GLOBAL INSTITUTE OF TECHNOLOGY

(Approved by AICTE and Affiliated to RTU, Kota)

LABORATORY MANUAL

(2019-2020)

JAVA LAB

II Year & IV Semester

Computer Science & Engineering

TABLE OF CONTENTS

Syllabus

Credit: 1 **Max. Marks: 50(IA:30, ETE:20)**
0L+0T+2P

List of Experiment:

1. Develop an in depth understanding of programming in Java: data types, variables, operators, operator precedence, Decision and control statements, arrays, switch statement, Iteration Statements, Jump Statements, Using break, Using continue, return.
2. Write Object Oriented programs in Java: Objects, Classes constructors, returning and passing objects as parameter, Inheritance, Access Control, Using super, final with inheritance Overloading and overriding methods, Abstract classes, Extended classes.
3. Develop understanding to developing packages & Interfaces in Java: Package, concept of CLASSPATH, access modifiers, importing package, Defining and implementing interfaces.
4. Develop understanding to developing Strings and exception handling: String constructors, special string operations, character extraction, searching and comparing strings, string Buffer class. Exception handling fundamentals, Exception types, uncaught exceptions, try, catch and multiple catch statements. Usage of throw, throws and finally.
5. Develop applications involving file handling: I/O streams, File I/O.
6. Develop applications involving concurrency: Processes and Threads, Thread Objects, Defining and Starting a Thread, Pausing Execution with Sleep, Interrupts, Joins, and Synchronization.

Indicative List of exercises:

7. Programs to demonstrate basic concepts e.g. operators, classes, constructors, control & iteration statements, recursion etc. such as complex arithmetic, matrix arithmetic, tower of Hanoi problem etc.
8. Development of programs/projects to demonstrate concepts like inheritance, exception handling, packages, interfaces etc. such as application for electricity department, library management, ticket reservation system, payroll system etc.
9. Development of a project to demonstrate various file handling concepts.
10. Develop applications involving Applet: Applet Fundamentals, using paint method and drawing polygons. It is expected that each laboratory assignments to given to the students with an aim to In order to achieve the above objectives.

MARKS SCHEME

RTU Marks Scheme

Maximum Marks Allocation		
Sessional	End-Term	Total
30	20	50

LAB PLAN			
Java Programming Lab(4CS4-25)			
S.No.	Contents	Experiments	**Lab Turn**
1	Simple Programs without classes and objects, methods	• Byte code generation its meaning • Role of JDK, JRE, JVM • Data types • Various operators	Turn-01
2	Program based on the concepts of classes and objects, constructor, parameterized constructor	• Object instantiation • Constructors • Methods (defining & Calling) • Types of constructor • Parameter passing to methods	Turn-02
3	Method overloading, constructor overloading	• Method overloading • Constructor overloading	Turn-03
4	Single level & Multi level inheritance	• Single level Inheritance • Multiple inheritance • Super • Order of Constructor calling • Method overriding	Turn-04
5	Abstract Classes, Interface	• Final keyword • Abstract classes • Interfaces	Turn-05
6	Array	• Simple programs using array	Turn-06
7	Exception handling	• Exception handling through- • Try • Catch • Throw • Throws • Finally	Turn-07
8	Package	• Making own package	Turn-08

9	Multithreading	• Simple programs of multithreading	Turn-09
10	Applet	• Applets • Drawing various shapes through applets • String scrolling in applet	Turn-10
11	I/O& File Handling	• Input from user • Creation of file • Reading data from file	Turn-11

Lab Objectives:

4CS4-25.1: To be able to develop an in depth understanding of programming in Java: data types, variables, operators, operator precedence, Decision and control statements, arrays, switch statement, Iteration Statements, Jump Statements, Using break, Using continue, return.

4CS4-25.2: To be able to write Object Oriented programs in Java: Objects, Classes constructors, returning and passing objects as parameter, Inheritance, Access Control, Using super, final with inheritance Overloading and overriding methods, Abstract classes, Extended classes.

4CS4-25.3: To be able to develop understanding to developing packages & Interfaces in Java: Package,concept of CLASSPATH, access modifiers, importing package, Defining and implementing interfaces.

4CS4-25.4: To be able to develop understanding to developing Strings and exception handling: String constructors, special string operations, character extraction, searching and comparing strings, string Buffer class. Exception handling fundamentals, Exception types, uncaught exceptions, try, catch and multiple catch statements. Usage of throw, throws and finally.

4CS4-25.5: To be able to develop applications involving file handling: I/O streams, File I/O. To develop applications involving concurrency: Processes and Threads, Thread Objects, Defining and Starting a Thread, Pausing Execution with Sleep, Interrupts, Joins, and Synchronization.

Experiments

1. Java Basic.

Write a Program to print the text “Welcome to World of Java”. Save it with name Welcome.java in your folder.

```
Class Welcome
{
public static void main (String args[])
{
System.out.println (“welcome to world of Java”);
}
}
```

Write a Program to print the area of triangle. Save it with name Area.java in your folder.

```
class Area
{
public static void main(String args[])
{
int height =10, base=6;
float area=0.5F*base* height;
System.out.println(“area of triangle = ”+area);
}
}
```

Write a java Program to check the number is Prime or not.

```
Importjava.util.Scanner;

class Prime
  {
 public static void main(String arr[])
  {
 int c;
 Scanner in=new Scanner(System.in);
 System.out.println("Enter the number to be tested for prime ");
 int n=in.nextInt();
 for ( c = 2 ; c <= n - 1 ; c++ )
  {
if ( n%c == 0 )
  {
System.out.println(n+">>>>> not prime");
 break;
 }
  }
if ( c == n )
System.out.println(n+   ">>>>Number is prime.");

  }
```

```
}
```

Write a java Program to generate a Ladder of number.

```
importjava.util.Scanner;
class Ladder
{
 public static void main(String arr[])
 {
Scanner in=new Scanner(System.in);
System.out.println("Enter the number of rows");
int a=in.nextInt();
for(int i=1;i<=a;i++)
      {
for(int j=1;j<=i;j++)
System.out.print(j);
for(int k=i-1;k>=1;k--)
System.out.print(k);
System.out.print("\n");
      }
}
}
```

Viva Questions

Q1. Explain JDK, JRE and JVM?

JDK vs JRE vs JVM		
JDK	JRE	JVM
It stands for Java Development Kit.	It stands for Java Runtime Environment.	It stands for Java Virtual Machine.
It is the tool necessary to compile, document and package Java programs.	JRE refers to a runtime environment in which Java bytecode can be executed.	It is an abstract machine. It is a specification that provides a run-time environment in which Java bytecode can be executed.
It contains JRE + development tools.	It's an implementation of the JVM which physically exists.	JVM follows three notations: Specification, Implementation, and Runtime Instance.

Q2. Explain public static void main(String args[]) in Java.

main() in Java is the entry point for any Java program. It is always written as public static void main(String[] args).

- public: Public is an access modifier, which is used to specify who can access this method. Public means that this Method will be accessible by any Class.

- static: It is a keyword in java which identifies it is class-based. main() is made static in Java so that it can be accessed without creating the instance of a Class. In case, main is not made static then the compiler will throw an error as main() is called by the JVM before any objects are made and only static methods can be directly invoked via the class.
- void: It is the return type of the method. Void defines the method which will not return any value.
- main: It is the name of the method which is searched by JVM as a starting point for an application with a particular signature only. It is the method where the main execution occurs.
- String args[]: It is the parameter passed to the main method.

Q3. Why Java is platform independent?

Java is called platform independent because of its byte codes which can run on any system irrespective of its underlying operating system.

Q.4 why is main method declared as static?

AnsThe main method is static because it keeps things simpler. Since the main method is static JVM (Java virtual Machine) can call it without creating any instance of a class which contains the main method. The main() method must be declared public, static, and void. It must accept a single argument that is an array of strings. This method can be declared as either:

public static void main(String[] args)

Q.5 Is JDK required on each machine to run a java program?

Ans.No, JDK (Java Development Kit) isn't required on each machine to run a Java program. Only JRE is required, it is an implementation of the Java Virtual machine (JVM), which actually executes Java programs. JDK is development Kit of Java and is required for development only. It is a bundle of software components that is used to develop Java based applications.

2. Classes and Objects, Constructors

Write a program to create a class Student with data ‘name, city and age’ along with method printData to display the data. Create the two objects s1 ,s2 to declare and access the values.

```
class Student
{
String name, city;
int age;
staticint m;
voidprintData()
{
System.out.println("Student name = "+name);
System.out.println("Student city = "+city);
System.out.println("Student age = "+age);
}
}
classStest
{
public static void main(String args[])
{
```

```
Student s1=new Student();
Student s2=new Student();
s1.name="Amit";
s1.city="Dehradun";
s1.age=22;
s2.name="Kapil";
s2.city="Delhi";
s2.age=23;
s2.printData();
s1.printData();
s1.m=20;
s2.m=22;
Student.m=27;
System.out.println("s1.m = "+s1.m);
System.out.println("s2.m = "+s2.m);
System.out.println("Student.m ="+Student.m);
}
}
```

Write a program to create a class Student2 along with two method getData(),printData() to get the value through argument and display the data in printData. Create the two objects s1 ,s2 to declare and access the values from class STtest.

```
class Student2
{
private String name, city;
privateint age;
public void getData(String x, Stringy, int t)
{
name=x;
city=y;
age=t;
}
public void printData()
{
System.out.println("Student name ="+name);
System.out.println("Student city ="+city);
System.out.println("Student age ="+age);
}
}
classSTtest
{
public static void main(String args[])
{
Student2 s1=new Student2();
Student2 s2=new Student2();
s2.getData("Kapil","Delhi",23);
s2.printData();
s1.getData("Amit","Dehradun",22);
s1.printData();
}
}
```

WAP using parameterized constructor with two parameters id and name. While creating the objects obj1 and obj2 passed two arguments so that this constructor gets invoked after creation of obj1 and obj2.

```
class Employee {

intempId;
   String empName;

   //parameterized constructor with two parameters
Employee(int id, String name){
this.empId = id;
this.empName = name;
   }
void info(){
System.out.println("Id: "+empId+" Name: "+empName);
   }

public static void main(String args[]){
            Employee obj1 = new Employee(10245,"Chaitanya");
            Employee obj2 = new Employee(92232,"Negan");
            obj1.info();
            obj2.info();
   }
}
```

Viva Questions

Q.1 What is a Constructor?

Ans. Constructors are used to initialize the object's state. Like methods, a constructor also contains collection of statements(i.e. instructions) that are executed at time of Object creation.

Q.2 What are the differences between C++ and Java?

Ans.The differences between C++ and Java are given in the following table.

Comparison Index	C++	Java
Platform-independent	C++ is platform-dependent.	Java is platform-independent.
Mainly used for	C++ is mainly used for system programming.	Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications.
Design Goal	C++ was designed for systems and applications programming. It was an extension of C	Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed with a goal of being easy to use and accessible to a broader audience.

	programming language.	
Goto	C++ supports the goto statement.	Java doesn't support the goto statement.
Multiple inheritance	C++ supports multiple inheritance.	Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java.
Operator Overloading	C++ supports operator overloading.	Java doesn't support operator overloading.
Pointers	C++ supports pointers. You can write pointer program in C++.	Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in java.
Compiler and Interpreter	C++ uses compiler only. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform dependent.	Java uses compiler and interpreter both. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform independent.
Call by Value and Call by reference	C++ supports both call by value and call by reference.	Java supports call by value only. There is no call by reference in java.
Structure and Union	C++ supports structures and unions.	Java doesn't support structures and unions.
Thread Support	C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support.	Java has built-in thread support.
Virtual Keyword	C++ supports virtual keyword so that we can decide whether or not override a function.	Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default.
unsigned right shift >>>	C++ doesn't support >>> operator.	Java supports unsigned right shift >>> operator that fills zero at the top for the negative numbers. For positive numbers, it works same like >> operator.
Inheritance Tree	C++ creates a new inheritance tree always.	Java uses a single inheritance tree always because all classes are the child of Object class in java. The object class is the root of the inheritance tree in java.
Hardware	C++ is nearer to	Java is not so interactive with hardware.

	hardware.	
Object-oriented	C++ is an object-oriented language. However, in C language, single root hierarchy is not possible.	Java is also an object-oriented language. However, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object.

Q3. Can we have a class with no Constructor in it ? What will happen during object creation ?

Ans. Yes, we can have a class with no constructor, When the compiler encounters a class with no constructor then it will automatically create a default constructor for you.

Q4.What is No-arg constructor?
Ans. Constructor without arguments is called no-arg constructor. Default constructor in java is always a no-arg constructor.

Q.5 If I don't provide any arguments on the command line, then what will the value stored in the String array passed into the main() method, empty or NULL?

Ans.It is empty, but not null.

3. Method & Constructor Overloading

Write a program in JAVA to demonstrate the method and constructor overloading.

```
class Cs
{
intp,q;
public Cs()
{}
public Cs(int x, int y)
{
p=x;
q=y;
}
publicint add(int i, int j)
{
return (i+j);
}
publicint add(int i, int j, int k)
{
return (i+j+k);
}
public float add(float f1, float f2)
{
return (f1+f2);
```

```
}
public void printData()
{
System.out.print("p = "+p);
System.out.println(" q = "+q);
}
}
classConstructorOverlaoding
{
public static void main(String args[])
{
int x=2, y=3, z=4;
Cs c=new Cs();
Cs c1=new Cs(x, z );
c1.printData();
float m=7.2F, n=5.2F;
int k=c.add(x,y);
int t=c.add(x,y,z);
floatft=c.add(m, n);
System.out.println("k = "+k);
System.out.println("t = "+t);
System.out.println("ft = "+ft);
}}
```

Write a program in JAVA to create a class Bird also declares the different parameterized constructor to display the name of Birds.

```
class Bird
{
int age;
String     name;

Bird()
{
System.out.println("this is the perrot");
}

Bird(String x)
{

name=x;
System.out.println("this is the "+name);
}

Bird(inty,String z)
{

age=y;
name=z;
System.out.println("this is the "+age+"years\t"+name);
}
```

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

Bird a=new Bird();
a.Bird();

Bird b=new Bird("maina");

Bird c=new  Bird(20,"sparrow");
}

}
```

Viva Questions

Q.1 What is the constructor?

AnsThe constructor can be defined as the special type of method that is used to initialize the state of an object. It is invoked when the class is instantiated, and the memory is allocated for the object. Every time, an object is created using the new keyword, the default constructor of the class is called. The name of the constructor must be similar to the class name. The constructor must not have an explicit return type.

Q.2 How many types of constructors are used in Java?

Ans. Based on the parameters passed in the constructors, there are two types of constructors in Java.

- Default Constructor: default constructor is the one which does not accept any value. The default constructor is mainly used to initialize the instance variable with the default values. It can also be used for performing some useful task on object creation. A default constructor is invoked implicitly by the compiler if there is no constructor defined in the class.
- Parameterized Constructor: The parameterized constructor is the one which can initialize the instance variables with the given values. In other words, we can say that the constructors which can accept the arguments are called parameterized constructors.

Q.3 Is constructor inherited?

Ans.No, The constructor is not inherited.

Q.4 What are the differences between the constructors and methods?

Ans.There are many differences between constructors and methods. They are given below.

Java Constructor	Java Method

A constructor is used to initialize the state of an object.	A method is used to expose the behavior of an object.
A constructor must not have a return type.	A method must have a return type.
The constructor is invoked implicitly.	The method is invoked explicitly.
The Java compiler provides a default constructor if you don't have any constructor in a class.	The method is not provided by the compiler in any case.
The constructor name must be same as the class name.	The method name may or may not be same as class name.

Q.5Can we have both Default Constructor and Parameterized Constructor in the same class?

Ans.Yes, we have both Default Constructor and Parameterized Constructor in the same class.

4.Inheritance,Super& Method overriding

```
//Java program to illustrate the
// concept of single inheritance
importjava.util.*;
importjava.lang.*;
importjava.io.*;

classone
{
   publicvoidprint_geek()
   {
      System.out.println("Geeks");
   }
}

classtwo extendsone
{
   publicvoidprint_for()
   {
      System.out.println("for");
   }
}
// Driver class
```

```
publicclassMain
{
   publicstaticvoidmain(String[] args)
   {
      two g = newtwo();
      g.print_geek();
      g.print_for();
      g.print_geek();
   }
}

// Java program to illustrate the
// concept of Multilevel inheritance
importjava.util.*;
importjava.lang.*;
importjava.io.*;

classone
{
   publicvoidprint_geek()
   {
      System.out.println("Geeks");
   }
}

classtwo extendsone
{
   publicvoidprint_for()
   {
      System.out.println("for");
   }
}

classthree extendstwo
{
   publicvoidprint_geek()
   {
      System.out.println("Geeks");
   }
}

// Drived class
publicclassMain
{
   publicstaticvoidmain(String[] args)
   {
      three g = newthree();
      g.print_geek();
      g.print_for();
      g.print_geek();
   }
}
```

```
// A Simple Java program to demonstrate
// method overriding in java

// Base Class
classParent {
   voidshow()
   {
      System.out.println("Parent's show()");
   }
}

// Inherited class
classChild extendsParent {
   // This method overrides show() of Parent
   @Override
   voidshow()
   {
      System.out.println("Child's show()");
   }
}

// Driver class
classMain {
   publicstaticvoidmain(String[] args)
   {
      // If a Parent type reference refers
      // to a Parent object, then Parent's
      // show is called
      Parent obj1 = newParent();
      obj1.show();

      // If a Parent type reference refers
      // to a Child object Child's show()
      // is called. This is called RUN TIME
      // POLYMORPHISM.
      Parent obj2 = newChild();
      obj2.show();
   }
}
```

Super Keyword in Java

1. Use of super with variables

```
/* Base class vehicle */
classVehicle
{
   intmaxSpeed = 120;
}

/* sub class Car extending vehicle */
```

```
classCar extendsVehicle
{
   intmaxSpeed = 180;

   voiddisplay()
   {
      /* print maxSpeed of base class (vehicle) */
      System.out.println("Maximum Speed: "+ super.maxSpeed);
   }
}
/* Driver program to test */
classTest
{
   publicstaticvoidmain(String[] args)
   {
      Car small = newCar();
      small.display();
   }
}
```

2. Use of super with methods:

```
/* Base class Person */
classPerson
{
   voidmessage()
   {
      System.out.println("This is person class");
   }
}

/* Subclass Student */
classStudent extendsPerson
{
   voidmessage()
   {
      System.out.println("This is student class");
   }

   // Note that display() is only in Student class
   voiddisplay()
   {
      // will invoke or call current class message() method
      message();

      // will invoke or call parent class message() method
      super.message();
   }
}

/* Driver program to test */
classTest
{
```

```
   publicstaticvoidmain(String args[])
   {
      Student s = newStudent();

      // calling display() of Student
      s.display();
   }
}
```

3. Use of super with constructors:

```
/* superclass Person */
classPerson
{
   Person()
   {
      System.out.println("Person class Constructor");
   }
}

/* subclass Student extending the Person class */
classStudent extendsPerson
{
   Student()
{
      // invoke or call parent class constructor
      super();

      System.out.println("Student class Constructor");
   }
}

/* Driver program to test*/
classTest
{
   publicstaticvoidmain(String[] args)
   {
      Student s = newStudent();
   }
}
```

Viva Questions

Q1. What is super keyword ?

Ans.It is used to access members of the base class.

Q2. What are usage of java super Keyword ?

- super is used to refer immediate parent class instance variable.
- super() is used to invoke immediate parent class constructor.
- super is used to invoke immediate parent class method.

Q3. What is Inheritance in Java?
Ans. Inheritance is an Object oriented feature which allows a class to inherit behavior and data from other class.

Q4. How to use Inheritance in Java?
Ans. You can use Inheritance in Java by extending classes and implementing interfaces. Java provides two keywords extends and implements to achieve inheritance. A class which is derived from another class is known as a subclass and an interface which is derived from another interface is called subinterface. A class which implements an interface is known as implementation.

Q5.Can A Class Extends More Than One Class In Java?

Ans. No, a class can only extend just one more class in Java.

5. Interface, Final & Abstract keyword

Write a program in java to generate an abstract class A also class B inherits the class A. generate the object for class B and display the text "call me from B".

```
abstract class A
{
abstract void call();
}
class B extends A
{
public void call()
{

System.out.println("call me from B");
}

public static void main(String arr[])
{
B b=new B();
b.call();
}

}
```

Write a java program in which you will declare two interface sum and Add inherits these interface through class A1 and display their content.

```
interface sum
{
intsm=90;
voidsuma();
}
Interface add
{
int ad=89;
voidadda();
}
```

```
class A1 implements add ,sum
{
public void suma()
{
System.out.println(+sm);
}


public void adda()
{
System.out.println(+ad);
}
public static void main(String arr[])
{
A1 n= new A1();
n.adda();
n.suma();
}
}
```

Write a java program in which you will declare an abstract class Vehicle inherits this class from two classes car and truck using the method engine in both display “car has good engine” and “truck has bad engine”.

```
abstract class vechile
{
abstract void engine();
}
class car extends vechile
{
public void engine()
{
System.out.println("car has good engine");
}
}
class truck extends vechile
{
public void engine()
{
System.out.println("truck has bad engine");
}
}
public class TestVechile
{
public static void main(String arr[])
{
vechile v=new car();
v.engine();
vechile n=new truck();
n.engine();
}
}
```

Final Keyword In Java

1) Java final variable

Example of final variable

There is a final variable speedlimit, we are going to change the value of this variable, but It can't be changed because final variable once assigned a value can never be changed.

```
1.  class Bike9{
2.   final int speedlimit=90;//final variable
3.   void run(){
4.    speedlimit=400;
5.   }
6.   public static void main(String args[]){
7.   Bike9 obj=new  Bike9();
8.   obj.run();
9.   }
10. }//end of class
```

2) Java final method

If you make any method as final, you cannot override it.

Example of final method

```
1.  class Bike{
2.    final void run(){System.out.println("running");}
3.  }
4.
5.  class Honda extends Bike{
6.     void run(){System.out.println("running safely with 100kmph");}
7.
8.     public static void main(String args[]){
9.     Honda honda= new Honda();
10.    honda.run();
11.    }
12. }
```

3) Java final class

If you make any class as final, you cannot extend it.

Example of final class

```
1.  final class Bike{}
2.
3.  class Honda1 extends Bike{
4.    void run(){System.out.println("running safely with 100kmph");}
5.
```

```
6.  public static void main(String args[]){
7.  Honda1 honda= new Honda1();
8.  honda.run();
9.  }
10. }
```

Viva Questions

Q1. What is interface in java?
Java interface is a blueprint of a class and it is used to achieve fully abstraction and it is a collection of abstract methods.

Q2. Can we achieve multiple inheritance by using interface?
Ans. Yes, we can achieve multiple inheritance through the interface in java but it is not possible through class because java doesn't support multiple inheritance through class.

Q3.Is it compulsory for a class which is declared as abstract to have at least one abstract method?

Ans.Not necessarily. Abstract class may or may not have abstract methods.

Q4.Abstract class must have only abstract methods. True or false?

Ans.False. Abstract methods can also have concrete methods.

Q5.What is the main difference between abstract method and final method?

Ans.Abstract methods must be overridden in sub class where as final methods can not be overridden in sub class

6.Arrays

Write a Java Program to finds the average of numbers in an array.

```
classAvg
{
public static void main(String args[])
{
int n=args.length;
float [] x=new float[n];
for(int i=0; i<n; i++)
{
x[i]=Float.parseFloat(
args[i]);
}
float sum=0;
for(int i=0; i<n; i++)
sum=sum+x[i];
floatavg=sum/n;
System.out.println("Average of given numbers is "+avg);
}
}
```

Write a Java Program to finds addition of two matrices.

```
class Add
{
public static void main(String
args[])
{
int [][] x={{1,2,3},
{4,5,6},
{7,8,9}
};
int [][] y={
{11,12,13},
{14,15,16},
{17,18,19}
};
int [][] z=new int[3][3];
for(int i=0; i<3; i++)
for(int j=0; j<3; j++)
{
z[i][j]=x[i][j]+y[i][j];
}
for(int i=0; i<3; i++)
{
for(int j=0; j<3; j++)
{
System.out.print(z[i][j]+" ");
}
System.out.print("\n");
}
}
}
```

Viva Questions

Q1.What is ArrayStoreException in java? When you will get this exception?

Ans. ArrayStoreException is a run time exception which occurs when you try to store non-compatible element in an array object. The type of the elements must be compatible with the type of array object. For example, you can store only string elements in an array of strings. if you try to insert integer element in an array of strings, you will get ArrayStoreException at run time.

Q2. Can you pass the negative number as an array size?

Ans. No. You can't pass the negative integer as an array size. If you pass, there will be no compile time error but you will get NegativeArraySizeException at run time.

Q3.Can you change the size of the array once you define it? OR Can you insert or delete the elements after creating an array?

Ans. No. You can't change the size of the array once you define it. You can not insert or delete the elements after creating an array. Only you can do is change the value of the elements.

Q4. What is the difference between int[] a and int a[] ?

Ans.Both are the legal methods to declare the arrays in java.

Q5. What are the differences between Array and ArrayList in java?

Ans. Array	ArrayList
Arrays are of fixed length.	ArrayList is of variable length.
You can't change the size of the array once you create it.	Size of the ArrayList grows and shrinks as you add or remove the elements.
Array does not support generics.	ArrayList supports generics.
You can use arrays to store both primitive types as well as reference types.	You can store only reference types in an ArrayList.

7.Exception Handling

Write a program in java if number is less than 10 and greater than 50 it generate the exception out of range. Else it displays the square of number.

```
classCustomTest
        {
                public static void main(String arr[])
                {
                try
                {
                int a=Integer.parseInt(arr[0]);
                if(a<0|| a>50)
                throw(new outofRangeException("valid range is 10 to 50"));
                {
                int s=a*a;
                System.out.println("Square is:"+s);
                }
                }catch(Exception ex)
                {
                System.out.println(ex);
                }
}
}
```

Write a program in java to enter the number through command line argument if first and second number is not entered it will generate the exception. Also divide the first number with second number and generate the arithmetic exception.

```
class Divide2
{
public static void main(String arr[])
{
try
        {
        if(arr.length<2)

        throw(new Exception("two argument must be provided"));
        int a= Integer.parseInt(arr[0]);
        int b=Integer.parseInt(arr[1]);
        if(b==0)
        throw(new Exception("second argument should be non zero"));
        int c=a/b;
        System.out.println("result:"+c);
        }

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

Write a program in java to enter the number through command line argument if first and second number .using the method divides the first number with second and generate the exception.

```
class Divide3
        {
        public static int divide(intx,int y)
        {
int z=0;
        try
```

```
                {


                try
                {
                z= x/y;
                }

                finally
        {
        //return Z;
        }
        }
                catch(ArithmeticException ex)
        {
                System.out.println(ex);

                }
        return z;
                }

public static void main(String arr[])
{
try
{
int a=Integer.parseInt(arr[0]);
int b=Integer.parseInt(arr[1]);
int c=divide(a,b);
System.out.println("Result is="+c);
}
catch(Exception e)
{
System.out.println(e);
}
}
```

}

Viva Questions

Q1.What is Exception in Java?

Ans. Exception is an error event that can happen during the execution of a program and disrupts it's normal flow. Exception can arise from different kind of situations such as wrong data entered by user, hardware failure, network connection failure etc.

Whenever any error occurs while executing a java statement, an exception object is created and then JRE tries to find exception handler to handle the exception. If suitable exception handler is found then the exception object is passed to the handler code to process the exception, known as catching the exception. If no handler is found then application throws the exception to runtime environment and JRE terminates the program.

Java Exception handling framework is used to handle runtime errors only, compile time errors are not handled by exception handling framework.

Q2.What are the Exception Handling Keywords in Java?

Ans.There are four keywords used in java exception handling.

1. throw: Sometimes we explicitly want to create exception object and then throw it to halt the normal processing of the program. throw keyword is used to throw exception to the runtime to handle it.
2. throws: When we are throwing any checked exception in a method and not handling it, then we need to use throws keyword in method signature to let caller program know the exceptions that might be thrown by the method. The caller method might handle these exceptions or propagate it to it's caller method using throws keyword. We can provide multiple exceptions in the throws clause and it can be used with main() method also.
3. try-catch: We use try-catch block for exception handling in our code. try is the start of the block and catch is at the end of try block to handle the exceptions. We can have multiple catch blocks with a try and try-catch block can be nested also. catch block requires a parameter that should be of type Exception.
4. finally: finally block is optional and can be used only with try-catch block. Since exception halts the process of execution, we might have some resources open that will not get closed, so we can use finally block. finally block gets executed always, whether exception occurrs or not.

Q3.What is difference between throw and throws keyword in Java?

Ans. throws keyword is used with method signature to declare the exceptions that the method might throw whereas throw keyword is used to disrupt the flow of program and handing over the exception object to runtime to handle it.

Q4.What happens when exception is thrown by main method?

Ans.When exception is thrown by main() method, Java Runtime terminates the program and print the exception message and stack trace in system console.

Q5.What is difference between Checked and Unchecked Exception in Java?

Ans. Checked Exceptions should be handled in the code using try-catch block or else method should use throws keyword to let the caller know about the checked exceptions that might be thrown from the method. Unchecked Exceptions are not required to be handled in the program or to mention them in throws clause of the method.

8 .Package

1. Example of package that import the packagename.*

```
//save by A.java
package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
```

```
//save by B.java
package mypack;
import pack.*;

class B{
  public static void main(String args[]){
   A obj = new A();
   obj.msg();
  }
}
```

Output:Hello

---

2. Example of package by import package.classname

```
//save by A.java

package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
```

```
1. //save by B.java
2. package mypack;
3. import pack.A;
4.
5. class B{
6.   public static void main(String args[]){
7.    A obj = new A();
8.    obj.msg();
9.   }
10. }
```

Output:Hello

3. Example of package by import fully qualified name

```
1. //save by A.java
2. package pack;
3. public class A{
4.   public void msg(){System.out.println("Hello");}
5. }
```

```
1. //save by B.java
2. package mypack;
3. class B{
4.   public static void main(String args[]){
5.    pack.A obj = new pack.A();//using fully qualified name
6.    obj.msg();
7.   }
8. }
```

Output:Hello

Viva Questions

Q1.What is a Java package?

Ans. Package is a collection of related classes and interfaces. Related classes will have the package defined using package keyword.

Q2.Which package is always imported by default?

Ans. java.lang package and no import statement is necessary.

Q3.Is there a performance impact due to a large number of import statements that are unused?

Ans. No. the unused imports are ignored if the corresponding imported class/package is not used.

Q4. Can I import same package/class twice?

Ans. Yes. One can import the same package or same class multiple times. JVM will internally load the class only once no matter how many times one imports the same class.

Q5. Does importing a package imports the sub packages as well?

Ans. No. One will have to import the sub packages explicitly.

9.Multithreading

Write a java program in which thread sleep for 5 sec and change the name of thread.

```
importjava.lang.*;
classThreadTest extends Thread
{
static
{
 Thread t = Thread.currentThread();

//Thread t=new Thread.currentThread();
System.out.println("thread test is loaded by"+t.getName()+"thread");
t.setName("vishal");
System.out.println("changed the name of thread");
System.out.println("suspending thread for 5 sec");
try
{
Thread.sleep(5000);
}
catch(Exception ex){}
}
public static void main(String arr[])
{
Thread t=Thread.currentThread();
System.out.println("main() is invoked in"+t.getName()+"thread...");
}
}
```

Write a java program in which thread sleep for 6 sec in the loop in reverse order from 5 to 1 and change the name of thread.

```
importjava.lang.*;
class Thread1
        {
        public static void main(String arr[])
        {
        Thread t=Thread.currentThread();
        System.out.println("current thread is:"+t);
        t.setName("vishal thread");
        System.out.println("after name chage thread:"+t);
try
        {
for(int n=5;n>0;n--)
        {
System.out.println(n);
```

```
Thread.sleep(6000);
        }
        }catch(InterruptedException e)
        {
System.out.println("main thread is interrupted");
        }
}
}
```

Write a java program for multithread in which user thread and thread started from main method invoked at a time each thread sleep for 1 sec.

```
classUserThread extends Thread
{
public void run()
{
Thread t=Thread.currentThread();
System.out.println("run() is invoked in"+t.getName()+"thread...");
for(int i=1;i<=10;i++)
{
System.out.println("run:"+i);
try
{
Thread.sleep(1000);
}
catch(Exception e){}
}System.out.println("run() is completed");
}
}
classMultiThread
{
public static void main(String arr[])
{
System.out.println("main() started creating an object of user Thread.....");
UserThread t=new UserThread();
System.out.println("directly invoking run() of user thread");
t.run();
System.out.println("control back in main()....");
System.out.println("launching new thread for run() of user thread....");
t.start();
for(int i=10;i>0;i--)
{
System.out.println("main:"+i);
try
{
Thread.sleep(1000);
}
catch(Exception e){}
}System.out.println("main() completed");
}
}
```

Write a java program for to solve producer consumer problem in which a producer produce a value and consumer consume the value before producer generate the next value.

```
class Buffer
	{
	int value;
	boolean produced=false;
public synchronized void produce(int x)
{
	if(produced)
	{
System.out.println("producer enter monitor out of turn..suspend.....");
	try
		{
			wait();
		}catch(Exception e)
		{}
		}
		value=x;
		System.out.println(value+"is produced");
			produced=true;
			notify();
			}
public synchronized void consume()
	{
	if(! produced)
		{
		System.out.println("consumer enterd the monitor out of turn,suspend.....");
		try{
		wait();
		}catch(Exception e)
		{}
			}
		System.out.println(value+"is consumed");
		produced=false;
		notify();
		}
		}
class Producer extends Thread
		{
		Buffer buffer;
		public Producer(Buffer b)
		{
		buffer =b;
		}
public void run()
		{
		System.out.println("producer started ,producing value.....");
		for(int i=1;1<=10;i++)
			buffer.produce(i);
		}
```

```
                }
class Consumer extends Thread
        {
        Buffer buffer;
        public Consumer(Buffer b)
        {
        buffer =b;
        }
        public void run()
                {
                System.out.println("consumer started,consuming value.....");
                for(int i=1;i<=10;i++)
                buffer.consume();
}
}
        class PC1
{
        public static void main(String arr[])
{
        Buffer b=new Buffer();
        Producer p=new Producer(b);
        Consumer c=new Consumer(b);
        p.start();
        c.start();
}
}
```

Write a java program for to solve printer synchronization problem in which all the jobs must be completed in order.

```
class Printer
        {
        public synchronized void Print(String msg)
                {
                System.out.println("[");
try
                {
                Thread.sleep(1000);
                System.out.println(msg);
                Thread.sleep(1000);
}       catch(Exception e)
        {}
System.out.println("]");
        }
                }

        class User extends Thread
{
        String msg;
        Printer p;
        public User(Printer p,String m)
                {
        this.p=p;
        msg=m;
```

```
        }
        public void run()
        {
        p.Print(msg);
        }
        }
classSynDemo
        {
public static void main(String arr[])
        {
        System.out.println("creatin a pointer.....");
        Printer p=new Printer();
        System.out.println("creating two user threads and giving them reference of the printer....");
        User u1=new User(p,"it is user one");
        User  u2=new User(p,"it is user two");
        System.out.println("Starting user threads.....");
        u1.start();         u2.start();}}
```

Viva Questions

Q1.What is thread in Java?
Ans. Thread is a concurrent unit of execution. Or in other words you can say that it is part of process that can run concurrently with other parts of the process.

Q2.What is Multithreading?
Ans.The process of executing multiple threads simultaneously is known as multithreading. Java supports multithreading.The main advantage of multithreading is reducing CPU idle time and improving the CPU utilization. This makes the job to be completed in less time.

Q3.What are the two ways of creating a thread?
Ans. We can create a thread by using any of the two following methods.
1) By implementing Runnable interface
2) By extending Thread class

Q4.Can we call run() method of Thread class?
Ans. We can call run() method if we want but then it would behave just like a normal method and we would not be able to take the advantage of multithreading. In general run() methods starts execution when we call start() method of a Thread class.

Q5.What is synchronization?
Ans. It is a technique of granting access to the shared resources in multithread environment to avoid inconsistencies in the results.

10.I/O and File Handling

Write a java program to create a file and write the text in it and save the file.

```
import java.io.*;
classCreateFile
                {
                public static void main(String arr[])
```

```
                {
if(arr.length<1)
                {
        System.out.println("usage:javacreatefile file name");
        System.exit(0);
                }
try
        {
        BufferedReader b=new BufferedReader(new InputStreamReader(System.in));
        PrintStreamfos=new PrintStream(new FileOutputStream(arr[0]));
        System.out.println("Enter text end to save");
        PrintStream temp=System.out;
        System.setOut(fos);
do

                {
        String str=b.readLine();
        if(str.equalsIgnoreCase("end"));
System.out.println(str);
                break;


                }while(true);

        System.setOut(temp);
        fos.close();
        b.close();
        System.out.println("successfully created");
        }
        catch(Exception ex)
        {
        System.out.println(ex);
        }
        }
        }
```

Write a java program to read a file and display the content on screen.

```
import java.io.*;
class input
{
public static void main(String arr[])
{
try
{
FileInputStreamfis=new FileInputStream("J.text");
int a=fis.read();
System.out.println(a);
}catch(IOException e){}
```

}}

Write a java program to create a folder.

```
import java.io.*;
classMakeFolder
{
public static void main(String arr[])
{
if(arr.length==0)
{
System.out.println("usage:java make folder name or path");
System.exit(0);
}
File f=new File(arr[0]);
if(f.exists()&&f.isDirectory())
System.out.println("already exist");
else if(f.mkdir())
System.out.println("successfully created");
else
System.out.println("cannot be created");
}
}
```

Write a java program to rename a file.

```
import java.io.*;
classChangeName
                {
        public static void main(String arr[])
        {
        if(arr.length<2)
        System.out.println("usage:change name src target");
        System.exit(0);

        File f1= new File(arr[0]);
        File f2=new File(arr[1]);

if(f1.exists()&&!f2.exists())
{
        if(f1.renameTo(f2))
        System.out.println("successfully renamed");
        else
        System.out.println("cannot be renamed");
        }
        else
{
        System.out.println("either source doesnot exist or target already exist");
        }
    }
}
```

Write a java program in which data is read from one file and should be written in another file.name of both file is given through command line arguments.

```
import java.io.*;
classByteCopy
	{
	public static void main(String arr[])
	{
		if(arr.length<2)
	{
	System.out.println("please enter valid array");
		System.exit(0);
	}
try
{

FileInputStreamfis=new FileInputStream(arr[0]);
FileOutputStreamfos=new FileOutputStream(arr[1]);
long t1=System.currentTimeMillis();
//BufferedReader f= new BufferedReader(new InputStreamReader(new FileInputStream(arr[0])));
	//PrintStreamfos=new PrintStream(new FileOutputStream("k.text"));

	while(true)
	{
int c=fis.read();
System.out.println("hello");
if(c==-1)
break;
	fos.write(c);
System.out.println("hello");
	}
long t2=System.currentTimeMillis();
long t=t2-t1;
	fis.close();
	fos.close();
System.out.println("operation is complete in" +t+ " seconds");
	}catch(Exception ex)
	{
//System.out.println("operation is done");
}
}}
```

Write a java program in which data is read from one file and should be written in another file line by line.

```
import java.io.*;
class Copy
{
public static void main(String arr[])
{
if(arr.length<2)
```

```
{
System.out.println("Usage:Javalinecopy source to target file");
System.exit(0);
}
try
{
FileInputStreamfis= new FileInputStream(arr[0]);
FileOutputStreamfos= new FileOutputStream(arr[1]);
long t1=System.currentTimeMillis();
while(true)
{
String str=fis.readLine();
if(str==null)
break;
fos.println(str);
}
long t2=System.currentTimeMillis();
long t=t2-t1;
fis.close();
fos.close();
System.out.println("successfully copied in" +t+"milliseconds");
}
catch(Exception ex)
{
System.out.println(ex);
}
}
}
```

Write a java program to read the the file using BufferReader.

```
import java.io.*;
class Input2
        {
        public static void main(String arr[])
        {
                if(arr.length<1)
        {
        System.out.println("please enter valid array");
                System.exit(0);
        }
try
{

BufferedReader f= new BufferedReader(new InputStreamReader(new FileInputStream(arr[0])));
        PrintStreamfos=new PrintStream(new FileOutputStream("k.text"));
        int c=f.read();
        while((c=f.read())!=-1)
        {
        fos.write(c);
        }
        f.close();
        fos.close();
        }catch(IOException ex)
```

```
        {}
System.out.println("operation is complete");}}
```

Viva questions

Q1.What are the types of I / O streams?
Ans.Thereare two types of I / O streams: byte and character.

Q2.Difference between Reader/Writer and InputStream/Output Stream?
Ans.The Reader/Writer class hierarchy is character-oriented, and the Input Stream/Output Stream class hierarchy is byte-oriented.
Basically there are two types of streams.Byte streams that are used to handle stream of bytes and character streams for handling streams of characters.In byte streams input/output streams are the abstract classes at the top of hierarchy,while writer/reader are abstract classes at the top of character streams hierarchy.

Q3.What an I/O filter?
Ans. I/O filter is an object that reads from one stream and writes to another, usually altering the data in some way as it is passed from one stream to another.

Q4.What are the file access modes?
Ans. RandomAccessFile can open in read (r) or read / write (rw) mode. There is also a "rws" mode, when a file is opened for read-write operations and each change of file data is immediately recorded on a physical device.

Q5.What is common and how do the following streams differ: InputStream, OutputStream, Reader, Writer?
The base class InputStream represents classes that receive data from various sources:

- an array of bytes
- a string (String)
- a file

The class OutputStream is an abstract class that defines stream byte output. In this category are classes that determine where your data goes: to an array of bytes (but not directly to String; it is assumed that you can create them from an array of bytes), to a file or channel.
Character streams have two main abstract classes, Reader and Writer , which control the flow of Unicode characters. The Reader class is an abstract class that defines character streaming input. The Writer class is an abstract class that defines character stream output.

11. Applets

Write a java program to draw Oval, Rectangle,Line and fill the color in it.and display it on Applet.

```
importjava.awt.*;
import java.io.*;
importjava.lang.*;
importjavax.swing.*;
importjava.applet.*;
public class gr extends Applet
{ public void init()
{ setBackground(Color.white);
setForeground(Color.red);
}
```

```
public void paint(Graphics g)
{ g.drawRect(10,100,50,70);
g.fillOval(10,100,50,70);
g.drawString("vishal",50,7);
g.drawLine(100,20,400,70);
g.setColor(Color.blue);
g.drawOval(100,200,50,10);
}
}
```
Compile it by command javac gr.java. In a file a.html type following.

```
<html>
<applet code="gr.java" height=300 width=700>
</applet>
</html>
```

Now give commandappletviewer a.html

Write a java program to display "HelloWorld" on Screen.

```
public class HelloWorldFrame extends JFrame {

        public static void main(String args[]) {
                newHelloWorldFrame();
        }
        HelloWorldFrame() {
                JLabeljlbHelloWorld = new JLabel("Hello World");
                add(jlbHelloWorld);
                this.setSize(100, 100);
                // pack();
                setVisible(true);
        }
}
```

Viva questions

Q.1What is an Applet ?

Ans.A java applet is program that can be included in a HTML page and be executed in a java enabled client browser. Applets are used for creating dynamic and interactive web applications.

Q.2 Explain the life cycle of an Applet.

Ans. An applet may undergo the following states:

- Init: An applet is initialized each time is loaded.
- Start: Begin the execution of an applet.
- Stop: Stop the execution of an applet.
- Destroy: Perform a final cleanup, before unloading the applet.

Q.3 what is the difference between an Applet and a Java Application?

Ans. Applets are executed within a java enabled browser, but a Java application is a standalone Java program that can be executed outside of a browser. However, they both require the existence of a Java Virtual Machine (JVM). Furthermore, a Java application requires a main method with a specific signature, in order to start its execution. Java applets don't need such a method to start their execution. Finally, Java applets typically use a restrictive security policy, while Java applications usually use more relaxed security policies.

Q.4 what are the restrictions imposed on Java applets?

Ans. Mostly due to security reasons, the following restrictions are imposed on Java applets:

- An applet cannot load libraries or define native methods.
- An applet cannot ordinarily read or write files on the execution host.
- An applet cannot read certain system properties.
- An applet cannot make network connections except to the host that it came from.
- An applet cannot start any program on the host that's executing it.

Q.5 Should Applets Have Constructors?

Ans. We don't have the concept of Constructors in Applets. Applets can be invoked either through browser or through Appletviewer utility provided by JDK.