OBJECT ORIENTED PROGRAMMING THROUGH JAVA LAB (R22A0586)

LAB MANUAL

B.TECH
(II YEAR – II SEM)
(2023-24)

Prepared By:
Ms.K.Swetha
Mr. I.Uma Maheshwara Rao
Mr.R.Chandra Shekar
Ms.T.Shilpa

DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous Institution – UGC, Govt. of India)
Recognized under 2(f) and 12 (B) of UGC ACT 1956
Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2008 Certified)
Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India

DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

VISION

- To improve the quality of technical education that provides efficient software engineers with an attitude to adapt challenging IT needs of local, national and international arena, through teaching and interaction with alumni and industry.

MISSION

- Department intends to meet the contemporary challenges in the field of IT and is playing a vital role in shaping the education of the 21st century by providing unique educational and research opportunities.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1 – ANALYTICAL SKILLS

To facilitate the graduates with the ability to visualize, gather information, articulate, analyze, solve complex problems, and make decisions. These are essential to address the challenges of complex and computation intensive problems increasing their productivity.

PEO2 – TECHNICAL SKILLS

To facilitate the graduates with the technical skills that prepare them for immediate employment and pursue certification providing a deeper understanding of the technology in advanced areas of computer science and related fields, thus encouraging to pursue higher education and research based on their interest.

PEO3 – SOFT SKILLS

To facilitate the graduates with the soft skills that include fulfilling the mission, setting goals, showing self-confidence by communicating effectively, having a positive attitude, get involved in team-work, being a leader, managing their career and their life.

PEO4 – PROFESSIONAL ETHICS

To facilitate the graduates with the knowledge of professional and ethical responsibilities by paying attention to grooming, being conservative with style, following dress codes, safety codes, and adapting themselves to technological advancements.

PROGRAM SPECIFIC OUTCOMES (PSOs)

After the completion of the course, B. Tech Information Technology, the graduates will have the following Program Specific Outcomes:

1. **Fundamentals and critical knowledge of the Computer System:-** Able to Understand the working principles of the computer System and its components , Apply the knowledge to build, asses, and analyze the software and hardware aspects of it .

2. **The comprehensive and Applicative knowledge of Software Development:** Comprehensive skills of Programming Languages, Software process models, methodologies, and able to plan, develop, test, analyze, and manage the software and hardware intensive systems in heterogeneous platforms individually or working in teams.

3. **Applications of Computing Domain & Research:** Able to use the professional, managerial, interdisciplinary skill set, and domain specific tools in development processes, identify the research gaps, and provide innovative solutions to them.

PROGRAM OUTCOMES (POs)

Engineering Graduates should possess the following:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design / development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multi disciplinary environments.
12. **Life- long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:

a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.

b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab.

c. Proper Dress code and Identity card.

4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

HEAD OF THE DEPARTMENT **PRINCIPAL**

MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY
B.TECH - II- YEAR II-SEM-CSIT **L/T/P/C**
-/-/2/1

(R22A0586) OBJECT ORIENTED PROGRAMMING THROUGH JAVA LAB

COURSE OBJECTIVES:

1. To prepare students to become familiar with the Standard Java technologies ofJ2SE
2. To provide Students with a solid foundation in OOP fundamentals required to solve programming problems and also to learn Advanced Java topics like J2ME, J2EE, JSP, JavaScript
3. To train Students with good OOP programming breadth so as to comprehend, analyze, design and create novel products and solutions for the real life problems.
4. To inculcate in students professional and ethical attitude, multidisciplinary approach and an ability to relate java programming issues to broader application context.
5. To provide student with an academic environment aware of excellence, written ethical codes and guidelines and lifelong learning needed for a successful professional career

Week 1:
a) Write a java program to find the Fibonacci series using recursive and non-recursive functions
b) Write a program to multiply two given matrices.
c) Write a program for Method overloading and Constructor overloading

Week 2:
a) Write a program to demonstrate execution of static blocks ,static variables & static methods.
b) Write a program to display the employee details using Scanner class
c) Write a program for sorting a given list of names in ascending order

Week 3:
a) Write a program to implement single and Multi level inheritance
b) Write a program to implement Hierarchical Inheritance.
c) Write a program to implement method overriding.

Week 4:
a) Write a program to create an abstract class named Shape that contains two integers and an empty method named printArea (). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the method printArea () that prints the area of the given shape.
b) Write a program to implement Interface.
c) Write a program to implement multiple and Hybrid Inheritance.

Week 5:
a)Write a program to create inner classes
b)Write a program to create user defined package and demonstrate various access modifiers.
c) Write a program to demonstrate the use of super and final keywords.

Week 6 :
a) Write a program if number is less than 10 and greater than 50 it generate the exception out of range. else it displays the square of number.

b) Write a program with multiple catch Statements.
c) Write a program to implement nested try.

Week 7:
a) Write a Program to implement simple Thread by extending Thread class and implementing runnable interface.
b) Write a program that implements a multi-thread application that has three threads.
c) Write a program to set and print thread priorities.

Week 8:
Write a program to implement following collections
a)array List b) Vector
c)Hash table d)Stack

Week 9:

a) Write a program for producer and consumer problem using Threads.

Week 10:
a) Write a program to list all the files in a directory including the files present in all its subdirectories.
b) Write a Program to Read the Content of a File Line by Line.

Week 11:
a) Write a program that connects to a database using JDBC display all records in a table.
b) Write a program to connect to a database using JDBC and insert values into it.
c) Write a program to connect to a database using JDBC and delete values from it

Week 12:
Write a program that works as a simple calculator. Use a Grid Layout to arrange Buttons for digits and for the + - * % operations. Add a text field to display the result.

COURSE OUTCOMES: Upon successful completion of this course, the students will be able to:
1. Analyze the necessity for Object Oriented Programming paradigm and over structured programming and become familiar with the fundamental concepts in OOP.
2. Demonstrate an ability to design and develop Java programs, analyze, and interpret object oriented data and report results.
3. Analyze the difference between various types of inheritance.
4. Demonstrate an ability to design an object oriented system, AWT components or multithreaded process as per needs and specifications.
5. Demonstrate an ability to visualize and work on laboratory and multidisciplinary tasks like console and windows applications for standalone programs.

INDEX

S.No	Name of the program	Page No
1.	a) Write a java program to find the Fibonacci series using recursive and non-recursive functions	11
	b)Write a program to multiply two given matrices.	13
	c)Write a program for Method and Constructor overloading	15
2.	a) Write a program to demonstrate execution of static blocks ,static variables & static methods.	21
	b) Write a program to display the employee details using Scanner class	23
	c) Write a program for sorting a given list of names in ascending order	24
3.	a) Write a program to implement single and Multi level inheritance	29
	b) Write a program to implement Hierarchical Inheritance.	31
	c) Write a program to implement method overriding	32
4.	a) Write a program to create an abstract class named Shape that contains two integers and an empty method named printArea (). **Provide** three classes named Rectangle, Triangle andCircle such that each one of the classes extends the class Shape. Each one of the classes contains only the method printArea () that prints the area of the given shape.	37
	b) Write a program to implement Interface	39
	c) Write a program to implement multiple and Hybrid Inheritance	40
5.	a) Write a program to create inner classes	45
	b) Write a program to create user defined package and demonstrate various access modifiers.	46
	c) Write a program to demonstrate the use of super and final keywords.	48
6.	a) Write a program if number is less than 10 and greater than 50 it generate the exception out of range. else it displays the square of number.	54

	b) Write a **program** with multiple catch Statements.	55
	c) write a program to implement nested try	55
7.	a) Write a Program to implement simple Thread by extending Thread class and implementing runnable interface.	60
	b) Write a program that implements a multi-thread application that has three threads	61
	c) write a program to set and print thread priorities	63
8.	Write a program to implement following collections a)array List b) Vector c)Hash table d)Stack	68
9.	a) Write a program for producer and consumer problem using Threads	74
10.	a) Write a program to list all the files in a directory including the files present in all its subdirectories.	80
	b)Write a Program to Read the Content of a File Line by Line	82
11.	a) Write a program that connects to a database using JDBC display all records in a table.	86
	b) Write a program to connect to a database using JDBC and insert values into it	88
	c) Write a program to connect to a database using JDBC and delete values from it	89
12.	Write a program that works as a simple calculator. Use a Grid Layout to arrange Buttons for digits and for the + - * % operations. Add a text field to display the result	94

Week 1: **DATE:**

a) Write a java program to find the Fibonacci series using recursive and non-recursive functions

Program: fibonacci series program without using recursion.

```
class FibonacciExample1{
public static void main(String args[])
{
int n1=0,n2=1,n3,i,count=10;
System.out.print(n1+" "+n2);//printing 0 and 1
for(i=2;i<count;++i)//loop starts from 2 because 0 and 1 are already printed
{
 n3=n1+n2;
 System.out.print(" "+n3);
 n1=n2;
 n2=n3;
 }
 }
}
```

output:

Signature of the Faculty

Program: fibonacci series program using recursion.

```
class FibonacciExample2{
 static int n1=0,n2=1,n3=0;
 static void printFibonacci(int count){
    if(count>0){
         n3 = n1 + n2;
         n1 = n2;
         n2 = n3;
         System.out.print(" "+n3);
         printFibonacci(count-1);
     }
 }
 public static void main(String args[]){
  int count=10;
  System.out.print(n1+" "+n2);//printing 0 and 1
  printFibonacci(count-2);//n-2 because 2 numbers are already printed
 }
}
```

output:

Signature of the Faculty

b) Write a java program to multiply two given matrices.

```
class MatrixMultiplication
{
public static void main(String args[])
{
// Accept the number of rows and columns at run time. int
m=Integer.parseInt(args[0]);
int n=Integer.parseInt(args[1]);
// Initialize the arrays.
int a[][]=new int[m][n];
 int b[][]=new int[m][n];
int c[][]=new int[m][n];
int i=2;
// Loop to accept the values into a matrix. for(int
j=0;j<m;j++)
{
for(int k=0;k<n;k++)
{
a[j][k]=Integer.parseInt(args[i]); i++;
}
}
// Loop to accept the values into b matrix. for(int
j=0;j<m;j++)
{
for(int k=0;k<n;k++)
{
            b[j][k]=Integer.parseInt(args[i]);
             i++;
}
}
// Loop to multiply two matrices .
for(int j=0;j<m;j++)
{
for(int k=0;k<n;k++)
  { c[j][k]=0;
for(int l=0;l<m;l++)
{
        c[j][k]=c[j][k]+(a[j][l]*b[l][k]);
}
}
}
// Loop to display the result .
for(int j=0;j<m;j++)
{
for(int k=0;k<n;k++)
{
System.out.print(c[j][k]);
}
System.out.println();
}
}
}
```

outputs:

Signature of the Faculty

c) Write a program for Method overloading and Constructor overloading

Program: Method Overloading.

```
class Adder{
static int add(int a,int b)
{
return a+b;
}
static int add(int a,int b,int c)
{
return a+b+c;
}
}
class TestOverloading1
{
public static void main(String[] args)
{
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(11,11,11));
}
}
```

outputs:

Signature of the Faculty

Program: Constructor overloading.

```
public class Student {
//instance variables of the class
int id;
String name;
Student()
{
System.out.println("this a default constructor");
}
Student(int i, String n)
{
id = i;
name = n;
}
public static void main(String[] args)
{
//object creation
Student s = new Student();
System.out.println("\nDefault Constructor values: \n");
System.out.println("Student Id : "+s.id + "\nStudent Name : "+s.name);
System.out.println("\nParameterized Constructor values: \n");
Student student = new Student(10, "David");
System.out.println("Student Id : "+student.id + "\nStudent Name : "+student.name);
}
}
```

outputs:

Signature of the Faculty

Exercise Programs:

1. Write a java program to print the multiplication table .
2. Write a java program to find all even and odd integers up to a given integer.
3. Write a java program to add and subtract two given matrices.
4. Write a java program that reads a line of integers and displays each integers and the product of all integers use String Tokenizer.

Week 2:

a) Write a program to demonstrate execution of static blocks, static variables & static methods.

```
class Student
 {
    int rollno;
    String name;
    static String college = "MRCET";
    //static method to change the value of static variable
    static void change(){
    college = "JNTUH";
    }
    //constructor to initialize the variable
   Student(int r, String n)
    {
    rollno = r;
    name = n;
    }
    //method to display values
    void display()
     {
     System.out.println(rollno+" "+name+" "+college);
     }
    }
 //Test class to create and display the values of object
 public class TestStaticMethod
   {
    public static void main(String args[])
   {
    Student.change();//calling change method
    //creating objects
    Student s1 = new Student(111,"Karan");
    Student s2 = new Student(222,"Aryan");
    Student s3 = new Student(333,"Sonoo");
    //calling display method
    s1.display();
    s2.display();
    s3.display();
    }
    static
    {
    System.out.println("static block is invoked");
    }
}
```

outputs:

Signature of the Faculty

b) Write a program to display the employee details using Scanner class

```
import java.util.*;
class Employee
  {
    int emp_id, salary;
   String name;
  void read()
  {
  Scanner sc=new Scanner(System.in);
  System.out.println(“Enter employee id,salary and name”);
  Emp_id=sc.nextInt();
  Salary=sc.nextInt();
  Name=sc.next();
  }
  void display()
   {
  System.out.println(“Employee Details”);
  System.out.println(“Employee id=”+emp_id);
  System.out.println(“Employee Salary=”+salary);
  System.out.println(“Employee Name=”+name);
  }
  }
  class EmployeeDetails
  {
  public static void main(String args[])
  {
  Employee e= new Employee();
  e.read();
  e.display();
  }
  }
```

outputs:

Signature of the Faculty

c) **Write a program for sorting a given list of names in ascending order**

```
import java.util.Arrays;
public class SortStringArrayExample1
{
public static void main(String args[])
{
//defining an array of type String
String[] countries = {"Zimbabwe", "South-
Africa", "India", "America", "Yugoslavia", " Australia", "Denmark", "France", "Netherlands"
, "Italy", "Germany"};
int size = countries.length;
//logic for sorting
for(int i = 0; i<size-1; i++)
{
for (int j = i+1; j<countries.length; j++)
{
//compares each elements of the array to all the remaining elements
if(countries[i].compareTo(countries[j])>0)
{
//swapping array elements
String temp = countries[i];
countries[i] = countries[j];
countries[j] = temp;
}
}
}
//prints the sorted array in ascending order
System.out.println(Arrays.toString(countries));
}
}
```

outputs:

Signature of the Faculty

Exercise Programs:

1. Write a java program to Read and display the student details using Scanner class.
2. Write a java program to sort the given integers in ascending/descending order.
3. Write a java program to display characters in a string in sorted order.
4. Write a program that uses a sequence input stream to output the contents of two files.

Week 3:

a) **Write a program to implement single and Multi level inheritance**

//Single Inheritance

```
class Animal
{
void eat()
{
System.out.println("eating...");
}
}
class Dog extends Animal
{
void bark()
{
System.out.println("barking...");
}
}
class TestInheritance{
public static void main(String args[])
{
Dog d=new Dog();
d.bark();
d.eat();
}
}
```

outputs:

Signature of the Faculty

//Multilevel Inheritance:

```
class Animal
{
Void eat()
{
System.out.println("eating...");
}
}
class Dog extends Animal
{
void bark()
{
System.out.println("barking...");
}
}
class BabyDog extends Dog
{
void weep()
{
System.out.println("weeping...");
}
}
class TestInheritance2
{
public static void main(String args[])
{
BabyDog d=new BabyDog();
d.weep();
d.bark();
d.eat();
}
}
```

outputs:

Signature of the Faculty

b) Write a program to implement Hierarchical Inheritance.

```
class Animal
{
Void eat()
{
System.out.println("eating...");
}
}
class Dog extends Animal
{
void bark()
{
System.out.println("barking...");
}
}
class Cat extends Animal
{
void meow()
{
System.out.println("meowing...");
}
}
class TestInheritance3
{
public static void main(String args[])
{
Cat c=new Cat();
c.meow();
c.eat();
//c.bark();//C.T.Error
}
}
```

outputs:

Signature of the Faculty

c) Write a program to implement method overriding

```
class Vehicle
{
  void run()
{
System.out.println("Vehicle is running");
}
  }
  //Creating a child class
  class Bike extends Vehicle
{
   public static void main(String args[])
{
    //creating an instance of child class
    Bike obj = new Bike();
    //calling the method with child class instance
    obj.run();
    }
     }
```

outputs:

Signature of the Faculty

Exercise Programs:

1. Develop a project to find out the volume of a box, then create a child class, BoxWeight, and inherit the Box class. Create another class, Shipment, and inherit the BoxWeight class.

2. Develop code on Every employee has a standard salary of Rs.50000. For a full-time employee, increment the salary by 50%, and increment the salary by 25% for an intern. After increasing the salary, display the incremented salary.

3. Create the base class as Human, and the child classes are Teacher and Doctor. When the job() method is called, the job() method in the child class has a specific implementation. When the job() method is called, the appropriate method will be called depending on the runtime object.

Week 4:

a) Write a program to create an abstract class named Shape that contains two integers and an empty method named printArea (). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the method printArea () that prints the area of the given shape.

```
abstract class Shape
{
abstract void numberOfSides();
}
// Classes that illustrates the abstract method.
class Trapezoid
{
void numberOfSides()
{
System.out.println("The no. of side's in trapezoidal are6");
}
}
class Triangle
{
void numberOfSides()
{
System.out.println("The no. of side's in triangle are:3 ");
}
}
class Hexagon
{
void numberOfSides()
{
System.out.println("The no. of side's in hexagon are:6 ");
}
}
// Class that create objects and call the method.
class ShapeDemo
{
public static void main(String args[])
{
Trapezoid obj1 = new Trapezoid();
Triangle obj2 = new Triangle();
Hexagon obj3 = new Hexagon();
obj1.numberOfSides();
obj2.numberOfSides();
obj3.numberOfSides();
}
}
```

outputs:

Signature of the Faculty

b) Write a program to implement Interface

```
interface Inf1
{
public void method1();
}
interface Inf2 extends Inf1
{
public void method2();
}
public class Demo implements Inf2
{
/* Even though this class is only implementing the interface Inf2, it has to implement all
the methods of Inf1 as well because the interface Inf2 extends Inf1 */
public void method1()
{
System.out.println("method1");
}
public void method2()
{
System.out.println("method2");
}
public static void main(String args[])
{
Inf2 obj = new Demo();
obj.method2();
}
}
```

outputs:

Signature of the Faculty

c) Write a program to implement multiple and Hybrid Inheritance

```
class HumanBody
   {
   public void displayHuman()
   {
   System.out.println("Method defined inside HumanBody class");
   }
   }
   interface Male
   {
   public void show();
   }
   interface Female
   {
   public void show();
   }
   public class Child extends HumanBody implements Male, Female
   {
   public void show()
   {
   System.out.println("Implementation of show() method defined in interfaces Male and Female
   ");
   }
   public void displayChild()
   {
   System.out.println("Method defined inside Child class");
   }
   public static void main(String args[])
   {
   Child obj = new Child();
   System.out.println("Implementation of Hybrid Inheritance in Java");
   obj.show();
   obj.displayChild();
   }
   }
```

outputs:

Signature of the Faculty

Exercise Programs:

1. Design an application for different flight availability and bookings using abstract classes and interfaces.

Week 5:

a) Write a java program to create inner classes.

```
class Outer_Demo
 {
int num;
// inner class
private class Inner_Demo
 {
public void print()
 {
System.out.println("This is an inner class");
}
}
// Accessing he inner class from the method within
void display_Inner()
 {
Inner_Demo inner = new Inner_Demo();
inner.print();
}
}
public class My_class
{
public static void main(String args[])
 {
// Instantiating the outer class
Outer_Demo outer = new Outer_Demo();
// Accessing the display_Inner() method.
outer.display_Inner();
}
}
```

<u>**outputs:**</u>

Signature of the Faculty

b) Write a program to create user defined package and demonstrate various access modifiers.

```
 package mypack;
public class Simple
 {
 public static void main(String args[])
   {
    System.out.println("Welcome to package");
    }
   }
```

Compilation: **javac -d . Simple.java**

Using packagename.*

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

Using packagename.classname

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

Using fully qualified name

```
package mypack;
class B
{
public static void main(String args[])
{
pack.A obj = new pack.A();//using fully qualified name
obj.msg();
}
  }
```

Compilation: **javac -d . B.java**
Running: java mypack.B

<u>outputs:</u>

Signature of the Faculty

c) Write a program to demonstrate the use of super and this keywords.

```
//super keyword
    class Person
    {
    int id;
    String name;
    Person(int id,String name)
    {
    this.id=id;
    this.name=name;
    }
    }
    class Emp extends Person
    {
    float salary;
    Emp(int id,String name,float salary)
    {
    super(id,name);//reusing parent constructor
    this.salary=salary;
    }
    void display(){System.out.println(id+" "+name+" "+salary);
    }
    }
    class TestSuper5
    {
    public static void main(String[] args)
    {
    Emp e1=new Emp(1,"ankit",45000f);
    e1.display();
    }
    }
```

outputs:

Signature of the Faculty

```
//final keyword

class Bike
{
final void run()
{
System.out.println("running");
}
}
class Honda extends Bike
{
void run()
{
System.out.println("running safely with 100kmph");
}
public static void main(String args[])
{
Honda honda= new Honda();
honda.run();
}
}
```

outputs:

Signature of the Faculty

Exercise Programs:

1. How to invoke parent class method from child class,if method is overridden in the child class?

Week 6:

a) Write a java program for creating multiple catch blocks.

```
class MultipleExceptionHandling
{
public static void main(String args[])
{
try
{
System.out.println("Begin: try block");
String name="abc";
int nameLength = name.length();
int res= 10/0;
System.out.println("End: try block");
}
catch(ArithmeticException e)
{
System.out.println("Division is not allowed with zero denominator");
}
catch(NullPointerException e)
{
System.out.println("Name should not be Null");
}
catch(Exception e)
{
System.out.println("General Exception");
}
System.out.println("End: try-catch block");
}
}
```

outputs:

Signature of the Faculty

b) write a program to implement nested try

```
public class NestedTryBlock{
 public static void main(String args[]){
 //outer try block
  try{
  //inner try block 1
    try{
     System.out.println("going to divide by 0");
     int b =39/0;
   }
    //catch block of inner try block 1
    catch(ArithmeticException e)
    {
      System.out.println(e);
    }
    //inner try block 2
    try{
    int a[]=new int[5];
      //assigning the value out of array bounds
     a[5]=4;
     }
      //catch block of inner try block 2
    catch(ArrayIndexOutOfBoundsException e)
    {
       System.out.println(e);
    }
      System.out.println("other statement");
  }
  //catch block of outer try block
  catch(Exception e)
  {
    System.out.println("handled the exception (outer catch)");
  }
      System.out.println("normal flow..");
 }
}
```

<u>**outputs:**</u>

Signature of the Faculty

Exercise Programs:

1. Write a program if number is less than 10 and greater than 50 it generate the exception out of range. else it displays the square of number.
2. Write a program to implement try with catch block

Week 7:

a) Write a Program to implement simple Thread by extending Thread class and implementing runnable interface.

// **by extending Thread class**

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

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

outputs:

Signature of the Faculty

b) Write a Java program that implements a multi-thread application that has three threads. /* The first thread displays "Good Morning" for every one second, the second thread displays "Hello" for every two seconds and third thread displays "Welcome" for every three seconds */

```
class GoodMorning extends Thread
```

```
{
synchronized public void run()
{
try
{
int i=0;
while (i<5)
{
sleep(1000);
System.out.println("Good morning ");
i++;
}
}
catch (Exception e)
{
}
}
}
class Hello extends Thread
{
synchronized public void run()
{
try
{
int i=0;
while (i<5)
{
sleep(2000);
System.out.println("hello");
i++;
}
} catch (Exception e)
{
}
}
}
class Welcome extends Thread
{
synchronized public void run()
```

```
        {
        try
        {
        int i=0;
        while (i<5)
        {
        sleep(3000);
        System.out.println("welcome");
        i++;
        }
        } catch (Exception e)
        {
        }
        }
        }
        class MultithreadDemo
        {
        public static void main(String args[])
        {
        GoodMorning t1 = new GoodMorning();
        Hello t2 = new Hello();
Welcome t3 = new Welcome();
t1.start();
t2.start();
t3.start();
}
}
```

outputs:

Signature of the Faculty

c) write a program to set and print thread priorities

```
public class JavaSetPriorityExp4 extends Thread
{
        public void run()
        {
            System.out.println("running...");
        }
        public static void main(String args[])
        {
            // creating one thread
            JavaSetPriorityExp4 t1=new JavaSetPriorityExp4();
            JavaSetPriorityExp4 t2=new JavaSetPriorityExp4();
            // set the priority
            t1.setPriority(4);
            t2.setPriority(7);
            // print the user defined priority
            System.out.println("Priority of thread t1 is: " + t1.getPriority()); //4
            System.out.println("Priority of thread t2 is: " + t2.getPriority()); //7
            // this will call the run() method
            t1.start();
        }
    }
```

outputs:

Signature of the Faculty

Exercise Programs:

1. write a program to block a thread using sleep()
2. write a program to display and modify the name of threads

Week 8:

Write a program to implement following collections
a) Array List b) Vector c) Hash table d) Stack

```
//Array List
import java.util.*;
class TestJavaCollection1{
public static void main(String args[]){
ArrayList<String> list=new ArrayList<String>();//Creating arraylist
list.add("Ravi");//Adding object in arraylist
list.add("Vijay");
list.add("Ravi");
list.add("Ajay");
//Traversing list through Iterator
Iterator itr=list.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
}
}
}

//Vector
import java.util.*;
public class TestJavaCollection3{
public static void main(String args[]){
Vector<String> v=new Vector<String>();
v.add("Ayush");
v.add("Amit");
v.add("Ashish");
v.add("Garima");
Iterator<String> itr=v.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
}
}
}
```

// Hash Table

```
import java.util.*;
class Hashtable1{
 public static void main(String args[]){
  Hashtable<Integer,String> hm=new Hashtable<Integer,String>();
  hm.put(100,"Amit");
  hm.put(102,"Ravi");
  hm.put(101,"Vijay");
  hm.put(103,"Rahul");
  for(Map.Entry m:hm.entrySet()){
   System.out.println(m.getKey()+" "+m.getValue());
  }
 }
}
//Stack
import java.util.*;
public class TestJavaCollection4{
public static void main(String args[]){
Stack<String> stack = new Stack<String>();
stack.push("Ayush");
stack.push("Garvit");
stack.push("Amit");
stack.push("Ashish");
stack.push("Garima");
stack.pop();
Iterator<String> itr=stack.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
}
}
}
```

outputs:

Signature of the Faculty

Exercise Programs:

Week 9:

a) Write a java program for producer and consumer problem using Threads

```
classInterThreadDemo
{
```

```
          public static void main(String args[])
          {
          Producer p1=new Producer();
          Consumer c1=new Consumer(p1);
          Thread t1=new Thread(p1);
          Thread t2=new Thread(c1);
          t2.start();
          t1.start();
          }
          }
          class Producer extends Thread
          {
          StringBuffersb;
          Producer()
          {
          sb=new StringBuffer();
          }
          public void run()
          {
          synchronized(sb)
          {
          for(int i=0;i<=10;i++)
          {
          try
          {
          sb.append(i+":");
          Thread.sleep(1000);
          System.out.println("appending");
          }
          catch(InterruptedException e)
          {
          System.out.println(e);
          }
          }
          sb.notify();
          }
          }
          }
class Consumer extends Thread
{
```

```
Producer prod;
Consumer(Producer prod)
{
this.prod=prod;
}
public void run()
{
synchronized(prod.sb)
{
try
{
prod.sb.wait();
}
catch(Exception e)
{
System.out.println(e);
}
System.out.println(prod.sb);
}
}
}
```

outputs:

Signature of the Faculty

Exercise Programs:

Week 10:

a) Write a program to list all the files in a directory including the files present in all its subdirectories.

```
import java.io.File;
public class DisplayFileExample1
```

```
{
public void printFileNames(File[] a, int i, int lvl)
{
if(i == a.length)
{
return;
}
for (int j = 0; j < lvl; j++)
{
System.out.print("\t");
}
if(a[i].isFile())
{
System.out.println(a[i].getName());
}
else if(a[i].isDirectory())
{
System.out.println("[" + a[i].getName() + "]");
printFileNames(a[i].listFiles(), 0, lvl + 1);
}
printFileNames(a, i + 1, lvl);
}
public static void main(String[] argvs)
{
String path = "E:\\Documents";
File fObj = new File(path);
DisplayFileExample1 obj = new DisplayFileExample1();
if(fObj.exists() && fObj.isDirectory())
{
File a[] = fObj.listFiles();
System.out.println("= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = ="
);
System.out.println("Displaying Files from the directory: " + fObj);
System.out.println("= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = ="
);
obj.printFileNames(a, 0, 0);
}
}
}
```

outputs:

Signature of the Faculty

b) Write a Program to Read the Content of a File Line by Line

```
import java.io.*;
import java.util.Scanner;
public class ReadLineByLineExample2
{
public static void main(String args[])
{
try
{
FileInputStream fis=new FileInputStream("Demo.txt");
Scanner sc=new Scanner(fis);
while(sc.hasNextLine())
{
System.out.println(sc.nextLine());
}
sc.close();
}
catch(IOException e)
{
e.printStackTrace();
}
}
}
```

outputs:

Signature of the Faculty

Exercise Programs:

Week 11:

a) Write a program that connects to a database using JDBC display all records in a table.

```
import java.sql.Connection;
import java.sql.DriverManager;
public class PostgreSQLJDBC
{
```

```
public static void main(String args[])
{
Connection c = null;
try {
Class.forName("org.postgresql.Driver");
c = DriverManager .getConnection("jdbc:postgresql://localhost:5432/testdb",
"postgres", "123");
} catch (Exception e) {
e.printStackTrace();
System.err.println(e.getClass().getName()+": "+e.getMessage());
System.exit(0);
}
System.out.println("Opened database successfully");
}
}
```

outputs:

Signature of the Faculty

b) Write a program to connect to a database using JDBC and insert values into it.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
public class PostgreSQLJDBC
{
```

```
public static void main(String args[])
{
Connection c = null;
Statement stmt = null;
try {
Class.forName("org.postgresql.Driver");
c = DriverManager.getConnection("jdbc:postgresql://localhost:5432/testdb","manisha","123");
c.setAutoCommit(false);
System.out.println("Opened database successfully");
stmt = c.createStatement();
String sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) "
+ "VALUES (1, 'Paul', 32, 'California', 20000.00 );";
stmt.executeUpdate(sql);
sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) "
+ "VALUES (2, 'Allen', 25, 'Texas', 15000.00);";
stmt.executeUpdate(sql);
sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) "
+ "VALUES (3, 'Teddy', 23, 'Norway', 20000.00);";
stmt.executeUpdate(sql);
sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) "
+ "VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00);";
stmt.executeUpdate(sql);
stmt.close();
c.commit();
c.close();
} catch (Exception e) {
System.err.println( e.getClass().getName()+": "+ e.getMessage());
System.exit(0);
}
System.out.println("Records created successfully");
}
}
```

outputs:

Signature of the Faculty

c) Write a java program to connect to a database using JDBC and delete values from it

```
import
java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
public class PostgreSQLJDBC6 {
public static void main( String args[] )
{
Connection c = null;
Statement stmt = null;
try {
Class.forName("org.postgresql.Driver");
c = DriverManager.getConnection("jdbc:postgresql://localhost:5432/testdb","manisha",
"123");
c.setAutoCommit(false);
System.out.println("Opened database successfully");
stmt = c.createStatement();
String sql = "DELETE from COMPANY where
ID=2;"; stmt.executeUpdate(sql);
c.commit();
ResultSet rs = stmt.executeQuery( "SELECT * FROM COMPANY;");
while ( rs.next() ) {
int id = rs.getInt("id");
String name = rs.getString("name");
int age = rs.getInt("age");
String address = rs.getString("address");
float salary = rs.getFloat("salary");
System.out.println( "ID = " + id );
System.out.println( "NAME = " + name );
System.out.println( "AGE = " + age );
System.out.println( "ADDRESS = " + address );
System.out.println( "SALARY = " + salary );
System.out.println();
}
rs.close();
stmt.close();
c.close();
} catch ( Exception e ) {
System.err.println( e.getClass().getName()+": "+ e.getMessage()
); System.exit(0);
}
System.out.println("Operation done successfully");
}
}
```

outputs:

Signature of the Faculty

Week 12:

a) Write a java program that works as a simple calculator. Use a Grid Layout to arrange Buttons for digits and for the + - * % operations. Add a text field to display the result.

Program:

```
import javax.swing.*;
import javax.swing.JOptionPane; import java.awt.*;
import java.awt.event.*;

// Class that initialize the applet and create calculator. public
class Calculator extends JApplet
{
public void init()
{
CalculatorPanel calc=new CalculatorPanel(); getContentPane().add(calc);
}
}
// Class that creates the calculator panel .
class CalculatorPanel extends JPanel implements ActionListener
{
// Creation of JButton.
JButton n1,n2,n3,n4,n5,n6,n7,n8,n9,n0,plus,minus,mul,div,dot,equal;
static JTextField result=new JTextField("0",45); static String lastCommand=null;
// Create the JObjectPane.
JOptionPane p=new JOptionPane(); double preRes=0,secVal=0,res;
private static void assign(String no)
{
if((result.getText()).equals("0")) result.setText(no);
else    if(lastCommand=="=")
{
result.setText(no); lastCommand=null; } else
result.setText(result.getText()+no);
}

 // Creation of control panel of calculator and adding buttons using GridLayout.
 public CalculatorPanel()
{
setLayout(new GridLayout());
 result.setEditable(false);
 result.setSize(300,200); add(result);
JPanel panel=new JPanel();
panel.setLayout(new GridLayout(5,5));
n7=new JButton("7");
panel.add(n7);
n7.addActionListener(this); n8=new
JButton("8"); panel.add(n8);
n8.addActionListener(this); n9=new
JButton("9");
panel.add(n9);
n9.addActionListener(this); div=new
JButton("/"); panel.add(div);
```

```
div.addActionListener(this); n4=new
JButton("4"); panel.add(n4);
n4.addActionListener(this); n5=new
JButton("5"); panel.add(n5);
n5.addActionListener(this); n6=new
JButton("6"); panel.add(n6);
n6.addActionListener(this); mul=new
JButton("*"); panel.add(mul);
mul.addActionListener(this); n1=new
JButton("1"); panel.add(n1);
n1.addActionListener(this); n2=new
JButton("2"); panel.add(n2);
n2.addActionListener(this); n3=new
JButton("3"); panel.add(n3);
n3.addActionListener(this);
minus=new JButton("-");
panel.add(minus);
minus.addActionListener(this);
dot=new JButton("."); panel.add(dot);
dot.addActionListener(this); n0=new
JButton("0");
 panel.add(n0); n0.addActionListener(this);
 equal=new JButton("=");
panel.add(equal);
equal.addActionListener(this);
plus=new JButton("+");
panel.add(plus);
plus.addActionListener(this);
add(panel);

}
// Implementing method in ActionListener.
public void actionPerformed(ActionEvent ae)
{
if(ae.getSource()==n1)
            assign("1");
else if(ae.getSource()==n2)
            assign("2");
 else if(ae.getSource()==n3)
             assign("3");
else if(ae.getSource()==n4)

            assign("4");
 else if(ae.getSource()==n5)
            assign("5");
```

```
else if(ae.getSource()==n6)
            assign("6");
 else if(ae.getSource()==n7)
            assign("7");
else if(ae.getSource()==n8)
             assign("8");
 else if(ae.getSource()==n9)
            assign("9");
else if(ae.getSource()==n0)
            assign("0");
else     if(ae.getSource()==dot)
{
if(((result.getText()).indexOf("."))==-1) result.setText(result.getText()+"."); }
else     if(ae.getSource()==minus)
{
preRes=Double.parseDouble(result.getText()); lastCommand="-";
result.setText("0");
}
else     if(ae.getSource()==div)
{
preRes=Double.parseDouble(result.getText());
lastCommand="/";
result.setText("0");
 }
else if(ae.getSource()==equal)
{
 secVal=Double.parseDouble(result.getText());
 if(lastCommand.equals("/"))
            res=preRes/secVal;
else if(lastCommand.equals("*"))
            res=preRes*secVal;
else if(lastCommand.equals("-"))
            res=preRes-secVal;
else if(lastCommand.equals("+"))
            res=preRes+secVal;
result.setText(" "+res); lastCommand="=";
}
else     if(ae.getSource()==mul)
{
preRes=Double.parseDouble(result.getText());
lastCommand="*";
result.setText("0");
}
else     if(ae.getSource()==plus)
{
```

```
preRes=Double.parseDouble(result.getText());
lastCommand="+";
result.setText("0");
}
}
}
```

Calculator.html:

```
<applet code="Calculator" width=200 height=300> </applet>
```

outputs:

Signature of the Faculty

Exercise Programs: