# JAVA (401)

## QUESTIONS:

01. What is java? Discuss the features and application of java.

02. What is class and object? Write a Java program to demonstrate the concept of class and object.

03. What is constructor? Discuss different types of constructors. Write a Java program to calculate area and circumference of a circle using constructor.

04. What is inheritance? Discuss the types of inheritance with example.

05.What are control statements? Explain the different types of control statements with suitable example

06.Explain interface in java. Write a Java program to show the concept of interface.

07.What are looping control statements? Discuss the different types of loops with suitable example.

08. Define an array. Discuss its type. Write a Java program to enter ten number in an array prints the number in sorted order.

09. What is an applet. Discuss the life cycle of an applet. Write an applet code to display a message "Welcome to Java programming".

10.What is vector? Write a Java program to demonstrate the use of some of the methods of vectors class.

11. What is multithreading? Discuss the life cycle of a thread in details. Write a Java program for creating demonstration of threads by extending the thread class.

12.Define package. What are the advantages of using package? Write down the steps to create a package.

13.What is method overloading? Write a Java program to calculate perimeter of circle, square and rectangle using method overloading.

14.What is Java Virtual Machine (JVM). What are different component of JVM?

15.What is method overriding? Write a Java program to demonstrate the concept of method overriding.

16. Write Java Program for the following: –
(i) Write an applet code to demonstrate the use of various methods of graphics class.
(ii) Write an applet code to demonstrate passing of parameters to an applet.
(iii) Write an applet code to display name, class, roll no. and address.
(iv) Write a Java program to add two matrices and print the result matrix.
(v) Write a Java program to print all prime numbers from 0 to 100

17. Write the difference between the following: –
(i) core Java and advanced java
(ii) C++ and Java programming language.
(iii) local and remote applet
(iv) "for loop" and "do while loop".

18. Write short note on the following: –
(i) Abstract class (ii) Java string (iii) final

**01. What is Java? Discuss the features and application of Java.**

**Ans:** –Java is a high level, robust, object–oriented and secure programming language developed by Sun Microsystem in 1995. Java is developed by James Gosling and his team. When Java is introduced firstly its name was OAK later it is renamed as Java. The Java language is easy to learn and simple to use. The Java language is similar to C and C++ language but Java does not provide low level programming functionality like pointer, structure, union etc. The Java code are always written in the form of class and object. The Java language is fast, secure, platform independent and reliable language so, it is widely used in all over the world. the Java code save with .java extension and it use both compiler and interpreter to run Java code. The Java codes are first compiled into byte code (machine–independent code). Then the byte code runs on Java Virtual Machine (JVM) regardless of the underlying architecture. The Java use some components of C languages such as array, string, looping etc. and C++ languages such as constructor, class and object, inheritance etc. and also use various new component such as Applet, AWT etc. The Java languages does not use some c languages components like pointer, structure, union and C++ components like virtual, operator overloading, pure virtual etc. Java is used in all kinds of applications like Mobile Applications (Android is Java–based), desktop applications, web applications, client–server applications, enterprise applications, and many more.

**Example: Program in Java to print sum of two number**

```
import java.util.*;
class sum
{
public static void main(String arg[])
{
int x,y,z;
Scanner obj=new Scanner(System.in);
System.out.println("Enter 2 Numbers–");
x=obj.nextInt();
y=obj.nextInt();
z=x+y;
System.out.println("sum ="+z);
}}
```

**OUTPUT:**

```
B:\Java>javac j1.java
B:\Java>java sum
Enter 2 Numbers–
4
6
sum =10
```

**Features of Java languages**

**(i) Simple –**Java is very simple languages. It is easy to learn, and its syntax is simple, less complex and easy to understand. The Java does not use complex functionalities like pointer, goto etc.

**(ii) Object–oriented –** Java is an object–oriented programming language. Java supports major Object–Oriented programming features like Encapsulation, Abstraction, function overloading, Inheritance etc.

**(iii) Robust –** Java is robust because it is capable of handling run–time errors, support strong memory management, automatic garbage collection, exception handling, and it avoids explicit pointer concept.

**(iv) Platform Independent –** Java is platform independent programming language. Because it is different from other languages like C, C++, etc. which are compiled into platform specific machines while Java is a write once, run anywhere language.

**(v) Portable –** Java is portable because it facilitates us to carry the Java bytecode to any platform. It doesn't require any implementation.

**(vi) Secure –** Java is a more secure language as compared to C/C++, as it does not allow a programmer to explicitly create pointers and Java Programs run inside a virtual machine sandbox.

**(vii) Architecture–neutral** – Java compiler generates an architecture–neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.

**(viii) Multithreading** – Java provide multithreading feature. With the use of this feature, it is possible to write programs that can perform many tasks simultaneously.

**Application of Java** –Java is used in many applications for different purposes. Some of the major field where Java is used are as:

**(i) Mobile App Development** – The Java programming language can be considered as the official language for mobile application development. Most of the android applications build using Java.

**(ii) Scientific Applications** – Java has powerful security and robustness features that make it popular for developing scientific applications. Java also provides powerful mathematical calculations that give the same results on different platforms.

**(iii) Cloud–Based Applications** – A cloud application is the on–demand availability of IT resources via the internet. The cloud–based application provides the service at a low cost. Java provides the environment to develop cloud–based applications.

**(iv) Desktop GUI Applications**– We can also develop a GUI application using Java. Java provides AWT, JavaFX, and Swing for developing the GUI application.

**(v) Enterprise Applications** – Java becomes the first choice for the development of enterprise applications because of its features. Most of the enterprise applications are based on Java because it is the robust, secure, powerful, scalable language.

**(vi) Web–based Applications** – Java supports the development of web–applications with the help of servlets, struts, JSP (Java Server Pages) and JSF (Java Server Faces), Spring etc. With the help of these technologies, we can develop any kind of web–based application.

**02. What is class and object? Write a Java program to demonstrate the concept of class and object.**

**Ans: Class** – In object–oriented programming language, class create container and implement data encapsulation feature. It is a basic concept of Object–Oriented Programming which revolve around the real–life entities. The class create container in which we can store the data member and Member function. It is the template or blueprint from which objects are created. the class use "class" keyword for declaration followed by class name. In java, the class create class file save with .class extension, The class data member and member function is only accessible by using inheritance or object of the same class. The class does not take any space in the memory. In java, the class can also contain the main function and the Java code run by using these class name.

**Syntax of Class**

```
class <class name>
{
   data member;
   member function;
}
```

**Example:**

```
class student
{
   int x,y;
   public void print()
   {
        System.out.print("Hello");
   }
```

**Object –**The object is a basic unit of Object–Oriented Programming and represents the real–life entities. In Object–Oriented Programming using object we can implement the data abstraction feature. The object use class name followed by the object name for declaration and it must be declared in the main function. In java, the object is initialized and allocate the memory for the object by using "new' operator. The object creates separate memory block for each instance of the class and assign its properties according to the class for which it is object. Using object, we can access the class data member and Member function by using dot(.) operator where object only access the public data member and Member function of the class. The object takes space in the memory and it is created at runtime and it is run time entity. We can create n number of objects of a class.

**Syntax:** <classname> <objectname> = new <classname>();

**Example:** student s = new student();

**Program:**

```
import java.util.*;
class circle
{
float r,ar;
public void get()
{
Scanner sc=new Scanner(System.in);
System.out.print("Enter radius of circle: –");
r=sc.nextFloat();
}
public void calc()
{
ar=3.14f*r*r;
}
public void show()
{
System.out.println("Area of Circle="+ar);
}}
class ex
{
public static void main(String arg[])
{
circle c=new circle();
c.get();
c.calc();
c.show();
} }
```

**OUTPUT:**

```
B:\Java>javac cir.java
B:\Java>java ex
Enter radius of circle: –3.4
Area of Circle=36.298405
```

**03. What is constructor? Discuss different types of constructors. Write a Java program to demonstrate the concept of constructor. / Write a Java program to calculate area and perimeter of a circle using constructor.**

**Ans: –** A constructor in Java is a special method that is used to initialize objects. It can be used to set initial values for object attributes. In java, the constructor use same name as class name and It is declare publicly within the class. The constructor does not have any return type even void and it may or may not accept the parameters. The constructor is automatically called whenever object of class is created using a new() keyword.

In java, the constructor is of two types: –

**(i) Default Constructor: –** The default constructor also known as non–parametrized constructor. it is a type of constructor which do not accept any parameter it has empty parameter list. In a class we can define only one default constructor because it has no return type and no parameters so, it cannot be overloaded.

The default constructor is automatically called whenever object of class is created and it do not pass any argument.

**Syntax:**

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

**Example:**

```
circle()
{
pi=3.14f;
}
```

**(ii) Parameterized Constructor: –** The parametrized constructor is also known as argument constructor. Whenever different data member of class needs to be initialized with a different value the parametrized constructor can be defined. The parametrized constructor is a type of constructor that accept one or more parameter at the time of declaration of object and initialize the data member of the object with these parameters. In a class we can Define Multiple parametrized constructors by overloading. The parametrized constructor is automatically called whenever object of class is created and it pass any argument.

**Syntax:**

```
<class_name>(parameters)
{
Statement;
}
```

**Example:**

```
circle(int x)
{
r=x;
}
```

**Program:**

```
import java.util.*;
class circle
{
float r,ar,pi,c;
public circle() //default constructor
{
System.out.println("Welcome");
}
public circle(float x) // Parameterized
constructor
{
r=x;
pi=3.14f;
}
public void calc()  {
ar=pi*r*r;
c=2.0f*pi*r;
}
public void show()
{
System.out.println("Area of Circle="+ar);
System.out.println("Circumference of
Circle="+c);
}}
class area
{
public static void main(String arg[])
{
circle c1=new circle(); //calling of default
constructor
float r;
Scanner sc=new Scanner(System.in);
System.out.print("Enter radius of circle: –");
r=sc.nextFloat();
circle c2=new circle(r); // calling of
Parameterized constructor
c2.calc();
c2.show();
}}
```

**OUTPUT:**

```
B:\Java>javac cons.java
B:\Java>java area
Welcome
Enter radius of circle: –3.4
Area of Circle=36.298405
Circumference of Circle=21.352001
```

**04. What is inheritance? Discuss the types of inheritance with example.**

**Ans:** – The inheritance implements the reusability feature.so, using inheritance we can inherit one class property to another class The inheritance provides another form of data abstraction. It is an important part of OOPs (Object Oriented programming system). The inheritance establishes a relationship between two or more classes. In simple word, we can say that the inheritance is a mechanism of designing a new class from the old class in such a way that the new class acquire or inherit all the data member and Member function of the old class. By using inheritance, the new class can only access the public data member and member function of the old class. The new class has properties and behaviour of the existing class as well as its own unique properties and behaviour.

There are two types of classes in inheritance:

**(i) Super class –** The class that is inherit the new class or the class from which a new class is inherited is called super class. It is also known as base or parent class.

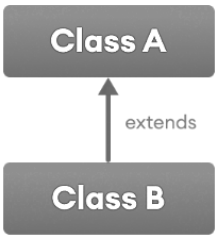
**(ii) sub class –** The new class which is inherited by the old class is called sub class. It is also known as child or derived class

In java "extends" keyword is used followed by the super class name to inherit the properties of super class to the sub class.

**Syntax:–**

```
class Subclass–name extends Superclass–name
{
data member;
member function();
}
```

**Types of inheritance:** In java, there are mainly 3 types of inheritance:(java does not support multiple and hybrid inheritance by using class)




**(i) Single inheritance** – Single inheritance is the type of inheritance in which a derived class is inherited from a single base class the derived class acquires all the properties and behaviour of the single base class and it also has its own unique properties and behaviour.

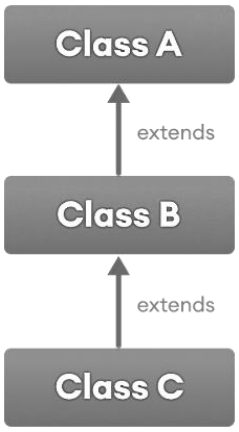
**Program:**

```
import java.util.*;
class base
{
public void show1()
{
System.out.println("Base Class");
}
}
class derived extends base
{
public void show2()
{
System.out.println("Derived class");
}
}
class si
{
public static void main(String arg[])
{
derived d=new derived();
d.show1();
d.show2();
}
}
```

**OUTPUT:**

```
B:\Java>javac single.java
B:\Java>java si
Base Class
Derived class
```

**(ii) Multilevel inheritance** – Multilevel inheritance is a type of inheritance in which one class is inherited by another class and it also inherit another class. In this inheritance, a derived class will be inherited from a base class as well as the derived class also act as the base class to other class. The lower-level classes have properties and behaviour of all higher-level classes.



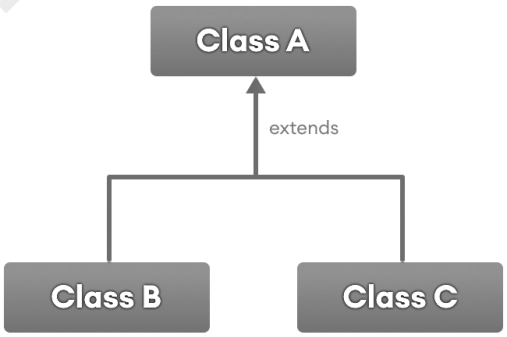

**Program:**

```
import java.util.*;
class base
{
public void show1()
{
System.out.println("Base Class");
}}
class middle extends base
{
public void show2()
{
System.out.println("middle class");
}}
class derived extends middle
{
public void show3()
{
System.out.println("Derived class");
}}
class mli
{
public static void main(String arg[])
{
derived d=new derived();
d.show1();
d.show2();
d.show3();
}}
```

**OUTPUT:**

```
B:\Java>javac ml.java
B:\Java>java mli
Base Class
middle class
Derived class
```

**(iii) Hierarchical Inheritance** – Hierarchical inheritance is a type of inheritance in which multiple derived class is inherited from a single base class. In this type of inheritance all the derived class acquire the properties and behaviour of the base class. In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one subclass.




**Program:**

```
import java.util.*;
class base
{
public void show1()
{
System.out.println("Base Class");
}}
class derived1 extends base
{
public void show2()
{
System.out.println("Derived1 class");
}}
class derived2 extends base
{
public void show3()
{
System.out.println("Derived2 class");
}}
class hi
{
public static void main(String arg[])
{
derived1 d1=new derived1();
derived2 d2=new derived2();
d1.show1();
d1.show2();
d2.show1();
d2.show3();
}}
```

**OUTPUT:**

```
B:\Java>javac hie.java
B:\Java>java hi
Base Class
Derived1 class
Base Class
Derived2 class
```

**05. What are control statements? Explain the different types of control statements with suitable example.**

**Ans:-** A programming language uses control statements to control the flow of execution of a program based on certain conditions. It is also known as Decision–making statements or conditional statement. It evaluates the Boolean expression and control the program flow depending upon the result of the condition provided.

In Java there are four types of control statement: –

**(i) if statement:** if statement is the simplest control statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e. if a certain condition is true then a block of statement is executed otherwise not.

**Syntax :**

```
if(condition)
{
statements; //executes when condition is true
}
```

**Program: program to print gross total after discount if total is greater than 1000**

```
import java.util.*;
class price
{
public static void main(String arg[])
{
Scanner sc=new Scanner(System.in);
int r,q,g;
System.out.print("Enter rate and quantity: –");
r=sc.nextInt();
q=sc.nextInt();
g=r*q;
if(g>1000)
{
g=g–(g*10/100);
}
System.out.print("gross total = "+g);
}}
```

**OUTPUT:**

```
B:\Java>javac j8.java
B:\Java>java price
Enter rate and quantity: –40
100
gross total = 3600
```

**(ii) if–else statement –** The if–else statement is an extension to the if–statement, which uses another block of code, i.e., else block. In If else statement, if check the condition first if the condition is true then execute the if block statement and jump the else block statement otherwise if the condition is false then jump the if block statement and execute the else block statement.

**Syntax:**

```
if(condition)
{
statement 1; //executes when condition is true
}
Else
{
statement 2; //executes when condition is false
}
```

**Program: Program to check number is odd or even**

```
import java.util.*;
class check
{
public static void main(String arg[])
{
Scanner sc=new Scanner(System.in);
int n;
System.out.print("Enter number: –");
n=sc.nextInt();
if(n%2==0)
System.out.print("even number");
else
System.out.print("odd number");
}}
```

**OUTPUT:**

```
B:\Java>javac oe.java
B:\Java>java check
Enter number: –6
even number
```

**(3) if–else–if statement:** The if–else–if statement contains the if–statement followed by multiple else–if statements. If statement check the condition first, if the condition is true then execute the if block statement otherwise move to the next else if statement and as soon as one of the conditions of else if is true, the statement associated with that is executed, and the rest of the else if is bypassed. If none of the conditions is true, then the final else statement will be executed.

**Syntax:**

```
if(condition 1)
{
statement 1; //executes when condition 1 is true
}
else if(condition 2)
{
statement 2; //executes when condition 2 is true
}
else if(condition 3)
{
statement 3; //executes when condition 2 is true
}
else
{
statement 4; //executes when all the conditions are false
}
```

**Program: print total marks, percentage, division using else–if.**

```
import java.util.*;
class result
{
public static void main(String arg[])
{
Scanner sc=new Scanner(System.in);
float a,b,c,t,p;
System.out.print("Enter marks of 3 subject:");
a=sc.nextFloat();
b=sc.nextFloat();
c=sc.nextFloat();
t=a+b+c;
p=t/3;
System.out.println("total marks="+t);
System.out.println("percentage="+p+"%");
if(p>=60)
System.out.print("division=1st");
else if(p>=45)
System.out.print("division=2nd");
else if(p>=30)
System.out.print("division=3rd");
else
System.out.print("fail");
} }
```

```
OUTPUT:
B:\Java>javac j10.java
B:\Java>java result
Enter marks of 3 subject: –43
56
76
total marks=175.0
percentage=58.333332%
division=2nd
```

**(iv) Switch case:** In Java, Switch statements are similar to if–else–if statements. The switch statement contains multiple blocks of code called cases and a single case is executed based on the variable which is being switched. If no case of the switch case is executed then the default case statement will be executed. The switch statement is easier to use instead of if–else–if statements. It also enhances the readability of the program. The switch case also uses break statement to avoid the execution of next cases of the switch case.

**Syntax:**

```
switch (expression)
{
 case value1:
  statement1;
  break;
case value2:
  statement2;
  break;
.
case value N:
  statement N;
  break;
default:
  Default statement;
}
```

**Program: print given character is vowel or consonant using switch case.**

```
import java.util.*;
class check
{
public static void main(String arg[])
{
Scanner sc=new Scanner(System.in);
char ch;
System.out.print("Enter Alphabet: –");
ch=sc.nextLine().charAt(0);
switch(ch)
{
case 'a' :
System.out.print(ch +" is a Vowel");
break;
case 'e' :
System.out.print(ch +" is a Vowel");
break;
same for  i, o, u
default:
System.out.print(ch +" is a consonant");
} } }
```

```
OUTPUT:
B:\Java>javac j12.java
B:\Java>java check
Enter Alphabet: –u
u is a Vowel
```

**06.Explain interface in Java. Write a Java program to show the concept of interface.**

**Ans: –** An Interface in Java programming language is mechanism to achieve abstraction. It is defined as an abstract type used to specify the behaviour of a class. It is similar to class. We can say that, an interface is a blueprint of a class. A Java interface contains static constants and abstract member function. Writing an interface is similar to writing a class. But a class describes the attributes and behaviours of an object. And an interface contains behaviours that a class implements. An interface cannot create object and it is only accessible by using inheritance. There are mainly three reasons to use interface. They are given below.

(i) It is used to achieve abstraction. (ii) It can be used to achieve loose coupling.

(iii) By interface, we can support the functionality of multiple inheritance.

**Declaration of an interface –** An interface is declared by using the "interface" keyword. It provides total abstraction; means all the member function in an interface are declared with the empty body, and all the data member are public, static and final by default. A class that implements an interface must implement all the member function declared in the interface.

**Syntax:**

```
interface <interface_name>
{
    Final/static data member;
    Abstract member function;
}
```

**Example:**

```
Interface base
{
Public void get();
Public void print();
}
```

**Implementing Interfaces –** A class uses the "implements" keyword after the classname and before the interface name to implement an interface. A class can implement more than one interface.

**Syntax:**

```
Class <classname> implements <interface>
{
Statement;
}
```

**Example:**

```
Class derived implements base
{
Statement;
}
```

**Program:**

```
import java.util.*;
interface subtract
{
public void get();
public void show();
}
class derived implements subtract
{
Scanner sc=new Scanner(System.in);
int a,b;
public void get()
{
System.out.print("Enter two numbers: –");
a=sc.nextInt();
b=sc.nextInt();
}
public void show()
{
System.out.println("Subtract ="+(a–b));
}
}
class sub
{
public static void main(String arg[])
{
derived d=new derived();
d.get();
d.show();
}
}
```

**OUTPUT:**

```
B:\Java>javac inter.java
B:\Java>java sub
Enter two numbers: – 34
12
Subtract =22
```

**07.What are looping control statements? Discuss the different types of loops with suitable example.**

**Ans: –** In programming, sometimes we need to execute the block of code repeatedly while some condition evaluates to true. It can be achieved by using the looping control statement. loop statements can execute the set of instructions in a repeated order for a several number of times. The execution of the set of instructions depends upon a particular condition. In Java, we have three types of looping statement

**(i) while loop: –** while loop is used to run a specific code until a certain condition is met. if we don't know the number of iterations in advance, it is recommended to use a while loop. Unlike for loop, the initialization and increment/decrement doesn't take place inside the loop statement in while loop. It is entry–controlled loop since the condition is checked at the start of the loop. If the condition is true, then the loop body will be executed; otherwise, the statements after the loop will be executed.

**Syntax:**

```
Initialization;
while(condition)
{
        statements;
        increment/decrement;
}
```

**Program: Program to print factorial of a number using while loop.**

```
import java.util.*;
class fact
{
public static void main(String arg[])
{
int n,f=1;
Scanner sc=new Scanner(System.in);
System.out.print("enter number: – ");
n=sc.nextInt();
while(n>0)
{
        f=f*n--;
}
System.out.print("factorial ="+f);
}}
```

**OUTPUT:**

```
B:\Java>javac j19.java
B:\Java>java fact
enter number: – 5
factorial =120
```

**(ii) do while loop: –** do while loop is similar to while loop but the only difference that it checks the condition at the end of the loop after executing the loop statements. If the condition of do while loop is false then the looping statement will execute at least once. It is exit–controlled loop since the condition is checked at the end of loop.

**Syntax:**

```
initialization;
do
{
  Statements;
  increment/decrement;
}while (condition);
```

**Program: Program to print odd number from 1 to 20**

```
import java.util.*;
class odd
{
public static void main(String arg[])
{
int i=1;
do
{
        System.out.print(i+ "\t");
        i=i+2;
}while(i<=20);
} }
```

**OUTPUT:**

```
B:\Java>javac do.java
B:\Java>java odd
1    3    5    7    9    11    13    15    17    19
```

**(iii) for loop:** – for loop provides a concise way of writing the loop structure. In Java, for loop is similar to C and C++. It enables us to initialize the loop variable, check the condition, and increment/decrement in a single line of code. We use the for loop only when we exactly know the number of times, we want to execute the block of code. It is also entry–controlled loop since the condition is checked at the start of the loop. If the condition is true then the body of the loop will execute otherwise statement after loop will executed.

**Syntax:**

```
for(initialization, condition, increment/decrement)
{
        Statements;
}
```

**Program: Program to print table of a number using for loop**

```
import java.util.*;
class table
{
public static void main(String arg[])
{
int n,i;
Scanner sc=new Scanner(System.in);
System.out.print("Enter any number: –");
n=sc.nextInt();
for(i=1;i<=10;i++)
{
System.out.println(n +"X" +i +"=" +(n*i));
}
} }
```

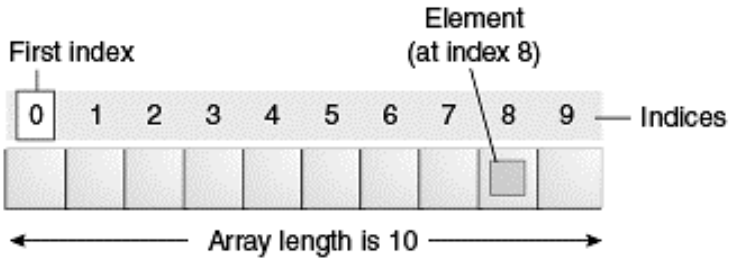
**OUTPUT:**

```
B:\Java>javac tab.java
B:\Java>java table
Enter any number: –5
5X1=5
5X2=10…….
5X10=50
```

**08. Define an array. Discuss its type. Write a Java program to enter ten number in an array prints the number in sorted order.**

**Ans:** – An array is a collection of similar type of elements stored at continuous memory location. It is used to store multiple values in a single variable, instead of declaring separate variables for each value. In Java, array is an object which contains elements of a similar data type. It is a linear data structure where we store similar elements same as C/C++.

We can store only a fixed set of elements in a Java array. Array 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. A Java array variable can also be declared like other variables with [] after the data type and variable name. we can access the element of array by using its index value. In java, the memory is allocated to an array by using the new operator.



There are two types of array:

**One Dimensional Array –** One–dimensional Array is also known as a linear array, in this type of array the elements are stored in a single row.

**Syntax to Declare 1D Array**

dataType[] arrayname; (or)

dataType arrayname[];

arrayname=new datatype[size];

**Example:**

int arr[];

arr=new int [5];

**Multidimensional Array: –** A multidimensional array is an array of arrays. That is, each element of a multidimensional array is an array itself. The Multidimensional Array store the elements in form of row and column.

**Syntax to Declare Multidimensional Array**

dataType [][]arrayname; (or)

dataType arrayname [][];

Arrayname= new datatype[size][size];

**Example:**

int arr[][];

arr=new int [3][3];

it create an array of 3 row and 3 column

**Program:**

```
import java.util.*;
class array
{
public static void
main(String arg[])
{
int arr[],i,j,t;
Scanner sc=new
Scanner(System.in);
arr=new int[10];
System.out.println("enter  10
elements: –");
for(i=0;i<10;i++)
{
System.out.print("enter
element: –");
arr[i]=sc.nextInt();
}
for(j=1;j<10;j++)
{
for(i=0;i<10–j;i++)
{
if(arr[i]>arr[i+1])
{
t=arr[i];
arr[i]=arr[i+1];
arr[i+1]=t;
}
}}
System.out.println("element
of array after sorting are: –");
for(i=0;i<10;i++)
{
System.out.print(arr[i]
+"\t");
}}}
```

**OUTPUT:**

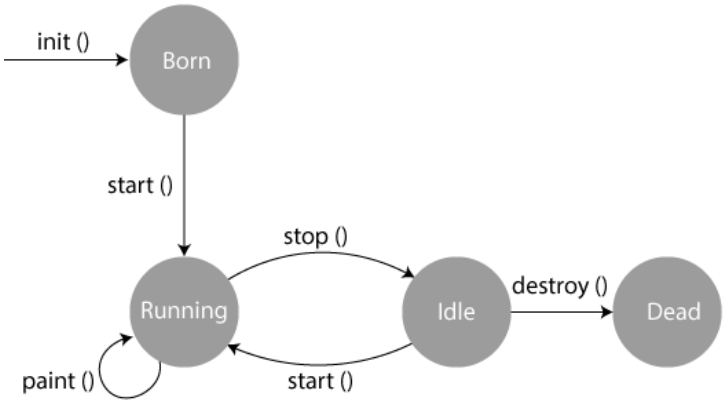
```
B:\Java>javac sort.java
B:\Java>java array
enter  10 elements: –
enter element: –32
enter element: –3
enter element: –22
enter element: –1
enter element: –14
enter element: –78
enter element: –54
enter element: –3
enter element: –54
enter element: –37
element of array after sorting
are: –
1     3     3     14     22
32     37     54     54     78
```

**09. What is an applet. Discuss the life cycle of an applet. Write an applet code to display a message "Welcome to Java programming".**

**Ans: –** Applet is a class or package use for create graphical user interface. An Applet program is a type of Java program that runs in a Web browser or in a Java enabled browser. An applet can be a fully functional Java application because it has the entire Java API at its disposal. Applet support graphics, Windows, event and database components. The applet program executes according to the applet life cycle where applet use various method for each life cycle.

The Applets can be used to make the website more dynamic and entertaining. A main() method is not invoked on an applet, and an applet class will not define main() method. To create an applet, we import java.applet package and class must extends the Applet class.

**Life cycle of an applet : There are five methods of an applet life cycle, and they are:**




**(i) init():** The init( ) method is the first method to be called. In an applet, this is where us should initialize variables. This method is called only once during the runtime of applet.

**(ii) start():** The start() method contains the actual code of 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.

**(iii) paint():** The paint() method belongs to the Graphics class in Java. It is used to draw shapes like circle, square, etc. in the applet. It is also use to draw string on applet screen. It is executed after the start() method and when the browser or applet windows are resized.

**(iv) stop():** The stop() method stops the execution of the applet. The stop () method is invoked whenever the applet is stopped, minimized, or moving from one tab to another in the browser. When we go back to that page, the start() method is invoked again.

**(v) destroy():** The destroy() method destroys the applet after its work is done. It is invoked when the applet window is closed or when the tab containing the webpage is closed. It removes the applet object from memory and is executed only once in the last. The stop( ) method is always called before destroy( ).

**Program:**

**//app1.**Java

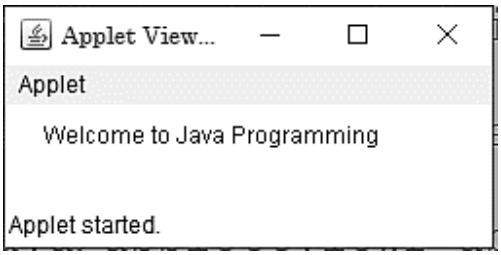
```
import java.applet.*;
import java.awt.*;
public class app1 extends Applet
{
String str;
public void init()
{
str=new String("Welcome to Java Programming");
}
public void paint(Graphics g)
{
g.drawString(str,20,20);
}
}
```

**//app1.html**

```
<html>
<head>
<applet code="app1.class"
height=50
width=250>
</applet>
</head>
</html>
```

**OUTPUT:**

```
B:\Java>javac app1.java
B:\Java>appletviewer app1.html
```




**10.What is vector? Write a Java program to demonstrate the use of some of the methods of vectors class.**

**Ans: –** Vector is like the dynamic array which can grow or shrink its size. Like an array, it contains components that can be accessed using an integer index. In array we can store only the fixed number of element but in vector we can store n–number of elements in it as there is no size limit.

The vector is a part of Java Collection framework and It is found in the java.util package and it implements the List interface, so we can use all the methods of List interface here. Vector is used in that cases where, us don't know the size of the array in advance or us just need one that can change sizes over the lifetime of a program.

**Syntax to create vector:**

Vector<type> vectorname = new Vector<type>(int size);

**Example:**

Vector <Integer> v=new Vector <Integer> (20);

**Basic Methods in Vector Class: –**

**(i) add(element)** – Appends the specified element to the end of this Vector.

**(ii) add(int index, element)** – Inserts the specified element at the specified position in this Vector.

**(iii) addElement(obj)** – Adds the specified component to the end of the vector and increasing its size by one.

**(iv) capacity()** – Returns the current capacity of this vector.

**(v) clear()** – Removes all of the elements from this Vector.

**(vi) size()** – Returns the number of components in this vector.

**(vii) elementAt(int index)** – Returns the component at the specified index.

**(viii) removeElementAt(int index)** – Deletes the component at the specified index.

**(ix) removeElement(Object)** – Removes the first (lowest–indexed) occurrence of the component from this vector.

**(x) removeAllElements()** Removes all components from this vector and sets its size to zero.

**Program:**

```
import java.util.*;
class vec
{
public static void main(String arg[])
{
Vector <Integer> v=new Vector <Integer> (20); //create vector of size 20
int s,i,n,k,r;
Scanner sc=new Scanner(System.in);
System.out.print("enter number of element us want to enter : –");
s=sc.nextInt();
System.out.println("enter " +s+" element");
for(i=0;i<s;i++)
{
    System.out.print("enter element: –");
    n=sc.nextInt();
    v.addElement(n); //add element to vector
}
System.out.println("size of vector="+v.size()); //print size of vecor
System.out.println("capacity of vector= "+v.capacity()); //print capacity of vector
System.out.print("element of vector are: –  " );
```

```
for(i=0;i<size;i++)
{
System.out.print(v.elementAt(i) +"\t"); // print vector element at index i
}
System.out.println();
System.out.print("Enter index number which  us want to remove–");
k=sc.nextInt();
v.removeElementAt(k); //remove specific index number value from vector
System.out.print("Enter value us want to remove–");
r=sc.nextInt();
v.removeElement(r); // remove any value from vector
System.out.print("After removing element of vector are: –  ");
for(i=0;i<v.size();i++)
{
System.out.print(v.elementAt(i) +"\t");
}}}
```

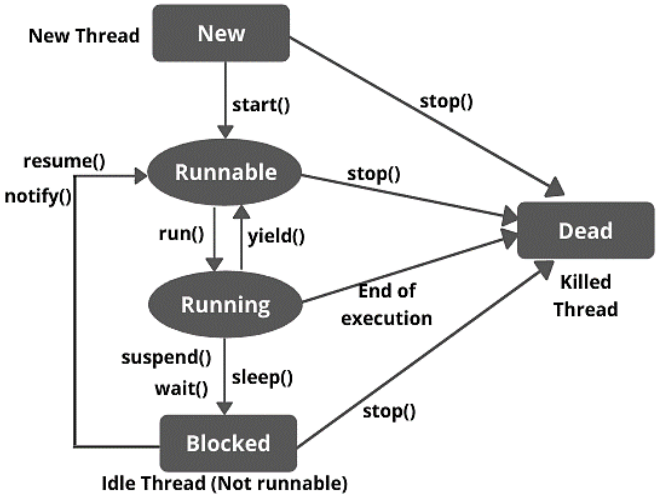
**OUTPUT:**

```
B:\Java>javac vc.java
B:\Java>java vec
enter number of element us want to enter : –
6
enter 6 element
enter element: –1
enter element: –2
enter element: –3
enter element: –4
enter element: –5
enter element: –6
size of vector=6
capacity of vector=20
element of vector are: –  1    2    3    4
5    6
Enter index number which us want to
remove–1
Enter value us want to remove–5
After removing element of vector are: –  1
3    4    6
```

**11. What is multithreading? Discuss the life cycle of a thread in details. Write a Java program for creating demonstration of threads by extending the thread class.**

**Ans: –** Multithreading is an important concept supported by java language which allow to implement multitasking feature.  it is a process of executing multiple threads simultaneously. A thread is a lightweight sub–process, the smallest unit of processing. The thread exexcute according to the thread life-cycle. Multithreading uses a shared memory area. Java has great support for multithreaded applications. Multithreading saves time as us can perform multiple operations together. The threads are independent, so it does not block the user to perform multiple operations at the same time and also, if an exception occurs in a single thread, it does not affect other threads. Java supports multithreading through Thread class. Java Thread allows us to create a lightweight process that executes some tasks. Thread class provides constructors and methods to create and perform operations on a thread. Thread class extends Object class and implements Runnable interface. The thread class is present in the java.lang package which is by default imported in the every program of java.

**Life Cycle of a Thread** There are five states a thread has to go through in its life cycle. This life cycle is controlled by JVM (Java Virtual Machine). These states are:




**(i) New state:** Whenever a new thread is created, it is always in the new state. The thread has not yet started to run when the thread is in this state. When a thread lies in the new state, its code is yet to be run and hasn't started to execute.

**(ii) Runnable state:** Runnable state means a thread is ready for execution. When the start() method is called on a new thread, thread enters into a runnable state.In runnable state, thread is ready for execution and is waiting for availability of the processor (CPU time). It is the duty of the thread scheduler to provide the thread time to run, i.e., moving the thread from runnable to the running state.

**(iii) Running state:** Running means Processor (CPU) has allocated time slot to thread for its execution. When thread scheduler selects a thread from the runnable state for execution, it goes into running state. In running state, processor gives its time to the thread for execution. This is the state where thread performs its actual functions. A thread can come into running state only from runnable state.

**(iv) Blocked state:** A thread is considered to be in the blocked state when it is suspended, sleeping, or waiting for some time in order to satisfy some condition. Thread goes back to the runnable state only when another thread signals the waiting thread to continue executing or time interval expires.

**(v) Dead state:** A thread moves into dead state when the code of the thread has been entirely executed by the program or due to some unusual event, like segmentation fault or an unhandled exception. It is the last state of every thread.

**Program:**

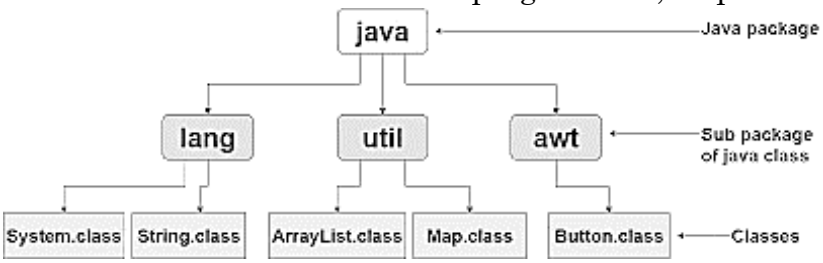
```
class mythread extends Thread
{
public void run()
{
for(int i=1;i<=5;i++)
{
System.out.print(i +"\t");
}
}
}
class th
{
public static void main(String arg[])
{
mythread t1,t2,t3; //declare object
t1= new mythread(); // creating object
t2= new mythread();
t3= new mythread();
t1.start(); //calling run() method
t2.start();
t3.start();
}
}
```

**OUTPUT:**

```
B:\Java>javac mt.java
B:\Java>java th
1   1   1   2   3   4   5
2   3   2   4   3   5   4   5
```

**12.Define package. What are the advantages of using package? Write down the steps to create a package.**

**Ans: –** A java package is a group of similar types of classes, interfaces and sub–packages. In other word we can say that, Package in Java is a mechanism to encapsulate a group of similar classes, sub packages and interfaces. A package inside another package is known as sub package. The package is used to prevent the naming conflicts, to control access and to import the classes and method in the program. To, import a class or a package in the program, we use the "import" keyword, we can either import a single class along with its methods and attributes, or a whole package that contain all the classes that belong to the specified package.




**Syntax:**

import package.Class; // Import a single class of package

import package.*; // Import the whole package

**Example:**

import java.util.Scanner; // Import Scanner class only

import java.util.*; // Import all Classes of util package

Package in java are of two types: built–in package and user–defined package

**(i) Built– in Packages –** These packages consist of a large number of classes which are a part of Java API. It has prewritten classes, that are free to use, included in the Java Development Environment. Some of the commonly used built–in packages are:

**(1) java.lang:** this package Contains language support classes which define primitive data type, math operation etc. This package is automatically imported.

**(2) java.io:** Contains classed for supporting input / output operations.

**(3) java.util:** Contains utility classes which implement data structures like Linked List, Dictionary and support for Date / Time operations.

**(4) java.applet:** Contains classes for creating Applets and implement various methods of applet.

**(5) java.awt:** Contain classes for implementing the components for graphical user interfaces (like button, menus etc).

**(ii) User– defined packages –** Java also allows us to create packages as per our need. These packages are called user–defined packages in other word we can say that, the user–defined packages are the packages that are defined by the user.

**Advantage of Java Package**

(i) Java package is used to categorize the similar classes and interfaces so that they can be easily maintained.

(ii) Java package prevent from the naming conflicts.

(iii) It is also easier to locate the related classes.

(iv) Java package provides access protection.

(v) Using packages, it is easier to provide access control

(vi) Programmers can define their own packages to containg a group of classes/interfaces, etc.

**Steps to create a package.**

To create a package, follow the steps given below:

**Step 1:** Choose a package name according to the naming convention. Ex – mypackage

**Step 2:** Include the package command as the first line of code Followed by the package name Ex – package mypackage;

**Step 3:** The classes, interfaces, methods etc. which we want to include in the package can be made inside the package. Example –

```
package mypackage;
class welcome // Class which belong to the above created package
{
   public static void main(String[] args)
   {
     System.out.println("This Is Java");
   }
}
```

**Step 4:** save the code with .java extension. ex –mypackage.java

**Step 5:** Now, to create a package, use the command – ***javac –d . filename.java***

**Ex– javac –d . mypackage.java** This command creates a package mypackage.

**Step 6:** Now we can use this package in us program.

**13.What is method overloading? Write a Java program to calculate area perimeter of circle, square and rectangle using method overloading.**

**Ans:** –The method overloading is also known as function overloading. It is a type of compile time polymorphism. The method overloading is defined as a process of having two or more function with same name but different in parameters. the function is redefined with the same name by using either different number of argument or different datatype of argument. Whenever these functions are called the compiler automatically determine and call the most appropriate function which match the parameter type. The method overloading is used to improve the reliability of the code and It helped the programmer because they do not need to remember various function name. The function overloading is generally used when similar operation is performed on different types of argument or different number of arguments.

In method overloading the method can be redefined in two ways–

**(i) By using different datatype of argument** Example –

```
public void area(int a)
{
return(a*a);
}
```

```
public void area(float r)
{
return(3.14f*r*r);
}
```

**(ii) By using different number of argument** Example –

```
public void sum(int a, int b)
{
return(a+b);
}
```

```
public void sum(int a, int b, int c)
{
return(a+b+c);
}
```

**Program:**

```
import java.util.*;
class over
{
public void area(float x)//fun1
{
float ar,cr;
ar=3.14f*x*x;
cr=2.0f*3.14f*x;
System.out.println("Area of circle="+ar);
System.out.println("Circumference of
circle="+cr);
}
public void area(int x)//fun2
{
int ar,p;
ar=x*x;
p=4*x;
System.out.println("Area of square="+ar);
System.out.println("Perimeter of
square="+p);
}
public void area(int x, int y) //fun3
{
int ar,p;
ar=x*y;
p=2*(x+y);
System.out.println("Area of  Rectangle="+ar);
System.out.println("Perimeter of
Rectangle="+p);
}
public static void main(String arg[])
{
Scanner sc=new Scanner(System.in);
int s,l,b;
float r;
over o=new over();
System.out.print("Enter radius of circle: –");
r=sc.nextFloat();
System.out.print("Enter side of square: –");
s=sc.nextInt();
System.out.print("Enter length and breadth
of  rectangle: –");
l=sc.nextInt();
b=sc.nextInt();
o.area(r); //calling fun1
o.area(s); //calling fun2
o.area(l,b); //calling fun3
} }
```

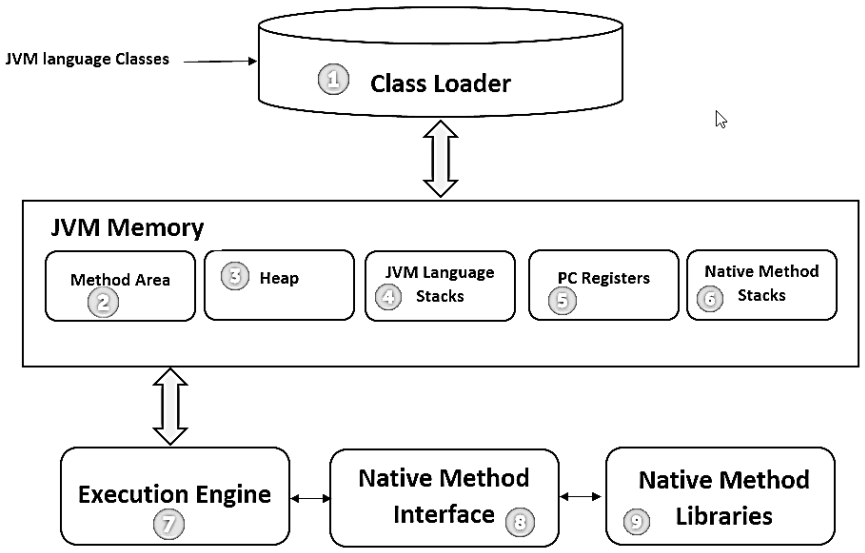
**OUTPUT:**

```
B:\Java>javac metover.java
B:\Java>java over
Enter radius of circle: –2
Enter side of square: –3
Enter length and breadth of  rectangle: –4
5
Area of circle=12.56
Circumference of circle=12.56
Area of square=9
Perimeter of square=12
Area of  Rectangle=20
Perimeter of  Rectangle=18
```

**14.What is Java Virtual Machine (JVM). What are different components of JVM?**

**Ans: –** JVM stands for Java virtual machine. JVM 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). It provides the specification for runtime environment in which Java bytecode can be executed. Mostly in other Programming Languages, compiler produce code for a particular system but Java compiler produce Bytecode for a Java Virtual Machine. When we compile a Java program, then bytecode is generated. Bytecode is the source code that can be used to run on any platform. Bytecode is an intermediary language between Java source and the host system. The JVM makes it possible for Java–based software programs to follow the "write once, run anywhere" approach. Us can write Java code on one machine, and run it on any other machine using the JVM. JVMs are available for many hardware and software platforms.

**Components of JVM:** – There are mainly 9 components in JVM




**(1) ClassLoader** – The class loader is a subsystem used for loading class files. Whenever we run the Java program, it is loaded first by the classloader. It performs three major functions– Loading, Linking, and Initialization.

**(2) Method Area** – JVM Method Area stores class structures like metadata, the constant runtime pool, code for methods etc. There is only one method area per JVM, and it is a shared resource.

**(3) Heap Area** – All the Objects, their related instance variables, and arrays are stored in the heap area. There is also one Heap Area per JVM. It is also a shared resource.

**(4) JVM language Stacks** – Java language Stacks store local variables, and it's partial results. Each thread has its own JVM stack, created simultaneously as the thread is created. A new frame is created whenever a method is invoked, and it is deleted when method invocation process is complete.

**(5) PC Registers** – PC register store the address of the Java virtual machine instruction which is currently executing. In Java, each thread has its separate PC register.

**(6) Native Method Stacks** – Native method stacks hold the instruction of native code depends on the native library. For every thread, a separate native stack is created. It stores native method information.

**(7) Execution Engine** – Execution engine executes the ".class" (bytecode). It reads the byte–code line by line, uses data and information present in various memory area and executes instructions. It has three parts: A virtual processor, Interpreter, Just–In–Time(JIT) compiler.

**(8) Native interface** – it is a framework which provides an interface to communicate with another application written in another language like C, C++, Assembly etc. Java uses this framework to send output to the Console or interact with OS libraries.

**(9) Native Method Libraries** – Native Method Libraries are libraries that are written in other programming languages, such as C, C++ which are needed by the Execution Engine.

**15.What is method overriding? Write a Java program to demonstrate the concept of method overriding.**

**Ans:** – In any object–oriented programming language, Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super–classes or parent classes. Method overriding is one of the ways by which java achieve Run Time Polymorphism. For method Overriding, The method must have the same name and same parameter as in the parent class and there must be an inheritance. i.e., the super class must inherit the child class. We cannot override the method declared as final and static. The version of a method that is executed will be determined by the object that is used to invoke it. If an object of a parent class is used to invoke the method, then the method of the parent class will be executed, but if an object of the subclass is used to invoke the method, then the method of the child class will be executed.

**Condition for Java Method Overriding**

(i) The method must have the same name as in the parent class

(ii) The method must have the same parameter as in the parent class.

(iii)There must be an inheritance. i.e., the super class must inherit the child class.

(iv) We cannot override the method declared as final and static.

**Program:**

```
import java.util.*;
class Parent //Base Class
{
void show()
{
System.out.println("Parent class");
}}
class Child extends Parent // child class
{
void show()// This method overrides show() of Parent
{
System.out.println("Child class");
}}
class metov
{
public static void main(String[] args)
{
Parent obj1 = new Parent(); //obj. of parent class
obj1.show(); //calling of parent class show method
Parent obj2 = new Child(); //object of child class
obj2.show(); //calling of child class show method
}}
```

**OUTPUT:**

```
B:\Java>javac mto.java
B:\Java>java metov
Parent class
Child class
```

**16. Write Java Program for the following: –**

**(i) Write an applet code to demonstrate the use of various methods of graphics class.**

**//detail.**Java

```
import java.applet.Applet;
import java.awt.*;
public class graph extends Applet
{
public void paint(Graphics g)
{
g.setColor(Color.red);
g.drawString("Welcome",20, 20);
g.setColor(Color.blue);
g.drawLine(20,40,80,40);
g.drawRect(20,60,30,30);
g.fillRect(60,60,30,30);
g.drawOval(20,100,30,30);
g.fillOval(60,100,30,30);
g.drawArc(20,140,30,30,30,180);
g.fillArc(60,140,30,30,30,180);
}
}
```

**//detail.html**

```
<html>
<head>
<applet code="graph.class"
height=200
width=100>
</applet>
</head>
</html>
```

**OUTPUT:**

```
B:\Java>javac graph.java
B:\Java>appletviewer graph.html
```

**(ii) Write an applet code to demonstrate passing of parameters to an applet.**

**//param.java**

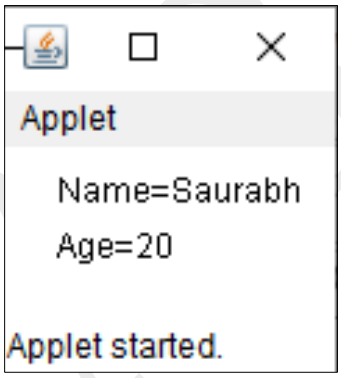
```
import java.applet.*;
import java.awt.*;
public class param extends Applet
{
String name,age;
public void init()
{
name=getParameter("name");
age=getParameter("age");
}
public void paint (Graphics g)
{
g.drawString("Name="+name,20,20);
g.drawString("Age="+age,20,40);
} }
```

**//param.html**

```
<html>
<head>
<applet code="param.class"
height=60 width=120>
<param name="name" value="Saurabh">
<param name="age" value="20">
</applet>
</head>
</html>
```

**OUTPUT:**

```
B:\Java>javac param.java
B:\Java>appletviewer param.html
```




**(iii) Write an applet code to display name, class, roll no and address.**

**//detail.java**

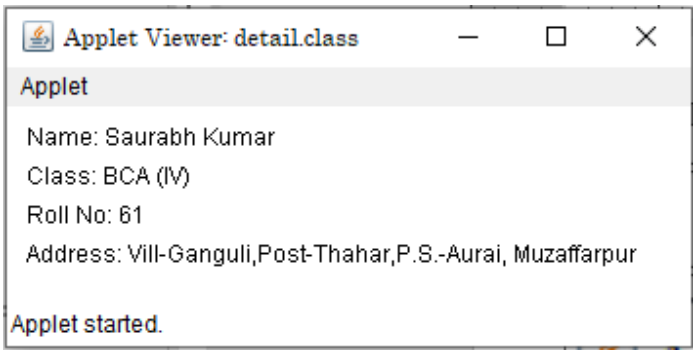
```
import java.applet.*;
import java.awt.*;
public class detail extends Applet
{
String str1,str2,str3,str4;
public void init()
{
str1=new String("Name: Saurabh Kumar");
str2=new String("Class: BCA (IV)");
str3=new String("Roll No: 61");
str4=new String("Address: Vill–
Ganguli,Post–Thahar,P.S.–Aurai,
Muzaffarpur");
}
public void paint(Graphics g) {
g.drawString(str1,10,20);
g.drawString(str2,10,40);
g.drawString(str3,10,60);
g.drawString(str4,10,80);
} }
```

**//detail.html**

```
<html>
<head>
<applet code="detail.class"
height=100
width=350>
</applet>
</head>
</html>
```

**OUTPUT:**

```
B:\Java>javac detail.java
B:\Java>appletviewer detail.html
```



**(iv) Write a Java program to add/subtract/multiply two matrices and print the result matrix.**

```
import java.util.*;
class matrix
{
public static void main(String arg[])
{
int m1[][],m2[][],m3[][],row,col,k;
Scanner sc=new Scanner(System.in);
m1=new int[3][3];
m2=new int[3][3];
m3=new int[3][3];
System.out.println("Enter 1st matrix of 9 element: –");
for(row=0;row<3;row++)
{
for(col=0;col<3;col++)
{
System.out.print("enter " +row+col +" index element: –");
m1[row][col]=sc.nextInt();
} }
System.out.println();
System.out.println("Enter 2nd matrix of 9 element: –");
for(row=0;row<3;row++)
{
for(col=0;col<3;col++)
{
System.out.print("enter " +row+col +" index element: –");
m2[row][col]=sc.nextInt();
} }
```

**For sum and subtract**

```
for(row=0;row<3;row++)
{
for(col=0;col<3;col++)
{
m3[row][col]=m1[row][col]+ / - m2[row][col];
} }
```

**For multiply**

```
for(row=0;row<3;row++)
{
for(col=0;col<3;col++)
{
m3[row][col]=0;
for(k=0;k<3;k++)
{
m3[row][col]=m3[row][col]+m1[row][k]*m2[k][col];
} } }
```

```
System.out.println("Result of both matrix are as: –");
for(row=0;row<3;row++)
{
for(col=0;col<3;col++)
{
System.out.print(m3[row][col]+"\t");
}
System.out.println();
} } }
```

**OUTPUT:**

```
B:\Java>javac j37.java
B:\Java>java matrix
Enter 1st matrix of 9 element: –
enter 00 index element: –1
enter 01 index element: –2
enter 02 index element: –3
enter 10 index element: –4
enter 11 index element: –5
enter 12 index element: –6
enter 20 index element: –7
enter 21 index element: –8
enter 22 index element: –9
```

Enter 2nd matrix of 9 element: –
enter 00 index element: –9
enter 01 index element: –8
enter 02 index element: –7
enter 10 index element: –6
enter 11 index element: –5
enter 12 index element: –4
enter 20 index element: –3
enter 21 index element: –2
enter 22 index element: –1
result of both matrix are as: –

**for multiply**

30 24 18
84 69 54
138 114 90

**For sum**

10 10 10
10 10 10
10 10 10

**For subtract**

–8 –6 –4
–2 0 2
4 6 8

**(v) Write a Java program to print all prime numbers from 0 to 100**

```
import java.util.*;
class prime
{
public static void main
(String arg[])
{
int j,i,t;
for(i=2;i<=100;i++)
{
t=1;
for(j=2;j<i;j++)
{
if(i%j==0)
{
t=0;
break;
} }
if(t==1)
System.out.print(i +"\t");
} } }
```

**OUTPUT:** B:\Java>javac pr.java

B:\Java>java prime

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53
59 61 67 71 73 79 83 89 97

**17. Write the difference between the following: –**

**(i) core Java and advanced Java**
**(ii) C++ and Java programming language.**
**(iii) local and remote applet**
**(iv) "for loop" and "do while loop".**

**(i) The difference between core Java and advanced Java are as:**

| Core Java | Advanced Java |
|---|---|
| (i) Core Java covers the basic topics and concepts of the Java programming language. | (i) Advance Java covers the advanced topics and concepts of the Java programming language. |
| (ii) Core Java is used for developing computing or desktop applications. | (ii) Advance Java is used for developing enterprise applications. |
| (iii) The core Java is the first step, to begin with, Java. | (iii) It is the next step after completing the Core Java. |
| (iv) It comes under Java SE. | (iv) It comes under Java EE or J2EE. |
| (v) It covers core topics such as OOPs, inheritance, exception handling, etc. | (v) It covers advanced topics such as JDBC, servlets, JSP, web services etc. |
| (vi) Core Java is based on single–tier architecture. | (vi) Advance Java is based on two–tier architecture. |

**(ii) The difference between the C++ and Java programming language are as:**

| C++ programming language | Java programming language |
|---|---|
| (i) C++ is both a procedural and an object–oriented programming language. | (i) Java is only an object–oriented programming language. |
| (ii) C++ is mainly used for system programming. | (ii) Mainly used for application programming. |
| (iii) C++ program save with .cpp extension | (iii) Java program save with .java extension |
| (iv) C++ supports the goto statement | (iv) Java doesn't support the goto statement. |
| (v) C++ supports pointer and operator overloading | (v) Java doesn't support pointer and operator overloading |
| (vi) C++ uses compiler only. | (vi) Java uses both compiler and interpreter. |
| (vii) C++ supports structures and unions. | (vii) it doesn't support structures and unions. |
| (viii) C++ is platform–dependent. | (viii) Java is platform–independent. |
| (ix) C++ supports both Pass by Value and pass by reference. | (ix) Java supports only the Pass by Value technique. |
| (x) It supports both single and multiple Inheritance. | (x) It supports only single inheritance. Multiple inheritances are achieved partially using interfaces. |

**(iii) The difference between local applet and remote applet are as:**

| Local Applet | Remote Applet |
|---|---|
| (i) An applet developed locally and stored in a local system that is known as a Local Applet. | (i) A remote applet is that which is developed by someone else and stored on a remote computer connected to the Internet. |
| (ii) There is no need to define the Applet's URL in Local Applet. | (ii) We need to define the Applet's URL in Remote Applet. |
| (iii) Local Applet is available on our computer. | (iii) Remote Applet is not available on our computer. |
| (iv) In order to use it or access it, we don't need Internet Connection. | (iv) In order to use it or access it on our computer, we need an Internet Connection. |
| (v) It is written on our own and then embedded into the web pages. | (v) It was written by another developer. |
| (vi) We don't need to download it. | (vi) It is available on a remote computer, so we need to download it to our system. |

**(iv) The difference between "for loop" and "do while loop" are as:**

**For loop**

(i) for loop is entry–controlled loop.
(ii) It might be that statement(s) gets executed zero times.
(iii) Condition is checked before the statement(s) is executed.
(iv) For the single statement, bracket is not compulsory.
(v) **Syntax:**

```
for (initialization; condition; incr/decr)
{
  Statement;
  Statement;
}
```

**(vi) Example: ques no 7**

**Do while loop**

(i) do–while is exit controlled loop.
(ii) Statement(s) is executed at least once.
(iii) Condition is checked after the statement(s) is executed.
(iv) Brackets are always compulsory.
(v) **Syntax:**

```
Initialization;
do
{
  Statement;
  incr/decr;
} while(condition);
```

**(vi) Example: ques no 7**

**18. Write short note on the following: –**

**(i) Abstract class** – A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non–abstract methods. The Abstract method is a type of method which can only be used in an abstract class, and it does not have a body. The body is provided by the subclass Abstract class is a restricted class that cannot be used to create objects. To access the member of an abstract class, it must be inheriting the other class and the object of the inherited class can access the member of abstract class. An abstract class can have constructors, static methods and final methods.

**Syntax of abstract class**

```
Public abstract class classname
{
Data member;
Abstract methods
Non–abstract methods;
}
```

**Syntax of abstract method**

```
Public abstract <returntype> mth.name();
```

**Example:–**

```
Public abstract class abc
{
void run()
{
System.out.println("welcome");
} }
```

**Example:–**

```
Public abstract void print1();
```

**Program:**

```
abstract class base // Abstract class
{
  public abstract void show1(); // Abstract
method
 public void show2() // Regular method
 {
  System.out.println("Regular method");
 }
}
class child extends base // Subclass
{
 public void show1() // The body of show1()
 {
  System.out.println("Abstract method ");
 }
}
class abst
{
 public static void main(String args[])
 {
  child c= new child(); // Create object
  c.show1(); //calling Abstract method
  c.show2(); //calling Regular method
 }
}
```

**OUTPUT:**

```
B:\Java>javac abs.java
B:\Java>java abst
Abstract method
Regular method
```

**(ii) String** – Generally, String is a sequence of characters. But in Java, string is an object that represents a sequence of characters. The java.lang.String class is used to create a string object. In Java, we can create strings like primitive types and also as an objects, we can create strings as a object using the new keyword. An array of characters works same as Java string. For example: char[] ch={'j','a','v','a'}; is same as: String s="java"; or String str=new String("java"); Java String class provides a lot of methods to perform operations on strings such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), indexOf(), substring() etc.

**Syntax to create string:**

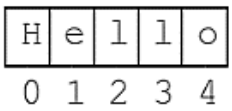
String <stringname> = "<sequence_of_string>";

String <stringname> = new String("<sequence_of_string>");

**Example:**

String str = "Hello";

String str=new String("Hello");




**Program:**

```
class str
{
public static void main(String args[])
{
// create strings
String first = "java";
String second = new String("python");
// print strings
System.out.println(first);
System.out.println(second);
//string methods
System.out.println(second.length());
System.out.println(first.toUpperCase());
System.out.println(first.charAt(3));
}
}
```

**OUTPUT:**

```
B:\Java>javac st.java
B:\Java>java str
java
python
6
JAVA
a
```

**(iii) Final** – The final keyword in java is used to restrict the user. In Java, the final keyword is used to denote constants. It can be used with variables, methods, and classes.

**(i) Variable** – If we make any variable as final by using final keyword before the datatype at the time of declaration, we cannot change the value of final variable (It will be constant). we must initialize a final variable at the time of declaration.

**Syntax:** final datatype <varname>= value **Example:** final int x = 5;

**(ii) Method** – If we make any method as final by using final keyword before return type then the final method cannot be overridden by the child class.

**Syntax:**

```
final <returntype> <name>()
{
Statement;
}
```

**Example:**

```
final void run()
{
System.out.println("welcome");
}
```

**(iii) class** – If we make any class as final by using final keyword before class keyword then the final class does not perform inheritance i.e. the child class cannot inherited from the final base class.

**Syntax:**

```
final class <classname>
{
Data member;
Member function;
}
```

**Example:**

```
Final class abc
{
void run() {
System.out.println("welcome");
} }
```

**Program:**

```
final class finkey  //final class
{
        final int i=10;  //final variable
        final void show()  //final method
        {
                System.out.print("value="+i);
        }
        public static void main(String arg[])
        {
                finkey f=new finkey();
                f.show();
        }
}
```

**OUTPUT:**

```
B:\Java>javac fin.java
B:\Java>java fi
value=10
```

**Prepared by Saurabh kumar**

**BCA – 4th Semester**

**L. S. College, Muzaffarpur**

**Mob: - 7488944009**



# THE END