B.M.S COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

Submitted in partial fulfillment of the requirements for record of

OBJECT ORIENTED JAVA PROGRAMMING

(23CS3PCOOJ)

Submitted by:

SAI KRUTHIN CR

1BM22CS232

Faculty incharge:

Dr Seema Patil

Department of Computer Science and Engineering
B.M.S College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019

2023-2024

B.M.S COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDEX

papergrid

Date: / /

SL NO	Contents	Date	Signature
1	Sample Program	5/12/23	
2	Quadratic equation program	12/12/23	
3	Program to Calculate SGPA	19/12/23	
4	Program to display book details	26/12/23	
5	Program on abstract class	2/1/24	
6	Program on Bank	16/1/24	
6	Program on CIE, SEE Marks	23/1/24	
7	Program on Exception	30/1/24	
8	Program on Thread	6/1/24	
9	Program on Deadlock	13/2/24	
10	User Interface	20/2/24	

Name: Sai Krutvin CR.
USN: 1BM22CS232.

papergrid
Date: / /

(1) Java program that prints all real solutions to the quadratic equation $ax^2+bx+c=0$. Read a, b, c and use the quadratic formula. If the discriminant b^2-4ac is negative, display that they are not real.

```
import java.util.Scanner;

class Quadratic
{
    int a,b,c;
    double r1,r2,d;
    void getd()
    {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter coefficients");
        a=s.nextInt();
        b=s.nextInt();
        c=s.nextInt();
    }
    void compute()
    {
        while(a==0)
        {
            System.out.println("Not a quadratic eq");
            System.out.println("Enter non zero value");
            Scanner s = new Scanner(System.in);
            a=s.nextInt();
        }
        d=b*b-4*a*c;
```

```
if (d==0)
{
r1 = (-b)/(2*a);
System.out.println("Roots are real");
System.out.println("r1=r2="+r1+r2);
}
else if (d>0)
{
r1 = ((-b)+(Math.sqrt(d)))/(2*a);   (double)
r2 = ((-b)-(Math.sqrt(d)))/(2*a);   (double)
System.out.println("Roots are real and distinct");
System.out.println("r1="+r1+"r2="+r2);
}
else if (d<0)
{
System.out.println("Roots are imginus");
r1 = (-b)/(2*a);
r2 = Math.sqrt(-d)/(2*a);
System.out.println("r1="+r1+"i "+r2);
System.out.println("r2="+r2+"-i"+r2);
}
}
}
class QuadraticMain
{
public static void main(String args[])
{ Quadratic q = new Quadratic();
  q.getd();
} q.compute();
}
```

Output:

Name: Sai Krushin CR

USN: 1BM22CS232.

Enter the coefficients

1

5

6

The roots are real and distinct

r1 = -2.0

r2 = -3.0

Enter the coefficients

3

4

5

The roots are imaginary.

r1 = 0.0 + i 1.1055415967851332

r2 = 0.0 - i 1.1055415967851332

Enter the coefficients

1

2

1

The roots are real and equal.

r1 = r2 = -1.0

Name: Sai Kruthin CR.
USN: 1BM22CS232.

papergrid
Date: / /

(2) Develop a java program to create a class Student with members usn, name, array credits and an array marks. Include methods and display details and method to calculate SGPA of a student.

```
import java.util.Scanner;
class Subject
{
    int subjectMarks;
    int credits;
    int grade;
}
class Student
{
    Subject subject[];
    int i;
    String name;
    String usn;
    double SGPA;
    Scanner s = new Scanner(System.in);
    Student()
    {
        subject = new Subject[8];
        for(i=0; i<8; i++)
            subject[i] = new Subject();
    }
    void getStudentDetails()
    {
        System.out.println("Enter name");
        name = s.next();
        System.out.println("Enter USN");
        usn = s.next();
    }
```

```
void getMarks()
{
  for(i=0; i<8; i++)
  {
    System.out.println("Enter marks of subject"
                       + (i+1) + " : ");
    subject[i].subjectMarks = s.nextInt();

    System.out.println("Enter credits");
    subject[i].subCredits = s.nextInt();
    subject[i].grade = (subject[i].subjectMarks/10)
  }
}
class main
{
  public static void main(String args[])
  {
    Student s1 = new Student();
    s1.getStudentDetails();
    s1.getMarks();
    s1.Compute();
    System.out.println("name: " + s1.name);
    System.out.println("usn: " + s1.usn);
    System.out.println("SGPA: " + s1.SGPA);
  }
}
```

Output:

Enter name:
Karthik
Enter USN:
1BM22CS232

Enter marks of subject 1
84
Enter credits of subject 1
4
Enter marks of subject 2
87
Enter credits of subject 2
3
Enter marks of subject 3
96
Enter credits of subject 3
3
Enter marks of subject 4
85
Enter credits of subject 4
3
Enter marks of subject 5
92
Enter credits of subject 5
3
Enter marks of subject 6
88
Enter credits of subject 6
2
Enter marks of subject 7
88
Enter credits of subject 7
2
Enter marks of subject 8
95
Enter credits of subject 8
1

Date: / /

Name: Krruthin
VSN: 1BM22CS232
SGPA: 9.318181818181818

Name: SaiKrruthin CR
VSN: 1BM22CS232

Name: Sai Krishnick.
USN: 1BM22CS232

papergrid
Date: / /

4. Create a class Book which contains four members: name, author, price, num_pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a toString() method that could display the complete details of the book. Develop a Java program to create n book objects.

```
import java.util.*;
class Book
{
    String name;
    String author;
    int price;
    int numPages;
    public Book(String name, String author, int
                price, int numpages)
    {
        this.name = name;
        this.author = author;
        this.price = price;
        this.numPages = numPages;
    }
public void set(String name, String author,
                int price)
{   this.name = name;
    this.author = author;
    this.price = price;
    this.numPages = numPages;
}
```

```
public String toString()
{
    String name = "Book name : " + this.name + "\n";
    String author = "Author : " + this.author + "\n";
    String price = "price : " + this.price + "\n";
    int numPages = "Number of pages : " + this.numPages + "\n";

    return name + author + price + numPages;
}
public String getName()
{ return this.name; }
public String author()
{ return this.author(); }
public int getNumberofPages()
{ return this.numPages; }
}

public class book_main
{
    public static void main(String args[])
    {
      Scanner s = new Scanner(System.in);
      int n;
      String name;
      String author;
      int price;
      int numPages;
      System.out.println("Enter the number of books");
      n = s.nextInt();
      Books b[];
      b = new Books[n];
    for(int i=0; i<n; i++)
    {   System.out.println("Enter the name");
        name = s.next();
```

```
System.out.println("Enter the author");
author = s.next();
System.out.println("Enter the price");
price = s.nextInt();
System.out.println("Enter the number of pages");
numPages = s.nextInt();
b[i] = new Books(name, author, price, numPages);
}
for(int i=0; i<n; i++)
System.out.println(b[i].toString());
}
}
```

Output:

```
Enter number of pages
2
Enter the name
book1
Enter the author
author1
Enter the price
50
Enter the number of pages
100
Enter the name
book 2
Enter the author
author2
Enter the price
100
Enter the number of pages
200
```

Book name: Book1
Author name: Author1
Price: 50
Number of pages: 100

Book name: BOOK 2
Author name: Author2
Price: 100
Number of pages: 200

Name: Sai Krushin CR
USN: 1BM22CS232

9/23

Name: Sai Kowshik CR
USN: 1BM22CS232.

papergrid
Date: / /

5 Develop a Java program to create an abstract class named Shape that contains two integers and an empty method name printArea(). Provide three classes named Rectangle, Triangle, 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 given shape.

```
import java.util.*;
class InputScanner
{
    int a,b;
    Scanner s=new Scanner(System.in);
    InputScanner()
    {
        if(this.getClass() == Circle.class)   // from main
        {
            System.out.println("Enter the radius");
            a=s.nextInt();
        }
        else
        {
            System.out.println("Enter height and
                                weight");
            a=s.nextInt();
            b=s.nextInt();
        }
    }
}
abstract class Shape extends InputScanner
{
    abstract void printArea();
}
```

```
class rectangle extends Shape
{
    void printArea()
    {
        System.out.println("Area of the rectangle
                            is " + (double)(a*b));
    }
}
class triangle extends Shape
{
    void printArea()
    {
        System.out.println("Area of triangle is" +
                            (double)(a*b)/2);
    }
}
class circle extends Shape
{
    void printArea()
    {
        System.out.println("Area of circle is " +
                            (double)(3.14 * a*a));
    }
}
class AreaMain
{
    public static void main(String args[])
    {
        System.out.println("For triangle ");
        triangle t = new triangle();
        System.out.println("For rectangle");
        rectangle r = new rectangle();
        System.out.println("For circle");
        circle c = new circle();
```

```
t.printArea();
r.printArea();
c.printArea();
    }
}
```

O/P :-

For triangle
Enter height and weight
2
3

For rectangle
Enter height and weight
4
5

For rectangle
Enter radius
5

Area of triangle is : 3.0
Area of rectangle is : 20.0
Area of circle is : 78.5

Name: Sai Krishna C R
USN: 1BM22CS232.

papergrid
Date: / /

WAP on using generics, where static class - using 5 integers and 5 double

LAB-5

Create a class Account that stores customer name, account number, type of account. From this derive the class cur-acct and sav acct to make them more specific to their requirements.
Access the following tasks.
(a) Accept deposit from customer and update balance.
(b) Display the balance.
(c) compute and deposit interest.
(d) Permit withdrawal and update balance.

```
import java.util.*;

class Input
{ Scanner sc = new Scanner(System.in); }

class Account extends Input
{ String name;
  int accNo;
  double balance;
  void getDetails()
  { System.out.println("Enter Name");
    name = sc.nextLine();
    System.out.println("Enter acc No");
    accNo = sc.nextInt();
  }
```

```
void deposit()
{
  System.out.println("Enter amount to deposit");
  double amt = sc.nextDouble();
  balance += amt;
}
void withdraw()
{
  System.out.println("Enter amount");
  double amt = sc.nextDouble();
    if(balance >= amt)
    { balance -= amt;
      System.out.println("Amount Withdrawn");
    }
    else
    System.out.println("Insufficient balance");
  }
void display()
  { System.out.println("Name: "+name);
    System.out.println("Account Number:"+accNo);
    System.out.println("Balance"+balance);
  }
}
Class current extends Account
 {
    double minbal = 500;
    double penalty = 100;
    void withdraw()
    { super.withdraw();
      checkMinBalance();
    }
```

```
private void checkMinBalance()
{   if (balance < MinBal)
        balance -= penalty;
    System.out.println("Penalty applied");
}
}
class Savings extends Account
{
    double interestRate = 0.04;
    void computeInterest()
    {   double interest = balance * InterestRate;
        balance += interest;
        System.out.println("Interest: " + interest);
    }
}
class Bank extends Account
{   public static void main(String[] args)
    {
        Saving ob1 = new Saving();
        Current ob2 = new Current();
        ob1.getDetails();
        ob2.getDetails();
        int choice; String ac;
        System.out.println("Menu");
        System.out.println("1. Deposit \n
        2. Withdraw \n 3. Display \n
        4. Compute Interest (saving only) \n
        5. Exit");
        while(1)
        {   System.out.println("Enter your choice:");
            choice = sc.nextInt();
            System.out.println("Enter acc type");
            ac = sc.next();
```

```
switch(choice)
{   case 1:
      if (acc.equals("Savings"))
       ob1.deposit();
      else
       ob2.deposit();
      break;
    case 2:
      if(acc.equals("Savings"))
         ob1.withdraw();
      else
         ob2.withdraw();
      break;
    case 3:
      if(acc.equals("Savings"))
       ob1.display();
      else
       ob2.display();
     break;
    case 4:
      ob1.computeInterest();
      break;
    case 5: exit(0);
    def
  }
 }
}
```

Output:
Enter customer name: Krishna
Enter acc number: 1
~~Account Type~~
Menu
1. Deposit
2. withdraw
3. Display
4. Compute interest (savings only)
5. Exit
Enter your choice: 1
Enter acc type: Savings
Enter deposit amount: 1000

Enter your choice: 2
Enter acc type: savings
Enter withdrawal amount: 500

Enter choice: 3
Enter account type: Savings
Name: Krishna
Account number: 1
Balance: 500

Enter choice: 1
Enter account type: Current
Enter deposit amount: 500

Enter choice: 2
Enter withdraw: 450
Penalty.

Enter your choice : 2
Enter account type : Saving
Enter withdrawn amount : 1050
Insufficient.

Name: Sai Koutkin CR
USN: 1BM22CS232

papergrid
Date: / /

Lab-6

Display a package CIE which has 12 class - students internal. Student has name, usn, sem. The class internal derived from student has an array that stores internal marks in 5 courses of current semester. Create another package SEE which has the class External which is derived of student. This class has an array to store marks of 5 subjects. Import 2 packages of a file and declare final marks of all 5 subjects of n students

Student.java

```
package ai;
import java.util.*;
public class Student
{ protected String usn = new String();
  protected String name = new String();
  protected int sem;

  public void inputStudentDetails()
  { Scanner in = new Scanner(System.in);
    System.out.println("Enter usn: ");
    usn = in.nextLine();
    System.out.println("Enter name: ");
    name = in.nextLine();
    System.out.println("Enter the sem: ");
    sem = in.nextInt();
  }
```

```
public void displayDetails()
{
  S.O.P("\n USN :" +usn);
  S.O.P("\n Name: " +name);
  S.O.P("\nSem:" +sem);
}
```

Internals.java

```
package cie;
import java.util.Scanner;
public class Internals extends Student
{ protected int marks[] = new int[5];
    public void InputCIEmarks()
      { int i;
        Scanner s = new Scanner(System.in);
        for(int i = 0; i<5; i++)
          System.marks[i] = s.nextInt();
      }
}
```

External.java

```
package see;
import cie.*;
import java.util.Scanner;
public class External extends Internals
{ public int marks[];
    protected int finalMarks[];
    public External
      { marks = new int[5];
        finalMarks = new int[5];
      }
    public void input SEE marks()
      { Scanner s = new Scanner(System.in);
```

```
for(int i=0; i<5; i++)
{ S.O.P("Enter marks");
  marks[i] = s.nextInt();
}
public void calculateFinalMarks()
{ for(int i=0; i<5; i++)
    finalMarks[i] = marks[i]/2 + super.marks[i];
}
public void displayFinalMarks()
{ displayStudentDetails();
  for(int i=0; i<5; i++)
  { S.O.P("Subject " + finalMarks[i]);
  }
}
}
```

Main.java

```
import ue.Externals;
class Main
{ public static void main(String args[])
  { int numOfStudents = 2;
    Externals finalMarks[] = new Externals[numOfStudents];
    for(int i=0; i<numOfStudent; i++)
    { finalMarks[i] = new Externals();
      finalMarks[i].inputStudentDetails();
      S.O.P("Enter CIE marks:");
      finalMarks[i].inputSEEMarks();
    }
    S.O.P("Displaying Data");
    for(int i=0; i<NoOfStudents; i++)
    { finalMarks[i].CalculateFinalMarks();
      finalMarks[i].displayFinalMarks();
    }
}
```

Output:

Enter usn: 136
Enter name: Krishna
Enter sem: 2
Enter CIE marks
Enter marks of sub1: 91
Enter marks of sub2: 92
Enter marks of sub3: 93
Enter marks of sub4: 94
Enter marks of sub5: 95
Enter SEE marks
Enter marks of sub1: 91
Enter marks of sub2: 92
Enter marks of sub3: 94
Enter marks of sub4: 95
Enter marks of sub5: 96
Display data
USN: 136
Name: Krishna
Sem: 2
Subject 1: 91
Subject 2: 92
Subject 3: 94
Subject 4: 95
Subject 5: 96

Viva

1-24

30

Name: Sai Krishna CR
USN: 1BM22CS232

papergrid
Date: / /

Lab 7

WAP that demonstrates handling of exceptions in inheritance tree. Create a base class "Father" and a derived class "Son". In Father class, implement a constructor which takes the age and throws the exception WrongAge() when the input age < 0. In Son class, implement constructor that calls both father and son's age.

```
import java.util.*;
class WrongFage extends Exception
{   WrongFage(String s)
    { super(s);
    }
}
class InputScanner
{ Scanner is = new Scanner(System.in);
}
class Father extends InputScanner
{
    int FatherFage;
    Father () throws WrongFage
    {
      System.out.println("Enter the age of father");
      FatherFage = is.nextInt();
      if (FatherFage < 0)
      { throw new WrongFage("Age cannot be negative");
      }
    }
```

```
void displayFather()
{ S.O.P ("Father Age is : + FatherAge);
}
class Son extends Father
{ int sonAge;
  Son() throws wrongAge
  { super();
    S.O.P ("Enter the age of son: ");
    SonAge = s.nextInt();
    if (sonAge > FatherAge)
    { throw new WrongAge("Son's age cannot be greater than father's age");
    }
    else if (sonAge < 0)
    { throw new wrongAge("Age cannot be negative");
    }
  }
  void displaySon()
  { S.O.P("son age is: " + &SonAge);
  }
}
class Excep
{ public static void main(String args[])
  { try
    { Son son = new Son();
      son.displayFather();
      son.displaySon();
    } catch (WrongAge e)
    { System.err.println("Error" + e.getMessage());
    }
  }
}
```

Output:
Enter Father Age
5
Enter son Age
10
Error! Son's Age cannot be greater than Father's age.

Enter Father Age
-2
Error! Age cannot be negative.

Enter Father Age
45
Enter son Age
-2
Error! Age cannot be negative

Lab-8

Write a program which creates two threads, one thread displaying "BMS College of Engineering" once every 10 seconds and another displaying "CSE" once every two seconds.

```
import java.lang.*;
class DisplayMessage extends Thread
{   String message;
    long interval;

    DisplayMessage(String message, long interval)
    { this.message = message;
      this.interval = interval;
    }

public void run()
 { try()
   { while(true)
     { System.out.println(message);
       Thread.sleep(interval);
     }
   } catch (InterruptedException)
     { S.O.P(Thread.currentThread().getName() +
                    "interrupted");
     }
   }
Public class ThreadDemo
 { public static void main (String[] args)
   {
     DisplayMessage Thread thread1 =
     new DisplayMessage ("BMSCollege of Eng", 10000);
     DisplayMessage thread2 =
       new Display("CSE", 2000);

     thread1.setName("Thread1");
     thread2.setName("Thread2");

     thread1.start();
     thread2.start();
```

Rotate RC.

```
try
{  Thread.sleep(30000);
} catch (InterruptedException e)
    { S.O.P ("Main thread interrupted");
    }

thread1.interrupt(); //interrupt
thread2.interrupt();

S.O.P("Main thread exiting");
}
}
```

Output :-

```
BMS College of Engineering
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
Multi thread exiting
Thread 2 interrupted
Thread 1 interrupted
```

2-24

Name: Sai Krutin C R
USN: 1BM22CS232.

8 Demonstrate inter process communication and deadlock

```
class Q {
int n;
synchronized int get()
{  System.out.println("Got " +n);
   return n;
}
synchronized void put(int n)
{  this.n = n;
   System.out.println("Put +''+n);
}
}
class Producer implements Runnable
{  Q q;
   Producer(Q q)
   {  this.q = q;
      new Thread(this, "Producer").start();
   }
public void run()
{  int i = 0;
   while(i<15)
   {  q.put(i++); }
}}
class Consumer implements Runnable
{  Q q;
   Consumer(Q q)
   {  this.q = q;
      new Thread(this, "Consumer").start(); }
public void run()
   {  int i=0;
      while(i<15) {  int r = q.get(); i++; }}}
```

```
class PC
{ public static void main(String args[])
    {   Q q = new Q();
        new Producer(q);
        new Consumer(q);
        System.out.println("Press Control-C to stop");
    }
}
```

Output:-

Put: 1 Put: 1
~~Get: 1~~ Get: 1
~~Get: 1~~ Put: 2
~~Get: 1~~ Get: 2
~~Get: 1~~ Put: 3
~~Get: 1~~ Get: 3
~~Put: 2~~ Put: 4
~~Put: 3~~ Get: 4
~~Put: 4~~ Put: 5
~~Put: 5~~ Get: 5
~~Put: 6~~
~~Put: 7~~
~~Get: 7~~

9 Program on deadlock

Name: Sai Krutkrim CR
USN: 1BM22CS232
Lab 9

```
class A
{ synchronized void foo(B b)
  { String name = Thread.currentThread().getName();
    System.out.println(name + "entered A.foo");
    try
    { Thread.sleep(1000);
    } catch(Exception e) {
        System.out.println("A interrupted");
    }
    System.out.println(name + "trying to call B.last()");
    b.last();
  }
  void last() {
   System.out.println("Inside A.last");
} }
class B
{ synchronized void bar(A a) {
   String name = Thread.currentThread().getName();
   System.out.println(name + "entered B.bar");
   try
   { Thread.sleep(1000);
   } catch (Exception e)
    { System.out.println("B interrupted");
    }
   System.out.println(name + "trying to call A.last()");
   a.last();
}
void last()
{ System.out.println("Inside A.last"); }
}
class Deadlock implements Runnable
{     A a = new A();
```

```
B b = new B();
Deadlock()
{   Thread.currentThread().setName("Main Thread");
    Thread t = new Thread(this, "Racing Thread");
    t.start();
    a.foo(b);
    System.out.println("Back in main thread");
}
public void run()
{   b.bar(a);
    System.out.println("Back in other thread");
}
public static void main(String args[])
{   new Deadlock();
}
}
```

Output:-

```
MainThread entered A.foo
RacingThread entered B.bar
MainThread trying to call B.last()
Inside A.last
Back in main thread
RacingThread trying to call A.last()
Inside A.last
Back in other thread
```

15/2/24

Name: Sai Krushna CR
USN: 1BM22CS232.

10 Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were zero, the program would throw an ArithmeticException. Display the exception in a message dialog box.

```
import java.awt.*;
import java.awt.event.*;
public class DivisionMain extends Frame
  implements ActionListener
{
  TextField num1, num2;
  Button dResult;
  Label outResult;
  String out = "";
  double resultNum;
  int flag = 0;
  public DivisionMain()
  {
    setLayout(new FlowLayout());
    dResult = new Button("RESULT");
    Label number1 = new Label("Number1: ");
    Label number2 = new Label("Number2: ");
    num1 = new TextField(5);
    num2 = new TextField(5);
    outResult = new Label("Result:", Label.RIGHT);
    add(number1);
    add(num1);
    add(number2); add(dResult);
```

```
add(outResult);
num1.addActionListener(this);
num2.addActionListener(this);
dResult.addActionListener(this);
addWindowListner(new WundowAdapter())
{    public void WundowClosing(WundowEvent we)
     {  System.exit(0); }
});
}
public void actionPerformed(ActionEvent ae)
{ int n1,n2;
  try
  {   if(ae.getSource()==dResult)
      { n1=Integer.parseInt(num1.getText());
        n2=Integer.parseInt(num2.getText());
        if(n2==0)
          throw new ArithmeticException();
        out=n1+" "+n2;
        resultNum=n1/n2;
        out+=String.valueOf(resultNum);
        repaint();
      }
  } catch(NumberFormatException e1)
    flag=1;
    out="Number Format Exception!"+e1;
    repaint();
} catch(ArithmeticException e2)
  { flag=1; out="Divide by 0 Exception!"+e2;
    repaint();
  }
}
public void paint(Graphics g)
{    if(flag==0)
       g.drawString(out,outResult.getX()+
outResult.getWidth(),outResult.getY()+outResult.getHeight()-8);
```

```
else
g.drawString(out, 100, 200);
flag=0;
}

public static void main(String[] args)
{
    DivisionMain1 dm = new DivisionMain1();
    dm.getSize(new Dimension(800,400));
    dm.setTitle("Division Of Integers");
    dm.setVisible(true);
}
}
```

Output :-

Number1 : 10 Number2 : 5 RESULT

Result : 2.0

classmate
Date
Page

~~ActionListener~~

*Action Listner – In Java AWT, Action Listner is an interface used to handle action events generated by user interactions with graphical components like buttons.

*addWindowListner() – Window Listner is an interface used to handle events related to a window, such as opening, closing and deactivating a window. addWindowListner() method is used to register a WindowListner with a window object to listen for these events.

*setLayout() – method allows you to set the layout of the container.

*Button – Button is basically a control component with a label that generates an event when pushed.

*repaint() – is an asynchronous method of applet class.

*setSize() sets the size of this Dimension Object of applet class.

*setTitle() – function defines the title to appear at the top of sketch window.

*setVisible() – method makes frame appears on the screen.

20/2/2024

SOURCE CODE:

PROGRAM-01

Develop a Java program that prints all real solutions to the quadratic equation ax 2 +bx+c = 0. Read in a, b,c and use the quadratic formula. If the discriminate b 2 -4ac is negative, display a message stating that there are no real solutions

```
import java.util.Scanner;
class Quadratic
{
int a,b,c;
double r1,r2,d;
void getd()
{
	Scanner s= new Scanner(System.in);
	System.out.println("Enter the coefficients of a,b,c");
	a=s.nextInt();
	b=s.nextInt();
	c=s.nextInt();
}
void compute()
{
	while (a==0)
	{
		System.out.println("Not a quadratic equation");
		System.out.println("Enter a non zero value for a:");
		Scanner s=new Scanner(System.in);
		a=s.nextInt();
	}
	d=b*b-4*a*c;
	if (d==0)
	{
		r1=(-b)/(2*a);
		System.out.println("Roots are real and equal");
```

```
                System.out.println("Root1=Root2= "+r1);
        }
        else if(d>0)
        {
                r1=((-b)+(Math.sqrt(d)))/(double)(2*a);
                r1=((-b)-(Math.sqrt(d)))/(double)(2*a);
                System.out.println("Roots are real and distinct");
                System.out.println("Root1= "+r1+ "Root2= "+r2);
        }
        else if(d<0)
        {
                System.out.println("Roots are imaginary");
                r1=(-b)/(2*a);
                r2=Math.sqrt(-d)/(2*a);
                System.out.println("Root1= "+r1+ "+i"+r2);
                System.out.println("Root1= "+r1+ "-i"+r2);
        }
}
}

class QuadraticMain
{
        public static void main(String args[])
        {
                Quadratic q=new Quadratic();
                q.getd();
                q.compute();
        }
}
```

PROGRAM-02

Develop a Java program to create a class Student with members usn, name, an array credits and an array marks. Include methods to accept and display details and a method to calculate SGPA of a student.

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

```
class Subject
{
	int subjectMarks;
	int credits;
	int grade;
}
class Student
{
	Subject subject[];
	String name;
	String usn;
	double sgpa;
	Scanner s;
	Student()
	{
		int i;
		subject =new Subject[9];
		for(i=0;i<9;i++)
			subject[i]=new Subject();
		s=new Scanner(System.in);
	}
	void getStudentDetails()
	{
		System.out.println("Enter student name");
		name=s.next();
		System.out.println("Enter student USN");
		usn=s.next();
	}
	void getMarks()
	{
		for(int i=0;i<9;i++)
		{
			System.out.println("Enter marks");
			subject[i].subjectMarks=s.nextInt();
			System.out.println("Enter number of +credits");
			subject[i].credits=s.nextInt();
			subject[i].grade=(subject[i].subjectMarks/10)+1;
			if(subject[i].grade==11)
				subject[i].grade=10;
			if(subject[i].grade<=4)
				subject[i].grade=0;
```

```
		}
	}
	void computeSGPA()
	{
		int effectiveScores=0;
		int totalCredits=0;
		for(int i=0;i<9;i++)
		{
			effectiveScores+=subject[i].grade*subject[i].credits;
			totalCredits+=subject[i].credits;
		}
		sgpa=(double)effectiveScores/(double)totalCredits;
	}
}



class Main
{
	public static void main(String args[])
	{
		Student s1=new Student();
		s1.getStudentDetails();
		s1.getMarks();
		s1.computeSGPA();
		System.out.println("Student name:"+s1.name);
		System.out.println("Student usn:"+s1.usn);
		System.out.println("Student SGPA:"+s1.sgpa);
	}
}
```

PROGRAM-03

Create a class Book which contains four members: name,author, price, num_pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a toString( ) method that could display the complete details of the book. Develop a Java program to create n book objects.

```
import java.util.Scanner;
class Book {
```

```
    String name;
    String author;
    int price;
    int numPages;

    Book(String name, String author, int price, int numPages) {
        this.name = name;
        this.author = author;
        this.price = price;
        this.numPages = numPages;
    }

    public String toString() {
        return "Book name: " + name + "\n" +
               "Author name: " + author + "\n" +
               "Price: " + price + "\n" +
               "Number of pages: " + numPages + "\n";
    }
}

class BooksMain {
    public static void main(String args[]) {
        Scanner s = new Scanner(System.in);
        int n;

        System.out.println("Enter the number of books: ");
        n = s.nextInt();
        Book[] books = new Book[n];

        for (int i = 0; i < n; i++) {
            System.out.println("Enter the name of the book");
            String name = s.next();
            System.out.println("Enter the author of the book");
            String author = s.next();
            System.out.println("Enter the price of the book");
            int price = s.nextInt();
            System.out.println("Enter the pages of the book");
            int numPages = s.nextInt();
            books[i] = new Book(name, author, price, numPages);
        }

        for (int i = 0; i < n; i++) {
```

```
            System.out.println(books[i].toString());
        }
    }
}
```

PROGRAM-04

Develop a Java 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 contain only the method printArea( ) that prints the area of the given shape.

```
import java.util.*;


abstract class Shape{
            double dim1;
            double dim2;
            double rad;
Shape(double a,double b)
{
            dim1=a;
            dim2=b;
}


Shape(double a)
{
            rad=a;
}
abstract void area();
}

class Rectangle extends Shape
{
            Rectangle(double a,double b)
            {
```

```
		super(a,b);
	}
	void area()
	{
		System.out.println("Area of rectangle is:"+(dim1*dim2));
	}
}

class Triangle extends Shape
{
	Triangle(double a,double b)
	{
		super(a,b);
	}
	void area()
	{
		System.out.println("Area of triangle is"+(dim1*dim2)/2);
	}
}

class Circle extends Shape
{
	Circle(double a)
	{
		super(a);
	}
	void area()
	{
		System.out.println("Area of circle is:"+(3.14*rad*rad));
	}
}

class AbstractMain{
	public static void main(String args[])
	{
		Scanner s=new Scanner(System.in);
		System.out.println("Enter the dimensions of rectangle");
		double l=s.nextInt();
		double b=s.nextInt();
		System.out.println("Enter the base and height of traingle");
		double h1=s.nextInt();
		double h2=s.nextDouble();
```

```
                System.out.println("Enter the radius of circle");
                double r=s.nextInt();
                Shape sh;
                Rectangle rect=new Rectangle(l,b);
                Triangle tri=new Triangle(h1,h2);
                Circle cir=new Circle(r);
                sh=rect;
                sh.area();
                sh=tri;
                sh.area();
                sh=cir;
                sh.area();
}
}
```

PROGRAM-05

Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.

Create a class Account that stores customer name, account number and type of account. From this derive the classes Cur-acct and Sav-acct to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks:

a)Accept deposit from customer and update the balance.

b)Display the balance.

c)Compute and deposit interest

d)Permit withdrawal and update the balance

Check for the minimum balance, impose penalty if necessary and update the balance.

```
import java.util.*;
class Account{
String name;
int accno;
String acc_type;
double balance;

Account(String name,int accno,String acc_type,double balance)
{
	this.name=name;
	this.accno=accno;
	this.acc_type=acc_type;
	this.balance=balance;
}

void deposit(double amount)
{
	balance+=amount;
}

void withdraw(double amount)
{
	if((balance-amount)>=0)
	{
		balance-=amount;
	}
	else
	{
		System.out.println("Insufficient balance! Cannot withdraw");
	}
}

void display()
{
	System.out.println("Name:"+" "+name+"Account number:"+" "+accno+"Account type:"+"
"+acc_type+"Balance:"+" "+balance);
}
}

class Sav_acct extends Account
```

```
{
	static double rate=5.0;
	Sav_acct(String name,int accno,double balance)
{
	super(name,accno,"Savings",balance);
}

void interest()
{
	balance+=balance*(rate)/100;
	System.out.println("Balance:"+balance);
}
}

class Curr_account extends Account
{
	private double minBal=500;
	private double serviceCharges=50;
	Curr_account(String name,int accno,double balance)
	{
	super(name,accno,"Current",balance);
	}

void checkMin()
{
	if (balance<minBal)
	{
		System.out.println("Balance is less than minimum, penlaty is imposed"+serviceCharges);
		balance-=serviceCharges;
		System.out.println("Balance is:"+balance);
	}
}
}

class Bank
{
public static void main(String[] args)
	{
		Scanner s=new Scanner(System.in);
		System.out.println("Enter customer name");
		String name=s.next();
		System.out.println("Enter account number");
```

```
int accno=s.nextInt();
System.out.println("Enter the type of account-Current or Savings");
String acc_type=s.next();
System.out.println("Enter the initial balance");
double balance=s.nextDouble();
Account ob1=new Account(name,accno,acc_type,balance);
Sav_acct sa=new Sav_acct(name,accno,balance);
Curr_account cu=new Curr_account(name,accno,balance);
while (true)
{
        if(acc_type.equals("Savings"))
        {
                System.out.println("Menu");
                System.out.println("1.Deposit");
                System.out.println("2.Withdraw");
                System.out.println("3.Compute Interest for savings account");
                System.out.println("4.Display account details");
                System.out.println("5.Exit");
                System.out.println("Enter your choice:");
                int choice=s.nextInt();

switch(choice)
{
        case 1:
        {
                System.out.println("Enter the amount to be deposited");
                double amt1=s.nextDouble();
                sa.deposit(amt1);
                break;
        }
        case 2:
        {
                System.out.println("Enter the amount to be withdrawn");
                double amt2=s.nextDouble();
                sa.withdraw(amt2);
                break;
        }
        case 3:
        {
                sa.interest();
                break;
        }
```

```
                        case 4:
                        {
                                sa.display();
                                break;
                        }
                        case 5:
                        {
                                break;
                        }
                        default:
                        {
                                System.out.println("Enter valid input");
                                break;
                        }
                }

}
else
{
System.out.println("Menu");
System.out.println("1.Deposit");
System.out.println("2.Withdraw");
System.out.println("3.Display account details");
System.out.println("4.Exit");
System.out.println("Enter your choice:");
int choice=s.nextInt();
switch(choice)
                {
                        case 1:
                        {
                                System.out.println("Enter the amount to be deposited");
                                double amt1=s.nextDouble();
                                cu.deposit(amt1);
                                break;
                        }
                        case 2:
                        {
                                System.out.println("Enter the amount to be withdrawn");
                                double amt2=s.nextDouble();
                                cu.withdraw(amt2);
                                break;
                        }
```

```
                        case 3:
                        {
                                cu.display();
                                cu.checkMin();
                                break;
                        }
                        case 4:
                        {
                                break;
                        }
                        default:
                        {
                                System.out.println("Enter valid input");
                                break;
                        }
                }

}
}
}
}
```

PROGRAM-06

Create a package CIE which has two classes- Student and Internals. The class Student has members like usn, name, sem. The class Internals derived from Student has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of Student. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

```
//Student.java
package CIE;

import java.util.Scanner;

public class Student {

protected String usn = new String();
```

```
protected String name = new String();
protected int sem;

public void inputStudentDetails()
{
Scanner sc=new Scanner(System.in);
System.out.println(&amp;quot;Enter student usn&amp;quot;);
usn=sc.next();
System.out.println(&amp;quot;Enter student name&amp;quot;);
name=sc.next();
System.out.println(&amp;quot;Enter student semester&amp;quot;);
sem=sc.nextInt();
}

public void displayStudentDetails() {
System.out.println(&amp;quot;student usn:&amp;quot;+ usn);
System.out.println(&amp;quot;student name:&amp;quot;+n ame);
System.out.println(&amp;quot; student sem:&amp;quot;+ sem);
}
}

//Internals.java
package CIE;
import java.util.Scanner;

public class Internals extends Student {

protected int marks[] = new int[5];

public void inputCIEmarks()
{
Scanner sc=new Scanner(System.in);
for (int i=0;i&amp;lt;5;i++)
{
System.out.println(&amp;quot;Enter 5 subject marks&amp;quot;);
marks[i]=sc.nextInt();
}
}
}

//Externals.java
package SEE;
```

```
import CIE.Internals;

import java.util.Scanner;

public class Externals extends Internals {

protected int marks[];

protected int finalMarks[];

public Externals() {

marks = new int[5];
finalMarks = new int[5];
}

public void inputSEEmarks()
{

Scanner sc = new Scanner(System.in);

for(int i=0;i&lt;5;i++)
{

System.out.print(&quot;Subject &quot;+(i+1)+&quot; marks: &quot;);

marks[i] = sc.nextInt();
}
}

public void calculateFinalMarks() {

for(int i=0;i&lt;5;i++)

finalMarks[i] = marks[i]/2 + super.marks[i];

}

public void displayFinalMarks() {

displayStudentDetails();
```

```
for(int i=0;i&lt;5;i++)

System.out.println(&quot;Subject &quot; + (i+1) + &quot;: &quot; + finalMarks[i]);

}

}

//Main1.java
import SEE.Externals;

class Main1 {

public static void main(String args[])

{

int numOfStudents = 2;

Externals finalMarks[] = new
Externals[numOfStudents];

for(int i=0;i&lt;numOfStudents;i++)

{

finalMarks[i] = new Externals();
finalMarks[i].inputStudentDetails();

System.out.println(&quot;Enter CIE marks&quot;);

finalMarks[i].inputCIEmarks();

System.out.println(&quot;Enter SEE marks&quot;);

finalMarks[i].inputSEEmarks();

}
System.out.println(&quot;Displaying data:\n&quot;);

for(int i=0;i&lt;numOfStudents;i++)
```

```
{

finalMarks[i].calculateFinalMarks();

finalMarks[i].displayFinalMarks();

} //end of for loop

} // end of public main

} //end of class main
```

PROGRAM-07

Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called “Father” and derived class called “Son” which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge( ) when the input age<0. In Son class, implement a constructor that cases both father and son’s age and throws an exception if son’s age is >=father’s age.

```
import java.util.*;
class WrongAge extends Exception
{
	public WrongAge(String s)
	{
		super(s);
	}
}

class Father
{
	Scanner sc=new Scanner(System.in);
	public int F_age;
	public Father() throws WrongAge
	{

		System.out.println("Enter Father's age");
		F_age=sc.nextInt();
```

```
		if (F_age<0)
		{
			throw new WrongAge("Age cannot be negative");
		}
	}
	public void display()
	{
		System.out.println("Father's age="+F_age);
	}
}

class Son extends Father
{
	private int S_age;
	public Son() throws WrongAge
	{
		System.out.println("Enter son's age");
		S_age=sc.nextInt();
		if (S_age>F_age)
		{
			throw new WrongAge("Son's age cannot be greater than father's age");
		}
		else if (S_age<0)
		{
			throw new WrongAge("Age cannot be negative");
		}
		else if (S_age==F_age)
		{
			throw new WrongAge("Age cannot be same");
		}
		else
		{
			System.out.println("Valid age");
		}
	}
	public void display()
	{
		System.out.println("Father's age="+F_age);
		System.out.println("Son's age="+S_age);
	}
}


class Lab7
{
	public static void main(String args[])
	{
```

```
		try
		{
			Son son=new Son();
			son.display();

		}
		catch(WrongAge e)
		{
			System.out.println("Exception caught");
			System.out.println(e.getMessage());
		}
	}
}
```

PROGRAM-08

Write a program which creates two threads, one thread displaying “BMS College of Engineering” once every ten seconds and another displaying “CSE” once every two seconds.

```
class BMS extends Thread{
	public void run(){
		for(int i=1;i<=5;i++){
			System.out.println("bms college of engineering"+i);
			try
			{
				Thread.sleep(10000);
			}
			catch(InterruptedException e){
				e.printStackTrace();
			}
			}
		}
}
class CSE extends Thread{
	public void run(){
		for(int i=1;i<=10;i++){
			System.out.println("Cse"+i);
			try
			{
				Thread.sleep(2000);
			}
```

```
                        catch(InterruptedException e){
                                e.printStackTrace();
                        }
                        }
                }
}
class threadMain{
        public static void main(String a[]){
                BMS obj1=new BMS();
                CSE obj2=new CSE();
                obj1.start();
                obj2.start();
        }
}
```

PROGRAM-09

Write a program that creates a user interface to perform integer divisions.

The user enters two numbers in the text fields, Num1 and Num2. The division of

Num1 and Num2 is displayed in the Result field when the Divide button is clicked.

If Num1 or Num2 were not an integer, the program would throw a

NumberFormatException. If Num2 were Zero, the program would throw an

Arithmetic Exception Display the exception in a message dialog box.

```
import java.awt.*;
import java.awt.event.*;

public class DivisionMain1 extends Frame implements ActionListener
{
        TextField num1,num2;
        Button dResult;
        Label outResult;
        String out="";
        double resultNum;
        int flag=0;
```

```
public DivisionMain1()
{
	setLayout(new FlowLayout());

	dResult = new Button("RESULT");
	Label number1 = new Label("Number 1:",Label.RIGHT);
	Label number2 = new Label("Number 2:",Label.RIGHT);
	num1=new TextField(5);
	num2=new TextField(5);
	outResult = new Label("Result:",Label.RIGHT);

	add(number1);
	add(num1);
	add(number2);
	add(num2);
	add(dResult);
	add(outResult);

	num1.addActionListener(this);
	num2.addActionListener(this);
	dResult.addActionListener(this);
	addWindowListener(new WindowAdapter()
	{
		public void windowClosing(WindowEvent we)
		{
			System.exit(0);
		}
	});
}

public void actionPerformed(ActionEvent ae)
{
	int n1,n2;
	try
	{
		if (ae.getSource() == dResult)
		{
			n1=Integer.parseInt(num1.getText());
			n2=Integer.parseInt(num2.getText());

			/*if(n2==0)
				throw new ArithmeticException();*/
```

```
                        out=n1+" "+n2;
                        resultNum=n1/n2;
                        out+=String.valueOf(resultNum);
                        repaint();

                    }
                }
                catch(NumberFormatException e1)
                {
                    flag=1;
                    out="Number Format Exception!  "+e1;
                    repaint();
                }
                catch(ArithmeticException e2)
                {
                    flag=1;
                    out="Divide by 0 Exception!  "+e2;
                    repaint();
                }

        }

        public void paint(Graphics g)
        {
                if(flag==0)

g.drawString(out,outResult.getX()+outResult.getWidth(),outResult.getY()+outResult.getHeight()-8);
                else
                g.drawString(out,100,200);
                flag=0;
        }

        public static void main(String[] args)
        {
                DivisionMain1 dm=new DivisionMain1();
                dm.setSize(new Dimension(800,400));
                dm.setTitle("DivionOfIntegers");
                dm.setVisible(true);
        }

}
```

PROGRAM-10

10.A)Demonstrate Inter process Communication and deadlock

```
class Q {
	int n;
	boolean valueSet = false;
	synchronized int get() {
		while(!valueSet){
			try {
				System.out.println("Consumer waiting");
				wait();
			} catch(InterruptedException e) {
			System.out.println("InterruptedException caught");
			}

			}
		System.out.println("Got: " + n);
		valueSet = false;
		System.out.println("Intimate Producer\n");
		notify();
		return n;
	}
	synchronized void put(int n) {
		while(valueSet){
			try {
				System.out.println("Producer waiting");
				wait();
				} catch(InterruptedException e) {
					System.out.println("InterruptedException caught");
				}

			}
		this.n = n;
		valueSet = true;
		System.out.println("Put: " + n);
		System.out.println("Intimate Consumer\n");
		notify();
		}
```

```
}

class Producer implements Runnable {
	Q q;
	Producer(Q q) {
		this.q = q;
		new Thread(this, "Producer").start();
	}
	public void run() {
	int i = 0;
	while(i<5) {
		q.put(i++);
	}
	}
}

class Consumer implements Runnable {
	Q q;
	Consumer(Q q) {
		this.q = q;
		new Thread(this, "Consumer").start();
	}
	public void run() {
		int i=0;
		while(i<5) {
			int r=q.get();
			System.out.println("consumed:"+r);
			i++;
		}
	}
}
class PCFixed {
	public static void main(String args[]) {
	Q q = new Q();
	new Producer(q);
	new Consumer(q);
	System.out.println("Press Control-C to stop.");
	}
}
```

10B)DEADLOCK

```
class A {
synchronized void foo(B b) {
String name =Thread.currentThread().getName();
System.out.println(name + " entered A.foo");
try {
Thread.sleep(1000);
} catch(Exception e) {
System.out.println("A Interrupted");
}
System.out.println(name + " trying to call B.last()");
b.last();
}
void last() {
System.out.println("Inside A.last");
}
}
class B {
synchronized void bar(A a) {
String name =Thread.currentThread().getName();
System.out.println(name + " entered B.bar");
try {
```

```
Thread.sleep(1000);

} catch(Exception e) {
System.out.println("B Interrupted");
}
System.out.println(name + " trying to call A.last()");
a.last();
}
void last() {
System.out.println("Inside A.last");
}
}
class Deadlock implements Runnable
{
A a = new A();
B b = new B();

Deadlock() {
Thread.currentThread().setName("MainThread");
Thread t = new Thread(this,"RacingThread");
t.start();
a.foo(b); // get lock on a in this thread.
System.out.println("Back in main thread");
}
public void run() {
b.bar(a); // get lock on b in other thread.
System.out.println("Back in other thread");
}
public static void main(String args[]) {
new Deadlock();

}

}
```