



V2V EDTECH LLP

Online Coaching at an Affordable Price.

OUR SERVICES:

- Diploma in All Branches, All Subjects
- Degree in All Branches, All Subjects
- BSCIT / CS
- Professional Courses



+91 93260 50669



v2vedtech.com



V2V EdTech LLP



v2vedtech

1. Define Constructor. Explain its types with example

Ans:

1

YOUTUBE : [SUBSCRIBE NOW](#) INSTA : [FOLLOW NOW](#)

Download V2V APP on Playstore for more [FREE STUDY MATERIAL](#)

Contact No : 9326050669 / 9326881428

- Constructor in JAVA is a special type of method that is used to initialize the object.
- JAVA constructor is invoked at the time of object creation.
- It constructs the values i.e. provides data for the object that is why it is known as constructor.
- A constructor has same name as the class in which it resides.
- Constructor in JAVA cannot be abstract, static, final or synchronized.
- These modifiers are not allowed for constructor.

There are two types of constructors

1. Default constructor (no-arg constructor)
2. Parameterized constructor

1. Default Constructor

- A constructor that have no parameter is known as default constructor.

Syntax of default constructor

```
<class_name> ( )  
{  
}
```

Example of default constructor.

```
class Bike1  
{  
    Bike1()  
    {  
        System.out.println("Bike is created");  
    }  
    public static void main(String args[ ] )  
    {  
        Bike1 b=new Bike1();  
    }  
}
```

This will produce the following result Bike is created

Note – If there is no constructor in a class, compiler automatically creates a default constructor.

2. Parameterized Constructor

- A constructor that have parameters is known as parameterized constructor.
- Parameterized constructor is used to provide different values to the distinct objects.

Example of parameterized constructor.

```
class Student  
{  
    int id;  
    String name;  
    Student(int i, String n)  
    {
```

```
id = i;
name = n;
}
void display( )
{
System.out.println(id+" "+name);
}
public static void main(String args[ ])
{
Student s1 = new Student4(111,"Ram");
Student s2 = new Student4(222,"Shyam");
s1.display();
s2.display();
}
}
```

This will produce the following result

```
111 Ram
222 Shyam
```

2. Explain access specifiers in Java with example

Ans:

The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it. There are four types of Java access modifiers:

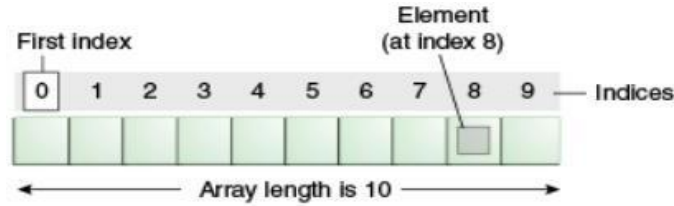
1. Private: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. Default: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
3. Protected: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
4. Public: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

3. Define array. Explain its types with examples.

Ans:

- An array is a collection of similar type of elements which has contiguous memory location.
- Java array is an object which contains elements of a similar data type.
- We can store only a fixed set of elements in a Java array.
- Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.

- Java provides the feature of anonymous arrays which is not available in C/C++.



Type of Array There are two types of array.

- One/ Single Dimensional Array
- Two/Multidimensional Array

1. One/ Single Dimensional Array :

The One dimensional array can be represented as

Array a[5]

10	20	70	40	50
0	1	2	3	4

Example

```
class Testarray
{
public static void main(String args[])
{
int a[] = new int[5]; //declaration and instantiation
a[0]=10; //initialization
a[1]=20;
a[2]=30;
a[3]=40;
a[4]=50;
//traversing array
for(int i=0; i<a.length; i++)
System.out.println(a[i]);
}
}
```

OUTPUT

10
20
70
40
50

2. Two dimensional Array :

In such case, data is stored in row and column based index (also known as matrix form).

Example to instantiate Multidimensional Array in Java

```
int[ ][ ] arr =new int[3][3]; //3 row and 3 column
```

Example to initialize Multidimensional Array in Java

```
arr[0][0]=1;
```

```
arr[0][1]=2;
```

```
arr[0][2]=3;
```

```
arr[1][0]=4;
```

```
arr[1][1]=5;
```

```
arr[1][2]=6;
```

```
arr[2][0]=7;
```

```
arr[2][1]=8;
```

```
arr[2][2]=9;
```

Example of Multidimensional Java Array

```
class Testarray2
```

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

```
{
```

```
//declaring and initializing 2D array
```

```
int arr[][]={{1,2,3},{2,4,5},{4,4,5}}; //printing 2D array
```

```
for(int i=0;i<3;i++)
```

```
{
```

```
for(int j=0;j<3;j++)
```

```
{
```

```
System.out.print(arr[i][j] + " ");
```

```
}
```

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

```
}
```

```
}
```

```
}
```

Output

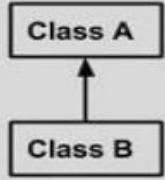
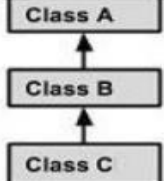
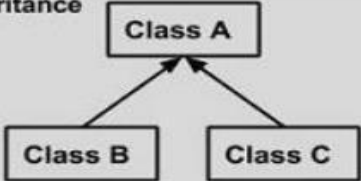
```
1 2 3
```

```
2 4 5
```

```
4 4 5
```

4. Explain types of inheritance which is supported by java with example

Ans:

<p>Single Inheritance</p>  <pre> graph BT B[Class B] --> A[Class A] </pre>	<pre> public class A { } public class B extends A { } </pre>
<p>Multi Level Inheritance</p>  <pre> graph BT C[Class C] --> B[Class B] B --> A[Class A] </pre>	<pre> public class A {} public class B extends A {.....} public class C extends B {.....} </pre>
<p>Hierarchical Inheritance</p>  <pre> graph BT B[Class B] --> A[Class A] C[Class C] --> A </pre>	<pre> public class A {} public class B extends A {.....} public class C extends A {.....} </pre>

Program for implanting Single Inheritance

```

class Teacher
{
void Display()
{
System.out.println("This is teacher ");
}
}
class student extends Teacher
{
void show()
{
System.out.println("This is Student");
}
}
class TestInheritance
{
public static void main(String args[])
{
Student s=new student();
s.show();
d.display();
}
}
    
```

Output

This is student

This is teacher

5. Write a java program to implement multilevel inheritance with 4 levels of hierarchy.

Ans:

```
class emp
{
int empid;
String ename;
emp(int id, String nm)
{
empid=id;
ename=nm;
}
}
class work_profile extends emp
{
String dept;
String job;
work_profile(int id, String nm, String dpt, String j1)
{ super(id,nm);
dept=dpt; job=j1;
}
}
class salary_details extends work_profile
{
int basic_
salary; salary_details(int id, String nm, String dpt, String j1,int bs)
{
super(id,nm,dpt,j1);
basic_salary=bs;
}
double calc()
{
double gs;
gs=basic_salary+(basic_salary*0.4)+(basic_salary*0.1);
return(gs);
}
}
class salary_calc extends salary_details
```

```
{
salary_calc(int id, String nm, String dpt, String j1,int bs)
{
super(id,nm,dpt,j1,bs);
}
public static void main(String args[])
{
salary_calc e1=new salary_calc(101,"abc","Sales","clerk",5000);
double gross_salary=e1.calc();
System.out.println("Empid :"+e1.empid);
System.out.println("Emp name :"+e1.ename);
System.out.println("Department :"+e1.dept);
System.out.println("Job :"+e1.job);
System.out.println("BASIC Salary :"+e1.basic_salary);
System.out.println("Gross salary :"+gross_salary);
}
}
```

6. Explain garbage collection and State use of finalize() method with its syntax.

Ans:

Garbage collection

- In JAVA destruction of object from memory is done automatically by the JVM.
- When there is no reference to an object, then that object is assumed to be no longer needed and the memory occupied by the object are released.
- This technique is called Garbage Collection.
- This is accomplished by the JVM.
- Unlike C++ there is no explicit need to destroy object.

finalize() method

- Sometime an object will need to perform some specific task before it is destroyed such as closing an open connection or releasing any resources held.
- To handle such situation finalize() method is used.
- finalize()method is called by garbage collection thread before collecting object.
- Its the last chance for any object to perform cleanup utility.

Signature of finalize() method

protected void finalize()

```
{
//finalize-code
}
```

Some Important Points to Remember

1. finalize() method is defined in JAVA.lang.Object class, therefore it is available to all the classes.
2. finalize() method is declare as protected inside Object class.
3. finalize() method gets called only once by GC(Garbage Collector) threads.

6. Explain package & Give syntax to create a package and accessing package in java

Ans:

Packages:

Java provides a mechanism for partitioning the class namespace into more manageable parts called package (i.e package are container for a classes). The package is both naming and visibility controlled mechanism. We can define classes inside a package that are not accessible by code outside that package. We can also define class members that are only exposed to members of the same package.

Creating Packages:- (Defining Packages)

Creation of packages includes following steps:

- 1) Declare a package at the beginning of the file using the following form.

```
package package_name
```

e.g.

```
package pkg; - name of package
```

package is the java keyword with the name of package. This must be the first statement in java source file.

- 2) Define a class which is to be put in the package and declare it public like following way.

```
Package first_package;
```

```
Public class first_class
```

```
{
```

```
Body of class;
```

```
}
```

In above example, "first_package" is the package name. The class "first_class" is now considered as a part of this package.

- 3) Create a sub-directory under the directory where the main source files are stored.

- 4) Store the listing of, as classname.java file is the sub-directory created. e.g. from the above package we can write "first_class.java"

- 5) Compile the source file. This creates .class file is the sub-directory. The .class file must be located in the package and this directory should be a sub-directory where classes that will import the package are located.

Accessing a package:-

To access package In a Java source file, import statements occur immediately. Following the package statement (if it exists) and before any class definitions.

Syntax:

```
import pkg1[.pkg2].(classname | *);
```

Here, “pkg1” is the name of the top level package. “pkg2” is the name of package is inside the package1 and so on.

“classname”-is explicitly specified statement ends with semicolon.

- Second way to access the package
import packagename.*;

7. Explain the syntax of try-catch-finally blocks with examples.

Ans:

Syntax of try-catch-finally blocks

```
try {  
.....  
}  
catch(.....) {  
.....  
}  
catch (.....) {  
.....  
}  
finally {  
.....  
}
```

Example:

```
class finallyEx  
{ public static void main(String[] args)  
{  
try  
{  
int[] myNumbers = {1, 2, 3};  
System.out.println(myNumbers[10]);  
}  
catch (Exception e)  
{  
System.out.println("Something went wrong.");  
}  
finally {  
System.out.println("The 'try catch' is finished.");  
}  
}  
}
```

OUTPUT:

Something went wrong. The 'try catch' is finished.



8. Define thread. Explain 2 ways to create thread.

Ans:

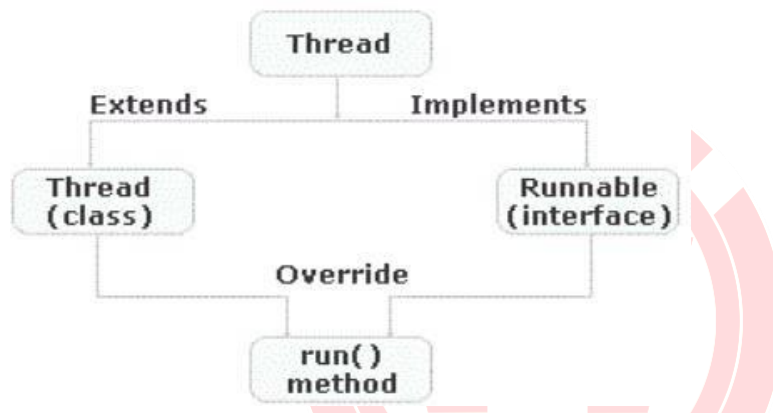
Definition: "A thread is a lightweight subprocess, the smallest unit of processing."

- It is a separate path of execution.
- Threads are independent.
- If exception occurs in one thread, it doesn't affect other threads.
- It uses a shared memory area.
- We can increase the speed of application using thread.

Creating thread:

Thread can be implemented through any one of two ways:

1. Extending the Java.lang.Thread class
2. Implementing Java.lang.Runnable interface



Extending Thread class

1. Extending the java.lang.Thread class:
 - a. Extend Java.lang.Thread class
 - b. Override run() method in subclass from Thread class
 - c. Create instance of this subclass. This subclass may call a Thread class constructor by subclass constructor.
 - d. Invoke start() method on instance of class to make thread eligible for running.

Example:

```
class Multi extends Thread
{
public void run()
{
System.out.println("thread is running...");
}
public static void main(String args[])
{
Multi t1=new Multi(); t1.start();
}
}
```

Output:
thread is running...

Implementing Runnable interface

2. Implementing Java.lang.Runnable interface :

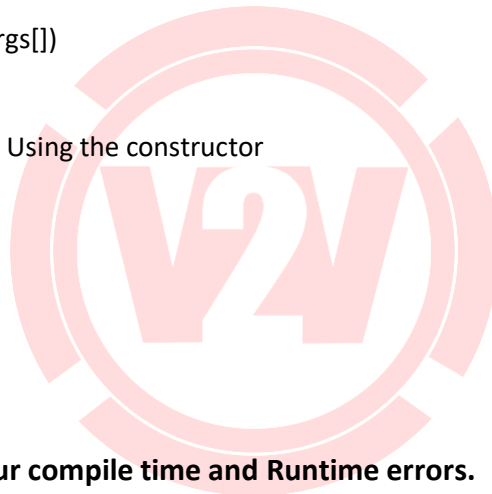
- a. An object of this class is Runnable object.
- b. Create an object of Thread class by passing a Runnable object as argument.
- c. Invoke start() method on the instance of Thread class.

Example:

class Multi3 implements Runnable

```
{  
public void run()  
{  
System.out.println("thread is running...");  
}  
public static void main(String args[])  
{  
Multi3 m1=new Multi3();  
Thread t1 =new Thread(m1); // Using the constructor  
Thread(Runnable r)  
t1.start();  
}  
}
```

Output:
thread is running...



9. Define error Enlist any four compile time and Runtime errors.

Ans:

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.

- Error may produce
- An incorrect output
- may terminate the execution of program abruptly
- may cause the system to crash
- It is important to detect and manage properly all the possible error conditions in program.

Types of Errors

1. Compile time Errors: Detected by javac at the compile time
2. Run time Errors: Detected by java at run time

1. Compile Time Errors(Syntactical errors) :

- Errors which are detected by javac at the compilation time of program are known as compile time errors.
- Most of compile time errors are due to typing mistakes, which are detected and displayed by javac.
- Whenever compiler displays an error, it will not create the .class file.
- Typographical errors are hard to find.
- The most common problems:
 - Missing semicolon
 - Missing or mismatch of brackets in classes and methods
 - Misspelling of identifiers and keywords
 - Missing double quotes in strings
 - Use of undeclared variables
 - Incompatible types in assignments/ Initialization
 - Bad references to objects
 - Use of = in place of == operator etc.
- Other errors are related to directory path like command not found

2. Run time Error(Logical Error)

- There is a time when a program may compile successfully and creates a .class file but may not run properly.
- It may produce wrong results due to wrong logic or may terminate due to errors like stack overflow, such logical errors are known as run time errors.
- Java typically generates an error message and aborts the program.
- It is detected by java (interpreter)
- Run time error causes termination of execution of the program.
- The most common run time errors are:
 - ♣ Dividing an integer by zero.
 - ♣ Accessing element that is out of the bounds of an array.
 - ♣ Trying to store a value into an array of an incompatible class or type.
 - ♣ Passing a parameter that is not in a valid range or value for a method.
 - ♣ Attempting to use a negative size of an array
 - ♣ Converting invalid string to a number.

10. Write a program to check whether the given number is prime or not.

Ans:

```
public class Main {
public static void main(String[] args) {
int num = 29;
boolean flag = false;
for (int i = 2; i <= num / 2; ++i) {
```

```
// condition for nonprime number
if (num % i == 0) {
    flag = true;
    break;
}
}

if (!flag)
    System.out.println(num + " is a prime number.");
else
    System.out.println(num + " is not a prime number.");
}
}
```

11. Explain logical & Relational operators in Java with example

Ans:

Logical Operators :

Assume Boolean variables A holds true and variable B holds false then the following table lists the logical operators

Operator	Description	Syntax
&&	Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
	Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.	(A B) is true.
!	Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true.

The following program Demonstrates the use of logical operators.

```
public class Test
{
    public static void main(String args[])
    {
        boolean a = true;
        boolean b = false;
        System.out.println("a && b = " + (a&&b));
        System.out.println("a || b = " + (a|b) );
        System.out.println("!(a && b) = " + !(a && b));
    }
}
```

```
}
```

Output from above program will be :

```
a && b = false  
a || b = true  
!(a && b) = true
```

Relational Operators :

The relational operators determine the relationship that one operand has to the other.

Operator	Meaning	Syntax
==	Equal to	(A == B)
!=	Not equal to	(A != B)
>	Greater than	(A > B)
<	Less than	(A < B)
>=	Greater than or equal to	(A >= B)
<=	Less than or equal to	(A <= B)

Following program demonstrate the use of Relational Operator ==, !=, >= and <=

```
class RelOptrDemo
```

```
{  
public static void main(String[] args)  
{  
int a = 10, b = 15, c = 15;  
System.out.println("Relational Operators and returned values");  
System.out.println(" a > b = " + (a > b));  
System.out.println(" a < b = " + (a < b));  
System.out.println(" b >= a = " + (b >= a));  
System.out.println(" b <= a = " + (b <= a));  
System.out.println(" b == c = " + (b == c));  
System.out.println(" b != c = " + (b != c));  
}  
}
```

Output from above program will be:

```
a > b = false  
a < b = true  
b >= a = true  
b <= a = false  
b == c = true  
b != c = false
```

12. Write a program to find the reverse of a number.

Ans:

```
class Main
{
public static void main(String[] args)
{
int num = 1234, reversed = 0;

System.out.println("Original Number: " + num);

// run loop until num becomes 0
while(num != 0) {

// get last digit from num
int digit = num % 10;
reversed = reversed * 10 + digit;

// remove the last digit from num
num /= 10;
}

System.out.println("Reversed Number: " + reversed);
}
}
```

13. Describe instance Of, dot (.) in Java with suitable example

Ans:

Instanceof :

In Java, instanceof is a keyword used for checking if a reference variable contains a given type of object reference or not. Following is a Java program to show different behaviors of instanceof. Henceforth it is known as a comparison operator where the instance is getting compared to type returning boolean true or false as in Java we do not have 0 and 1 boolean return types.

```
import java.io.*;

// Main class
class GFG {
public static void main(String[] args)
{
// Creating object of class inside main()
GFG object = new GFG();
// Returning instanceof
System.out.println(object instanceof GFG);
}
```



```
}  
}  
OUTPUT  
True
```

Dot operator :

It is just a syntactic element. It denotes the separation of class from a package, separation of method from the class, and separation of a variable from a reference variable. It is also known as separator or period or member operator.

o It is used to separate a variable and method from a reference variable.

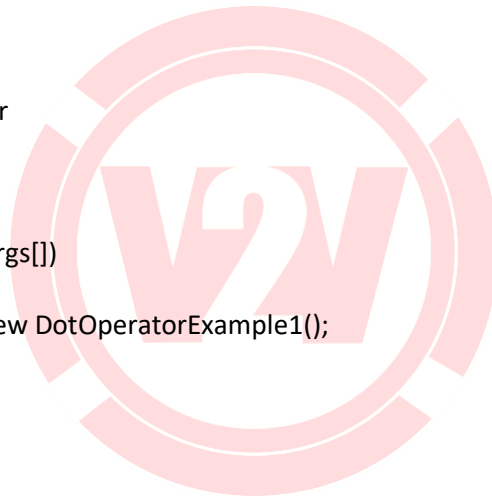
o It is also used to access classes and sub-packages from a package.

o It is also used to access the member of a package or a class.

```
public class DotOperatorExample1
```

```
{  
void display()  
{  
double d = 67.54;  
//casting double type to integer  
int i = (int)d;  
System.out.println(i);  
}  
public static void main(String args[])  
{  
DotOperatorExample1 doe = new DotOperatorExample1();  
//method calling  
doe.display();  
}  
}
```

```
Output  
67
```



14. Differentiate between String and StringBuffer.

Ans:

Sr. No	String	StringBuffer
1	The length of string object is fixed.	The length of StringBuffer can be increased.
2	The String object is immutable, that means we can not modify the string once created.	The StringBuffer class is immutable.
3	It is slower in performance.	It is faster in performance.
4	It consumes more memory.	It consumes less memory.

15. Differentiate between array and vector.

Ans:

Array	Vector
1) An array is a structure that holds multiple values of the same type.	1) The Vector is similar to array holds multiple objects and like an array; it contains components that can be accessed using an integer index.

2) An array is a homogeneous data type where it can hold only objects of one data type.	2) Vectors are heterogeneous. You can have objects of different data types inside a Vector.
3) After creation, an array is a fixed-length structure.	3) The size of a Vector can grow or shrink as needed to accommodate adding and removing items after the Vector has been created.
4) Array can store primitive type data element.	4) Vector are store non-primitive type data element.
5)Declaration of an array : <code>int arr[] = new int [10];</code>	5)Declaration of Vector: <code>Vector list = new Vector(3);</code>
6) Array is the static memory allocation.	6) Vector is the dynamic memory allocation.

16. List any four methods of String & StringBuffer class and state the use of each.

Ans:

String :

Method	Description
s1.charAt(position)	Return the character present at the index position.
s1.compareTo(s2)	If s1 < s2 then it returns positive. If s1 > s2 then it return negative and if s1 = s2 then it returns zero.
s1.concat(s2)	It returns the concatenated string of s1 and s2.
s1.equals(s2)	If s1 and s2 are both equal then it returns true.
s1.equalsIgnoreCase(s2)	By ignoring case, if s1 and s2 are equal then it returns true.
s1.indexOf('c')	It returns the first occurrence of character 'c' in the string s1.
s1.indexOf('c', n)	It returns the position of 'c' that occur at after nth position in string s1.
s1.lenght()	It gives the length of string s1.
String.valueOf(var)	Converts the value of the variable passed to it into string type.

StringBuffer:

Name of method	Description
append(String str)	Appends the string to the buffer
capacity()	It returns the capacity of the string buffer
charAt(int index)	It returns a specific character from the sequence which is specified by the index.
delete(int start, int end)	It deletes the characters from the string specified by the starting and ending index.
insert(int offset, char ch)	It inserts the character at the positions specified by the offset.
length()	It return the length of the string buffer.
setCharAt(int index, char ch)	The character specified by the index from the stringbuffer is set to ch.

17.Explain four methods of vector class with examples?

Ans:

Methods	Description
void addElement(object)	For adding the element in the vector
Object elementAt(int index)	Return the element present at specified location(index) in vector.
void insertElementAt(object obj, int pos)	For inserting the element in the vector specified by its position.
boolean removeElement(object ele)	Removes the specified element
void removeAllElements()	For removing all the elements from the vector.
void removeAllElements(int pos)	The elements specified by its position gets deleted from the vector.
int capacity()	Returns the capacity of the vector.
int size()	It returns the total no. of elements present in the vector.
boolean isEmpty()	Return true if the vector is empty.
Object firstElement()	Return the first element of the vector.
int indexOf(object ele)	Returns the index of corresponding element in the vector.
void setSize(int size)	This method is for setting the size of the vector.

18. Differentiate between class and interfaces

Ans:

Class	Interface
It has instance variable.	It has final variable.
It has non abstract method.	It has by default abstract method.
We can create object of class.	We can,,t create object of interface.
Class has the access specifiers like public, private, and protected.	Interface has only public access specifier
Classes are always extended.	Interfaces are always implemented.
The memory is allocated for the classes.	We are not allocating the memory for the interfaces.
Multiple inheritance is not possible with classes	Interface was introduced for the concept of multiple inheritance
<pre>class Example { void method1() { Body } void method2() { body } }</pre>	<pre>interface Example { int x =5; void method1(); void method2(); }</pre>

19. Describe interface in Java with suitable example.

Ans:

Interface is also known as kind of a class. Interface also contains methods and variables but with major difference, the interface consist of only abstract method (i.e. methods are not defined, these are declared only) and final fields (shared constants). This means that interface do not specify any code to implement those methods and data fields contains only constants. Therefore, it is the responsibility of the class that implements an interface to define the code for implementation of these methods. An interface is defined much like class.

Syntax:

```
access interface InterfaceName
{
return_type method_name1(parameter list);
....
}
```

```
return_type method_nameN(parameter list);  
type final-variable 1 = value1;
```

....

```
type final-variable N = value n;  
}
```

Example :

```
interface Area // interface defined
```

```
{  
final static float pi=3.14F;  
float Cal(float x, float y);  
}
```

```
class Rectangle implements Area
```

```
{  
public float Cal(float x, float y)  
{  
return(x * y);  
}  
}
```

```
class Circle implements Area
```

```
{  
public float Cal(float x, float y)  
{  
return(pi *x * x);  
}  
}
```

```
class InterfaceDemo
```

```
{  
Public static void main(String args[ ])  
{  
Rectangle r = new Rectangle( );  
Circle c = new Circle( );  
Area a ; // interface object  
a = r;  
System.out.println("Area of Rectangle=" +a.Cal(10,20));  
a = c;  
System.out.println("Area of Circle=" +a.Cal(10, 0));  
}}  
}
```

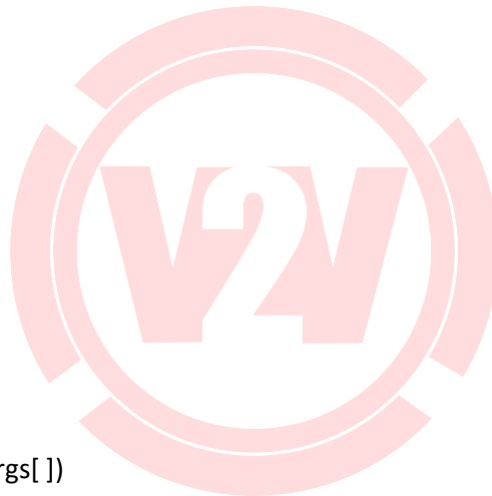
Output :-

```
C:\java\jdk1.7.0\bin> javac InterfaceDemo.java
```

```
C:\java\jdk1.7.0\bin> java InterfaceDemo
```

```
Area of Rectangle=200
```

```
Area of Circle=314
```



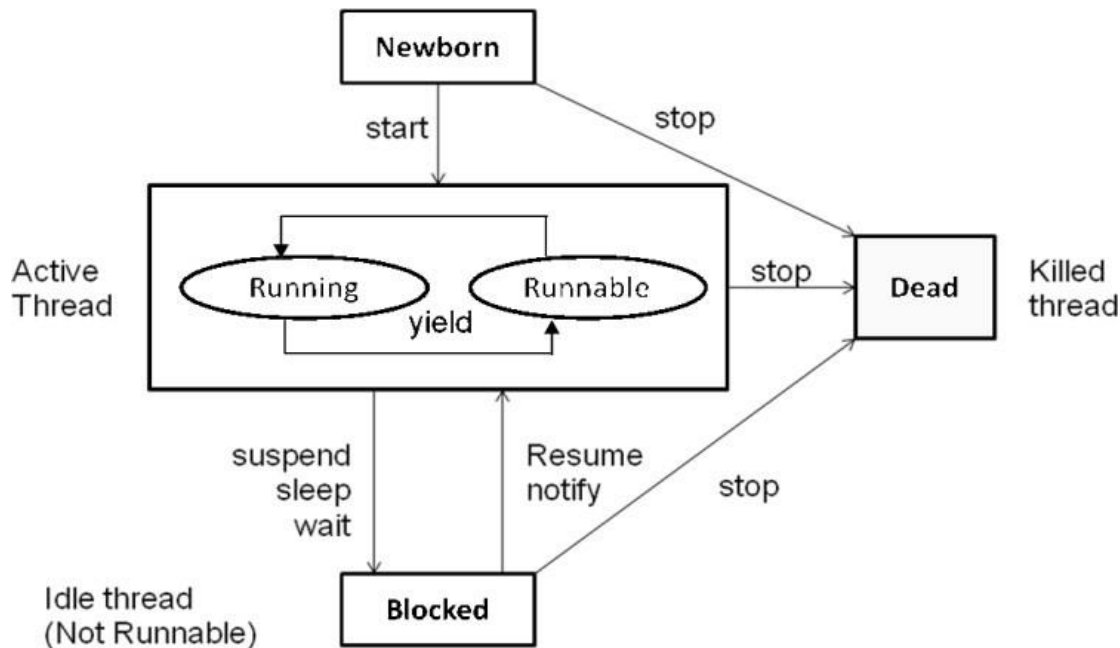
20. Explain the life cycle of thread.

Ans:

During the life time of a thread ,there are many states it can enter, They are:

1. Newborn state
2. Runnable state
3. Running state
4. Blocked state
5. Dead state

A thread is always in one of these five states. It can move from one state to another via a variety of ways.



1) New born state :

- Whenever a new thread is created, it is always in the new state.
- At this time we can scheduled it for running, using start() method or kill it using stop () method.
- If we attempt to use any other method at this stage, an exception will be thrown.

2) Runnable state :

- The thread is ready for execution and is waiting for the availability of the processor.
- The threads has joined waiting queue for execution.
- If all threads have equal priority, then they are given time slots for execution in round robin fashion. i.e. first-come, first serve manner.
- This process of assigning time to thread is known as time-slicing.

- If we want a thread to relinquish(leave) control to another thread of equal priority before its turn comes, then yield() method is used.

3) Running state :

- The processor has given its time to the thread for its execution.
- The thread runs until it relinquishes control on its own or it is preempted by a higher priority thread.
- A running thread may change its state to another state in one of the following situations.
 - 1) When It has been suspended using suspend() method.
 - 2) It has been made to sleep().
 - 3) When it has been told to wait until some events occurs.

4) Blocked state/ Waiting :

- A thread is waiting for another thread to perform a task. The thread is still alive.
- A blocked thread is considered “not runnable” but not dead and so fully qualified to run again.

5) Dead state/ Terminated :

- Every thread has a life cycle. A running thread ends its life when it has completed executing its run () method.
- It is natural death. However we can kill it by sending the stop message.
- A thread can be killed as soon it is born or while it is running or even when it is in “blocked” condition.

21.Explain the command line arguments with suitable example.

Ans:

- The command line argument is the argument passed to a program at the time when it is run.
- To access the command-line argument inside a JAVA program is quite easy, they are stored as string in String array passed to the args parameter of main() method.

Example

```
class demo
{
public static void main(String args[])
{
int n1=Integer.parseInt(args[0]);
int n2=Integer.parseInt(args[1]);
int add=n1+n2;
System.out.println(add);
}
}
C:\>javac demo.java
C:\>java demo
10 20
30
```

22. Write a program to check whether the string provided by the user is palindrome or not.

Ans:

```
class Main {
public static void main(String[] args) {
String str = "Radar", reverseStr = "";
int strLength = str.length();
for (int i = (strLength - 1); i >=0; --i) {
reverseStr = reverseStr + str.charAt(i);
}
if (str.toLowerCase().equals(reverseStr.toLowerCase())) {
System.out.println(str + " is a Palindrome String.");
}
else {
System.out.println(str + " is not a Palindrome String.");
}
}
}
```

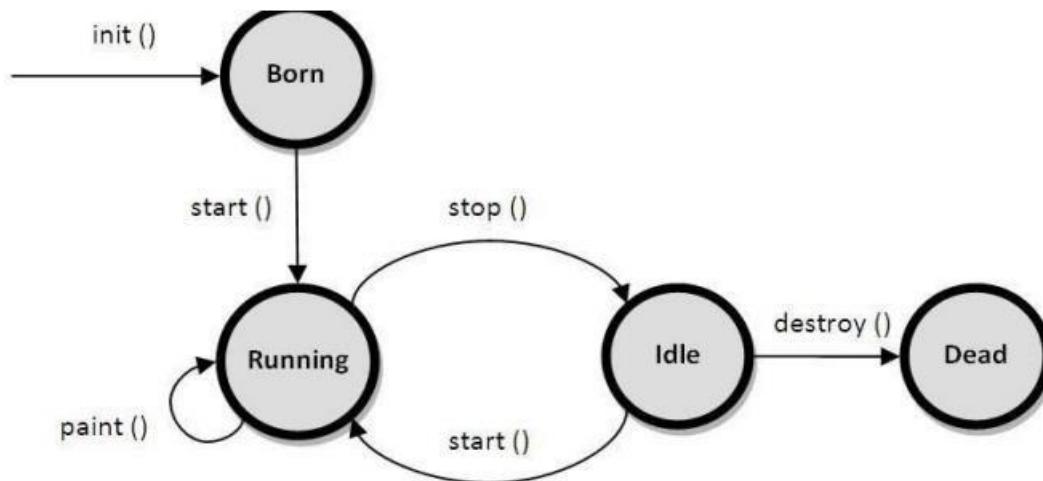
23. Differentiate between applet and application

Ans:

Applet	Application
Applet does not use main() method for initiating execution of code	Application use main() method for initiating execution of code
Applet cannot run independently	Application can run independently
Applet cannot read from or write to files in local computer	Application can read from or write to files in local computer
Applet cannot communicate with other servers on network	Application can communicate with other servers on network
Applet cannot run any program from local computer.	Application can run any program from local computer.
Applet are restricted from using libraries from other language such as C or C++	Application are not restricted from using libraries from other language

24. Applet life Cycle

Ans:



Applets are small applications that are accessed on an Internet server, transported over the Internet, automatically installed, and run as part of a web document. The applet states include: Born or initialization state Running state Idle state Dead or destroyed state

a) Born or initialization state:

Applet enters the initialization state when it is first loaded. This is done by calling the `init()` method of Applet class. At this stage the following can be done: Create objects needed by the applet Set up initial values Load images or fonts Set up colors Initialization happens only once in the life time of an applet.

```

public void init()
{
//implementation
}
  
```

b) Running state:

Applet enters the running state when the system calls the `start()` method of Applet class. This occurs automatically after the applet is initialized. `start()` can also be called if the applet is already in idle state. `start()` may be called more than once. `start()` method may be overridden to create a thread to control the applet.

```

public void start()
{
//implementation
}
  
```

```
}
```

c) Idle or stopped state:

An applet becomes idle when it is stopped from running. Stopping occurs automatically when the user leaves the page containing the currently running applet. stop() method may be overridden to terminate the thread used to run the applet.

```
public void stop()
{
//implementation
}
```

d) Dead state:

An applet is dead when it is removed from memory. This occurs automatically by invoking the destroy method when we quit the browser. Destroying stage occurs only once in the lifetime of an applet. destroy() method may be overridden to clean up resources like threads.

```
Public void destroy()
{
//implementation
}
```

e) Display state:

Applet is in the display state when it has to perform some output operations on the screen. This happens after the applet enters the running state. paint() method is called for this. If anything is to be displayed the paint() method is to be overridden.

```
public void paint(Graphics g)
{
//implementation
}
```

25. Explain the Param tag of applet. Write an applet to accept user name in the form of parameter and print Hello + username

Ans:

To pass parameters to an applet tag is used. Each tag has a name attribute and a value attribute. Inside the applet code, the applet can refer to that parameter by name to find its value.

The syntax of tag is as follows

```
<PARAM NAME = name1 VALUE = value1>
```

Program for an applet to accept user name in the form of parameter and print „Hello“

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

```
String str;
public void init()
{
str = getParameter("username");
str = "Hello "+ str;
}
public void paint(Graphics g)
{
g.drawString(str,10,100);
}
}
```

```
<HTML>
<Applet code = hellouser.class|| width = 400 height = 400>
<PARAM NAME = "username" VALUE = abc>
</Applet>
</HTML>
```

26. Write a simple applet program which displays three concentric circles.

Ans:

```
import java.awt.*;
import java.applet.*;
public class CircleDemo extends Applet
{
public void paint (Graphics g)
{
g.drawOval(100,100,190,190);
g.drawOval(115,115,160,160);
g.drawOval(130,130,130,130);
}
}
```

```
<html><applet code=||CircleDemo.class|| height=300 width=200>
</applet>
</html>
```

27. Explain drawArc () , fillOval () , drawOval () , drawPolygon () , drawRect()

Ans:

drawArc ()

It is used to draw arc

Syntax:

```
void drawArc(int x, int y, int w, int h, int start_angle, int sweep_angle);
```

```
fillOval ( )
```

Draws an oval within a bounding rectangle whose upper left corner is specified by top, left. Width and height of the oval are specified by width and height.

Syntax-

```
void fillOval(int top, int left, int width, int height)
```

```
. drawOval( )
```

To draw an Ellipses or circles used drawOval() method can be used.

Syntax:

```
void drawOval( int top, int left, int width, int height)
```

```
drawRect( )
```

The drawRect() method display an outlined rectangle

Syntax:

```
void drawRect(int top, int left, int width, int height)
```

28.Explain Font Class & Write method to set font of a text and describe its parameters.

Ans:

Font class A font determines look of the text when it is painted. Font is used while painting text on a graphics context & is a property of AWT component.

The Font class defines these variables:

Variable	Meaning
String name	Name of the font
float pointSize	Size of the font in points
int size	Size of the font in point
int style	Font style

Sr. No	Methods	Description
1	static Font decode(String <i>str</i>)	Returns a font given its name.
2	boolean equals(Object <i>FontObj</i>):	Returns true if the invoking object contains the same font as that specified by <i>FontObj</i> . Otherwise, it returns false .
3	String toString()	Returns the string equivalent of the invoking font.
4	String getFamily()	Returns the name of the font family to which the invoking font belongs.
5	static Font getFont(String <i>property</i>)	Returns the font associated with the system property specified by <i>property</i> . null is returned if <i>property</i> does not exist.
6	static Font getFont(String <i>property</i> , Font <i>defaultFont</i>)	Returns the font associated with the System property specified by <i>property</i> . The font specified by <i>defaultFont</i> is returned if <i>property</i> does not exist.
7	String getFontName()	Returns the face name of the invoking font.
8	String getName()	Returns the logical name of the invoking font.
9	int getSize()	Returns the size, in points, of the invoking font.
10	int getStyle()	Returns the style values of the invoking font.
11	int hashCode()	Returns the hash code associated with the invoking object.
12	boolean isBold()	Returns true if the font includes the BOLD style value. Otherwise, false is returned.
13	boolean isItalic()	Returns true if the font includes the ITALIC style value. Otherwise, false is returned.
14	boolean isPlain()	Returns true if the font includes the PLAIN style value. Otherwise, false is returned.

29.What are streams ? Write any five methods of character stream classes

Ans:

Definition: The java. IO package contain a large number of stream classes that provide capabilities for processing all types of data. These classes may be categorized into two groups based on the data type on which they operate.

Methods of Reader Class

- 1) void mark(int numChars) : Places a mark at the current point in the input stream that will remain valid until numChars characters are read.
- 2) boolean markSupported() : Returns true if mark() / reset() are supported on this stream.
- 3) int read() : Returns an integer representation of the next available character from the invoking input stream. -1 is returned when the end of the file is encountered.
- 4) int read(char buffer[]) : Attempts to read up to buffer. Length characters into buffer and returns the actual number of characters that were successfully read. -1 is returned when the end of the file is encountered.
- 5) abstract int read(char buffer[],int offset,int numChars): Attempts to read up to numChars characters into buffer starting at buffer[offset], returning the number of characters successfully read.-1 is returned when the end of the file is encountered.

Methods of Writer class are listed below: -

- 1) abstract void close() : Closes the output stream. Further write attempts will generate an IOException.
- 2) abstract void flush() : Finalizes the output state so that any buffers are cleared. That is, it flushes the output buffers.
- 3) void write(intch): Writes a single character to the invoking output stream. Note that the parameter is an int, which allows you to call write with expressions without having to cast them back to char.
- 4) void write(char buffer[]): Writes a complete array of characters to the invoking output stream
- 5) abstract void write(char buffer[],int offset, int numChars) :- Writes a subrange of numChars characters from the array buffer, beginning at buffer[offset] to the invoking output stream.

30.Write any four methods FileReader class

Ans:

1. public int read()throws IOException – Reads a single character.
2. public int read(char[] cbuf, int offset, int length) throws IOException – Reads characters into a portion of an array.
3. public void close()throws IOException – Closes the stream and releases any system resources associated with it. Once the stream has been closed, further read(), ready(), mark(), reset(), or skip() invocations will throw an IOException. Closing a previously closed stream has no effect

4. `public boolean ready()throws IOException` – Tells whether this stream is ready to be read. An `InputStreamReader` is ready if its input buffer is not empty, or if bytes are available to be read from the underlying byte stream
5. `public void mark(int readAheadLimit) throws IOException` – Marks the present position in the stream. Subsequent calls to `reset()` will attempt to reposition the stream to this point. Not all character-input streams support the `mark()` operation.

31. Write any four methods of File and FileInputStream class each.

Ans:

File Class Methods:

1. `String getName()` – returns the name of the file.
2. `String getParent()` – returns the name of the parent directory.
3. `boolean exists()` – returns true if the file exists, false if it does not.
4. `void deleteOnExit()` – Removes the file associated with the invoking object when the Java Virtual Machine terminates.
5. `boolean isHidden()` - Returns true if the invoking file is hidden. Returns false otherwise.

FileInputStream Class Methods:

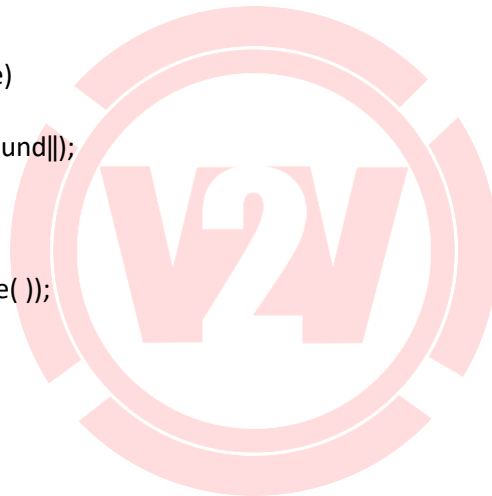
1. `int available()` - Returns the number of bytes of input currently available for reading.
2. `void close()` - Closes the input source. Further read attempts will generate an `IOException`.
3. `void mark(int numBytes)` – Places a mark at the current point in the inputstream that will remain valid until `numBytes` bytes are read.
4. `boolean markSupported()` – Returns true if `mark()/reset()` are supported by the invoking stream.
5. `int read()` - Returns an integer representation of the next available byte of input. `-1` is returned when the end of the file is encountered.
6. `int read(byte buffer[])` - Attempts to read up to `buffer.length` bytes into `buffer` and returns the actual number of bytes that were successfully read. `-1` is returned when the end of the file is encountered.

32. Write a program to copy contents of one file to another file using character stream class.

Ans:

```
import java.io.*;
class CopyData
{
public static void main(String args[ ])
{
//Declare input and output file stream
FileInputStream fis= null; //input stream
```

```
FileOutputStream fos=null; //output Stream
//Declare a variable to hold a byte
byte byteRead;
try
{
// connect fis to in.dat
fis=new FileInputStream("in.dat");
// connect fos to out.dat
fos= new FileOutputStream("out.dat");
//reading bytes from in.dat and write to out.dat
do {
byteRead =(byte)fis.read( );
fos.write(byteRead);
}
while(byteRead != -1);
}
Catch(FileNotFoundException e)
{
System.out.println("file not found");
}
Catch(IOException e)
{
System.out.println(e.getMessage( ));
}
finally // close file
{
try
{
fis.close( );
fos.close( );
}
Catch(IOException e)
{ }
}
}
```



33.What is the use of ArrayList Class ? State any three methods with their use from ArrayList

Ans:

1. ArrayList supports dynamic arrays that can grow as needed.

2. ArrayList is a variable-length array of object references. That is, an ArrayList can dynamically increase or decrease in size. Array lists are created with an initial size. When this size is exceeded, the collection is automatically enlarged. When objects are removed, the array may be shrunk.

Methods of ArrayList class :

1. void add(int index, Object element)

Inserts the specified element at the specified position index in this list. Throws IndexOutOfBoundsException if the specified index is out of range ($\text{index} < 0$ || $\text{index} > \text{size}()$).

2. boolean add(Object o)

Appends the specified element to the end of this list.

3. boolean addAll(Collection c)

Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator. Throws NullPointerException if the specified collection is null.

4. boolean addAll(int index, Collection c)

Inserts all of the elements in the specified collection into this list, starting at the specified position. Throws NullPointerException if the specified collection is null.

5. void clear()

Removes all of the elements from this list.

6. Object clone()

Returns a shallow copy of this ArrayList.

34. Write any four mathematical & date functions used in Java

Ans:

1) min() :

Syntax: static int min(int a, int b)

Use: This method returns the smaller of two int values.

2) max() :

Syntax: static int max(int a, int b)

Use: This method returns the greater of two int values.

3) sqrt()

Syntax: static double sqrt(double a)

Use : This method returns the correctly rounded positive square root of a double Values

4) pow(double a, double b)

Use : This method returns the value of the first argument raised to the power of the second argument.

5) exp()

Syntax: static double exp(double a)

Use : This method returns Euler's number e raised to the power of a double value.

6) round() :

Syntax: static int round(float a)

Use : This method returns the closest int to the argument.

7) abs()

Syntax: static int abs(int a)

Use : This method returns the absolute value of an int value.

Date class :

i) getTime ()

ii) getDate ()

The Date class encapsulates the current date and time.

i. getTime():

Syntax: long getTime()

Returns the number of milliseconds that have elapsed since January 1, 1970.

ii. getDate()

Syntax: public int getDate()

Returns the day of the month. This method assigns days with the values of 1 to 31.

1. setTime():

void setTime(long time)

the parameter time - the number of milliseconds. Sets this Date object to represent a point in time that is time milliseconds after January 1, 1970 00:00:00 GMT

2. getDay()

int getDay():

Returns the day of the week represented by this date. The returned value (0 = Sunday, 1 = Monday, 2 = Tuesday, 3 = Wednesday, 4 = Thursday, 5 = Friday, 6 = Saturday) represents the day of the week that contains or begins with the instant in time represented by this Date object, as interpreted in the local time zone.

35. What is use of setclass ? Write a program using setclass.

Ans:

The Set interface defines a set. It extends Collection and declares the behavior of a collection that does not allow duplicate elements. Therefore, the add() method returns false if an attempt is made to add duplicate elements to a set. The Set interface contains only methods inherited from Collection and adds the restriction that duplicate elements are prohibited. Set also adds a stronger contract on the behavior of the equals and hashCode operations, allowing Set instances to be compared meaningfully even if their implementation types differ.

```
import java.util.*;
```

```
public class SetDemo
```

```
{
```

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

```
{
```

```
int count[] = {34, 22,10,60,30,22};
```

```
Set set = new HashSet();
```

```
Try
{
for(int i = 0;i<5;i++){
set.add(count[i]);
System.out.println(set);
TreeSet sortedSet = new TreeSet(set);
System.out.println("The sorted list is:");
System.out.println(sortedSet);
System.out.println("The First element of the set is: "+ (Integer)sortedSet.first());
System.out.println("The last element of the set is: "+ (Integer)sortedSet.last());
}
catch(Exception e)
{}
}
}
```

Executing the program.

[34, 22, 10, 30, 60]

The sorted list is:

[10, 22, 30, 34, 60]

The First element of the set is: 10

The last element of the set is: 60

