1)public final int getPriority ()

2)public final void setPriority(int new priority);//the allowed values are 1 to 10

• the allowed values are 1 to 10 otherwise we will get run time exception saying "IllegalArgumentException".

Default priority:

The default priority only for the main thread is 5 (NORM_PRIORITY).

Example program:

```
Class mythread extends thread
{
    public void run()
    {
```

```
        for( int i=0;i<10;i++)
        {
            System.out.println("child thread");
         }
      }
}
Class threadPriorityDemo
{
    public static void main(string args[])
        {
          Mythread t=new mythread();
         //t.setPriority(10);----------1
          t.start();
          for(int i=0;i<10;i++)
            {
               System.out.println("main thread");
            }
        }
  }
```


• if we are commenting line 1 then both main and child threads will have the same priority and hence we can not expect execution order

• if we are not commenting line one then child thread has the priority 10 and main thread has the priority 5 hence chid thread will get chance for execution and after completing child thread ,main thread will get the chance in this the output is

out put:

child thread

child thread

child thread

child thread

child thread

child thread

child thread

child thread

child thread

child thread

main thread

main thread

main thread

main thread

main thread

main thread

main thread

main thread

main thread

main thread

synchronization

• synchronized is the keyword applicable for methods and blocks but not for classes and variables.

• if a method or block declared as the synchronized then at a time only one thread is allow tin execute that method or block on the given object

• the main advantage of synchronized keyword is we can resolve data inconsistency problems

• but the main disadvantage of synchronized keyword is it increases waiting time of the thread and effects performance of the system

• hence if there is not specific requirement then never recommended to use synchronized keyword

• internally synchronization concept is implemented by using lock concept

• every object in java as unique lock.when ever we are using synchronized keyword

then only lock concept will come in to the picture.

• If a thread wants to execute any synchronized method on the given object 1st it has to get the lock of that object.once a thread got the lock of that object then its allow to execute any synchronized method on that object.if the synchronized method execution completes then automatically thread releases lock.

• **While a thread executing any synchronized method the remaining threads are not allowed execute any synchronized method on that object simultaneously . but remaining threads are allowed to execute any non synchronized method simultaneously.[lock concept is implemented based on object but not based on method]**

Example:

Synchronized method:

```
Class display
{
        Synchronized void wish(string name)
          {
                ------------
                ------------
          }
}
```


Synchronized block:

The synchronized block is used when we want to synchronize only a specific sequence of lines in a method. For example, let's consider a method with 20 lines of code where we want to synchronize only a sequence of 5 lines code, we use the synchronized block.

The folllowing syntax is used to define a synchronized block.

Syntax

```
synchronized(object)
{
   ...
   block code
}
```

example for synchronized block

```
Class display
{
        Void wish(string name)
        {
                ------------
                ------------
                Synchronized(this)
                {
                        ------------
                        ------------
                }
        }
}
```

Program:

```
class Table{
        synchronized void printTable(int n) {
                for(int i = 1; i <= 10; i++)
                        System.out.println(n + " * " + i + " = " + i*n);
        }
```

```
}

class MyThread_1 extends Thread{
	Table table = new Table();
	int number;
	MyThread_1(Table table, int number){
		this.table = table;
		this.number = number;
	}
	public void run() {
		table.printTable(number);
	}
}

class MyThread_2 extends Thread{

	Table table = new Table();
	int number;
	MyThread_2(Table table, int number){
		this.table = table;
		this.number = number;
	}
	public void run() {
		table.printTable(number);
	}
}
public class ThreadSynchronizationExample {

	public static void main(String[] args) {
		Table table = new Table();
		MyThread_1 thread_1 = new MyThread_1(table, 5);
		MyThread_2 thread_2 = new MyThread_2(table, 10);
		thread_1.start();
		thread_2.start();
	}
```

Inter thread communication

(wait(),notify(),notifyAll()):

• Two threads can communicate with each other by using wait(),notify() and notifyAll() methods.

• The thread which is expecting updation it has to call wait() method and the thread which is performing updation it has to call notify() method .after getting notification the waiting thread will get those updations Diagram

• Wait(),notify() and notifyAll() methods are available in object class but in thread class because thread can call these methods on any common object

• To call wait(),notify() and notifyAll() methods are compulsory the current thread should owner of that object that is current thread should has lock of that object that is current thread should in synchronized area .hence we can call wait(),notify() and notifyAll() methods only from synchronized area otherwise we will get run time exception saying IllgalMonitorStateException.

• Once a thread calls wait() method on the given object 1st it releases the lock of that object immediatly and entered into waiting state .

• Once a thred calls notify() or notifyAll() methods ot releases the lock of that object but may not immediately

• Expect these (wait(),notify(),notifyAll()) methods there is no other place (method) Where the lock release will be happen .

Method	Description

Method	Description
void wait( )	It makes the current thread to pause its execution until other thread in the same monitor calls notify( )
void notify( )	It wakes up the thread that called wait( ) on the same object.
void notifyAll()	It wakes up all the threads that called wait( ) on the same object.

Files in java

(only theory part given)

The File is predefined class in Java. In java, the File class has been defined in the java.io package. There are several File Operations like creating a new File, getting information about File, writing into a File, reading from a File and deleting a File.

Q) what is stream? Explain different types of streams?

Stream

A series of data is referred to as a stream. In Java, Stream is classified into two types, i.e., Byte Stream and Character Stream.

Byte Stream:

Byte Stream is mainly involved with byte data. A file handling process with a byte stream is a process in which an input is provided and executed with the byte data.

Character Stream:

Character Stream is mainly involved with character data. A file handling process with a character stream is a process in which an input is provided and executed with the character data.

(also write FileInputStream and FileOutputStream)

Q) EXPLAIN OPERATIONS OF FILES?

File Operations

We can perform the following operation on a file:

- **Create a File**
- **Get File Information**
- **Write to a File**
- **Read from a File**
- **Delete a File**

Create a File:

Create a File operation is performed to create a new file. We use the createNewFile() method of file. The createNewFile() method returns true when it successfully creates a new file and returns false when the file already exists.

Get File Information

The operation is performed to get the file information. We use several methods to get the information about the file like name, absolute path, is readable, is writable and length.

1. **We get the name of the file using the getName()**

2. We get the absolute path of the file using the getAbsolutePath() method of the file.
3. We check whether we can write data into a file or not using the canWrite()
4. We check whether we can read the data of the file or not using the canRead()
5. We get the length of the file by using the length()

Write to a File

The next operation which we can perform on a file is "writing into a file". In order to write data into a file, we will use the FileWriter class and its write() method together. We need to close the stream using the close() method to retrieve the allocated resources.

Read from a File

The next operation which we can perform on a file is "read from a file". In order to read data into a file, we will use the FileRead class and its read() method together. Here, we need to close the stream using the close() method..

Delete a File

The next operation which we can perform on a file is "deleting a file". In order to delete a file, we will use the delete() method of the file. We don't need to close the stream using the close() method because for

deleting a file, we neither use the FileWriter class nor the Scanner class.

Q) explain FileInputStream and FieOutputStream?

(or)

Explain readiing and writing the data into files?

File Handling using Byte Stream:

Java provides I/O Streams to read and write data where, a Stream represents an input source or an output destination which could be a file, i/o devise, other program etc.

There are two types of streams available −

- **InputStream − This is used to read (sequential) data from a source.**
- **OutputStream − This is used to write data to a destination.**

FileInputStream

This class reads the data from a specific file (byte by byte). It is usually used to read the contents of a file with raw bytes, such as images.

To read the contents of a file using this class −

- **First of all, you need to creating object of this class by passing a String variable or a File object, representing the path of the file to be read.**

FileInputStream inputStream = new FileInputStream(file);

- **Then read the contents of the specified file using either of the variants of read() method −**
  - **int read() − This simply reads data from the current InputStream and returns the read data byte by byte (in integer format).**

    This method returns -1 if the end of the file

FileOutputStream:

This writes data into a specific file or, file descriptor (byte by byte). It is usually used to write the contents of a file with raw bytes, such as images.

To write the contents of a file using this class −

- **First of all, you need to increating object of this class by passing a String variable or a File object, representing the path of the file to be read.**

FileOutputStream outputStream = new FileOutputStream (file);

You can also instantiate a FileOutputStream class by passing a FileDescriptor object.

```
FileDescriptor descriptor = new FileDescriptor();
FileOutputStream outputStream = new FileOutputStream(descriptor);
```

- **Then write the data to a specified file using either of the variants of write() method −**
  - **int write(int b) − This method accepts a single byte and writes it to the current OutputStream.**

File Handling using Character Stream:

In java, we can use a character stream to handle files. The character stream has the following built-in classes to perform various operations on a file.

- **FileReader - It is a built-in class in java that allows reading data from a file. This class has implemented based on the character stream. The FileReader class provides a method read() to read data from a file character by character.**

FileWriter - It is a built-in class in java that allows writing data to a file. This class has implemented based on the character stream. The FileWriter class provides a method write() to write data to a file character by character

Unit-5

APPLETS

In Java, an applet is a special type of program embedded in the web page to generate dynamic content. Applet is a class in Java.

Q) explain steps for create and execute applet program ?

CREATE AND EXECUTE APPLET PROGRAM

An applet is a Java program that runs in a Java-compatible browser such as Internetexplorer. This feature allows users to display graphics and to run programs over the Internet .

Step 1: Import applet package and awt package: To create an applet, our program must import the Applet class. This class is found in the *java.applet* package. We also need to import the *java.awt* package. "awt" stands for *"Abstract Window Toolkit"*. The java.awt package includes classes like *Graphics.*

Step 2: Extend the Applet class: Then, a class must be defined that inherits from the class 'Applet' and The inherited class must be declared public.

Step 3: Override the paint method to draw text or graphics: The paint method needs the Graphics object as its parameter.

public void paint(Graphics g) { ... }

The Graphics class object holds information about painting. We can invoke methods like drawLine(), drawCircle() and etc using this object.

Syntax: The program looks something like this.

```
import java.applet.*;
import java.awt.*;
public class MyApplet extends Applet
 {
    public void paint(Graphics g)
    {
        g.drawString("Welcome to Apoorva",50,50);
    }
}
```


Step 4: Compiling the Program: After writing the program, we compile it using the command "javac MyApplet.java". This command will compile our code so that we now have MyApplet.class file.

Step 5: Adding applet to HTML document: To run the applet we need to create the HTML document. The BODY section of the HTML document allows APPLET tag. The HTML file looks something like this:

```
<HTML>
<BODY>
 <APPLET codebase="MyAppPath" code="MyApp.class" width=200
height=200> </APPLET>
</BODY>
</HTML>
```

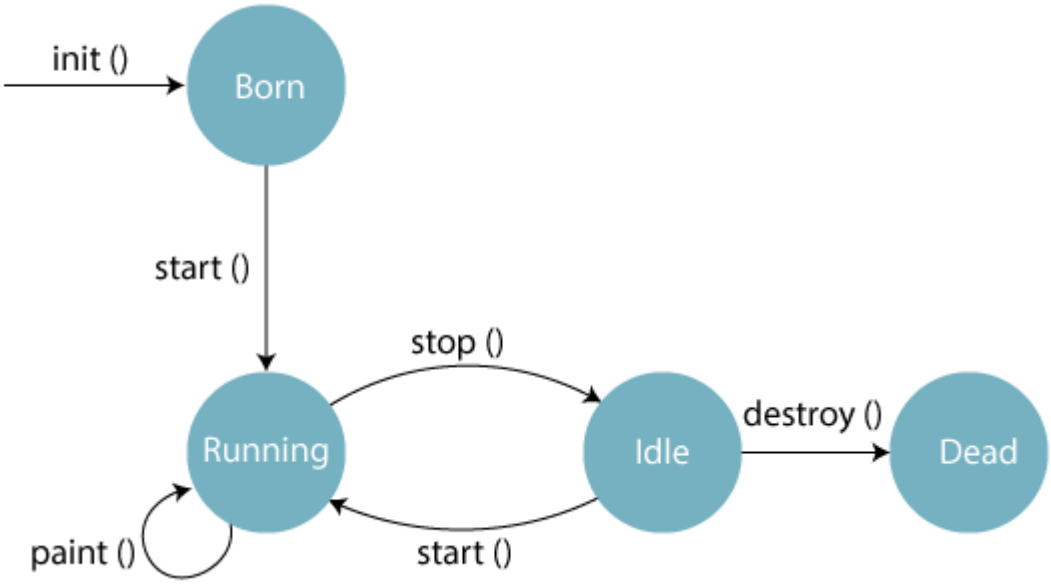
Step 6: Running an applet: The applet can be run in two ways

Using appletviewer: To run the appletviewer, type *appletviewer filename.html*

Using web browser: Open the web browser, type in the full address of html file.

Q) explain applet life cycle ?

Methods of Applet Life Cycle



The applet life cycle can be defined as the process of how the object is created, started, stopped, and destroyed during the entire execution of its application. It basically has five core methods namely init(), start(), stop(), paint() and destroy().These methods are invoked by the browser to execute.

here are five methods of an applet life cycle, and they are:

- **init(): The init() method is the first method to run that initializes the applet. It can be invoked only once at the time of initialization . The web browser creates the initialized objects, i.e., the web browser (after checking the security settings) runs the init() method within the applet.**
- **start(): The start() method contains the actual code of the applet and starts the applet. It is invoked immediately after the init() method is invoked. Every time the browser is loaded or refreshed, the start() method is invoked. It is also invoked whenever the applet is maximized, restored, or moving from one tab to another in the browser.**
- **stop(): The stop() method stops the execution of the applet . The stop () method is invoked whenever the applet is stopped, minimized , the stop() method is invoked. When we go back to that page, the start() method is invoked again.**
- **destroy(): The destroy() method destroys the applet after its work is done. It is invoked when the applet window is closed . It**

removes the applet object from memory and is executed only once. We cannot start the applet once it is destroyed.

- paint(): The paint() method belongs to the Graphics class in Java. It is used to draw shapes like circle, square, trapezium, etc., in the applet. It is executed after the start() method and when the browser or applet windows are resized.

Sequence of method execution when an applet is executed:

1. init()
2. start()
3. paint()

Sequence of method execution when an applet is executed:

1. stop()
2. destroy()

Syntax of entire Applet Life Cycle in Java

```
class TestAppletLifeCycle extends Applet {

public void init() {

// initialized objects

}

public void start() {
```

```
// code to start the applet

}

public void paint(Graphics graphics) {

// draw the shapes

}

public void stop() {

// code to stop the applet

}

public void destroy() {

// code to destroy the applet

}

}
```


Q) EXPLAIN parameters of applet ?

<APPLET> (parameters of applet tag):

The HTML <applet> tag specifies an applet. It is used for embedding a Java applet within an HTML document. It is not supported in HTML5.

```
<html>
```

```
   <head>
      <title>HTML applet Tag</title>
   </head>

   <body>
      <applet code = "newClass.class" width = "300" height =
"200"></applet>
   </body>

</html>
```

Here is the *newClass.java* file –

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

public class newClass extends Applet
{
   public void paint (Graphics gh)
   {
      g.drawString("Tutorialspoint.com", 300, 150);
   }
}
```

This will produce the following result –

Specific Attributes

The HTML <> tag also supports following additional attributes −

Attribute	Value	Description
Align	URL	*Deprecated* − Defines the text alignment around the applet
Alt	URL	Alternate text to be displayed in case browser does not support applet
Height	Pixels	Height to display the applet
Hspace	Pixels	Deprecated − Defines the left and right spacing around the applet
Name	Name	Defines a unique name for the applet
Width	Pixels	Width to display the applet.

Q) exaplin an applet with swing component ?

Java Swing Example

AWT and Swing are used to develop window-based applications in Java. Awt is an abstract window toolkit that provides various component classes like Label, Button, TextField, etc., to show window components on the screen. All these classes are part of the Java.awt package.

On the other hand, Swing is the part of JFC (Java Foundation Classes) built on the top of AWT and written entirely in Java. The javax.swing API provides all the component classes like JButton, JTextField, JCheckbox, JMenu, etc.

In the following example, we have created a User form by using the swing component classes provided by the javax.swing package. Consider the example.

```
import javax.swing.*;
  public class SwingApp
 {
    SwingApp(){
    JFrame f = new JFrame();

     JLabel firstName = new JLabel("First Name");
```

```
    firstName.setBounds(20, 50, 80, 20);

     JLabel lastName = new JLabel("Last Name");
      lastName.setBounds(20, 80, 80, 20);

     JLabel dob = new JLabel("Date of Birth");
     dob.setBounds(20, 110, 80, 20);

JTextField  firstNameTF = new JTextField();
firstNameTF.setBounds(120, 50, 100, 20);

JTextField  lastNameTF = new JTextField();
lastNameTF.setBounds(120, 80, 100, 20);

JTextField  dobTF = new JTextField();
dobTF.setBounds(120, 110, 100, 20);

JButton sbmt = new JButton("Submit");
sbmt.setBounds(20, 160, 100, 30);

JButton reset = new JButton("Reset");
reset.setBounds(120,160,100,30);

f.add(firstName);
f.add(lastName);
```

```
f.add(dob);
f.add(firstNameTF);
f.add(lastNameTF);
f.add(dobTF);
f.add(sbmt);
f.add(reset);
f.setSize(300,300);
f.setLayout(null);
f.setVisible(true);
}

public static void main(String[] args) {

SwingApp s = new SwingApp();
}

}
```


Output:

First name: 

Last name: 

Date of birth : 


Q) a simple game with an applet ?

Animation in Applet

Applet is mostly used in games and animation. For this purpose image is required to be moved.

Example of animation in applet:

```
import java.awt.*;
import java.applet.*;
public class AnimationExample extends Applet {

  Image picture;

  public void init() {
    picture =getImage(getDocumentBase(),"bike_1.gif");
  }

  public void paint(Graphics g) {
    for(int i=0;i<500;i++){
      g.drawImage(picture, i,30, this);

      try{Thread.sleep(100);}catch(Exception e){}
    }
  }
```

```
}
```

in the above example, drawImage() method of Graphics class is used to display the image. The 4th argument of drawImage() method of is ImageObserver object. The Component class implements ImageObserver interface. So current class object would also be treated as ImageObserver because Applet class indirectly extends the Component class.

myapplet.html

```
1. <html>
2. <body>
3. <applet code="DisplayImage.class" width="300" height="300">
4. </applet>
5. </body>
6. </html>
```

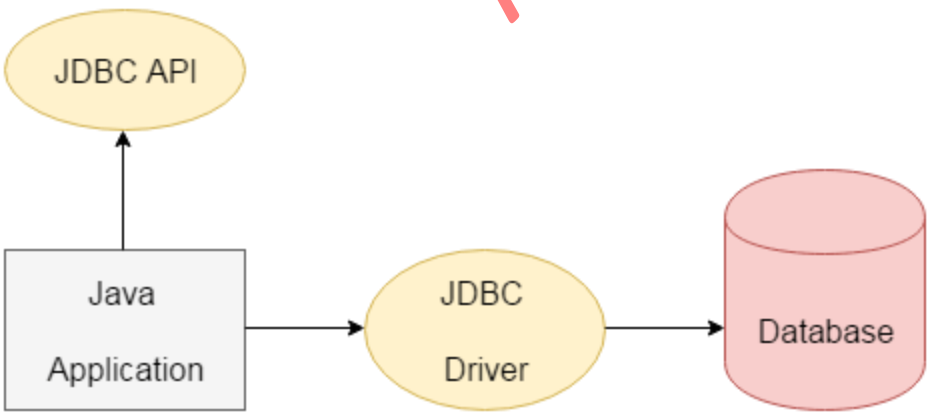

JDBC

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

- **JDBC-ODBC Bridge Driver,**
- **Native Driver,**
- **Network Protocol Driver, and**
- **Thin Driver**

We have discussed the above four drivers in the next chapter.

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database. It is like Open Database Connectivity (ODBC) provided by Microsoft.



Q)explain JDBC Driver?

JDBC Driver is a software component that enables java application to

interact with the database. There are 4 types of JDBC drivers:

1. **JDBC-ODBC bridge driver**
2. **Native-API driver (partially java driver)**
3. **Network Protocol driver (fully java driver)**
4. **Thin driver (fully java driver)**

Q) Explain about java database connectivity ?

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

java Database Connectivity with 5 Steps:

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

- **Register the Driver class**
- **Create connection**
- **Create statement**
- **Execute queries**
- **Close connection**

Register the driver class

The forName() method of Class class is used to register the driver

class. This method is used to dynamically load the driver class.

Syntax of forName() method

1. **public static void forName(String className)throws ClassNotFoundException**

Example to register the OracleDriver class

Here, Java program is loading oracle driver to esteblish database connection.

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

2) Create the connection object

The getConnection() method of DriverManager class is used to establish connection with the database.

Syntax of getConnection() method

1) public static Connection getConnection(String url)throws SQLException

2) public static Connection getConnection(String url,String name,String password) throws SQLException

example:

```
Connection con=DriverManager.getConnection(
```

```
"jdbc:oracle:thin:@localhost:1521:xe","system","password");
```

3) Create the Statement object

The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.

Syntax of createStatement() method

```
1. public Statement createStatement()throws SQLException
```

Example to create the statement object

```
1. Statement stmt=con.createStatement();
```


4) Execute the query

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

Syntax of executeQuery() method

```
public ResultSet executeQuery(String sql)throws SQLException
```

Example to execute query

```
ResultSet rs=stmt.executeQuery("select * from emp");
```

```
while(rs.next()){

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

}
```

5) Close the connection object

By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

Syntax of close() method

```
public void close()throws SQLException
```


Q) CONNECTING Java Database Connectivity with Oracle?

Java Database Connectivity with Oracle

To connect java application with the oracle database, we need to follow 5 following steps. In this example, we are using Oracle 10g as the database. So we need to know following information for the oracle database:

1. **Driver class: The driver class for the oracle database is oracle.jdbc.driver.OracleDriver.**
2. **Connection URL: The connection URL for the oracle10G database**

is jdbc:oracle:thin:@localhost:1521:xe where jdbc is the API, oracle is the database, thin is the driver, localhost is the server name on which oracle is running, we may also use IP address, 1521 is the port number and XE is the Oracle service name. You may get all these information from the tnsnames.ora file.

3. Username: The default username for the oracle database is system.
4. Password: It is the password given by the user at the time of installing the oracle database

Create a Table

Before establishing connection, let's first create a table in oracle database. Following is the SQL query to create a table.

1. create table emp(id number(10),name varchar2(40),age number(3));

Example to Connect Java Application with Oracle database

In this example, we are connecting to an Oracle database and getting data from emp table. Here, system and oracle are the username and password of the Oracle database.

```
import java.sql.*;

class OracleCon{
```

```
public static void main(String args[]){

try{

//step1 load the driver class

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



//step2 create  the connection object

Connection con=DriverManager.getConnection(

"jdbc:oracle:thin:@localhost:1521:xe","system","oracle");



//step3 create the statement object

Statement stmt=con.createStatement();



//step4 execute query

ResultSet rs=stmt.executeQuery("select * from emp");

while(rs.next())

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

```
//step5 close the connection object

con.close();


}catch(Exception e){ System.out.println(e);}


}

}
```


Q) CONNECTING Java Database Connectivity with MySQL?

Java Database Connectivity with MySQL

To connect Java application with the MySQL database, we need to follow 5 following steps.

In this example we are using MySql as the database. So we need to know following informations for the mysql database:

1. **Driver class: The driver class for the mysql database is com.mysql.jdbc.Driver.**

2. **Connection URL: The connection URL for the mysql database is jdbc:mysql://localhost:3306/sonoo where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we may also use IP address, 3306 is the port number and sonoo is the database name. We may use any database, in such case, we need to replace the sonoo with our database name.**
3. **Username: The default username for the mysql database is root.**
4. **Password: It is the password given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.**

Let's first create a table in the mysql database, but before creating table, we need to create database first.

1. **create database sonoo;**
2. **use sonoo;**
3. **create table emp(id int(10),name varchar(40),age int(3));**

Example to Connect Java Application with mysql database

In this example, sonoo is the database name, root is the username and password both.

```
import java.sql.*;

class MysqlCon{
```

```
public static void main(String args[]){

try{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection(

"jdbc:mysql://localhost:3306/sonoo","root","root");

//here sonoo is database name, root is username and password

Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("select * from emp");

while(rs.next())

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

con.close();

}catch(Exception e){ System.out.println(e);}

}

}
```