s3. get

s1.display();

s2.display();

s3.display();

Student s4;

s4 = s2;

s4.display(); }

}

26/9
LAB-1

QUADRATIC Equation:-

Develop a Java program that prints all real solutions to the quadratic equations $ax^2+bx+c=0$ read in a, b, c and use quad formal if discriminant is b^2-4ac is negative, display message stating that there is no real solution

```
import java.util.Scanner;

public class QuadraticEquation{

    Public static void main(strings args[]){

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter coefficient a:");

            double a = sc.nextDouble();

        System.out.println("Enter coefficient b:");

            double b = sc.nextDouble();
```

```
System.out.println("Enter coefficient c:");
double c = sc.nextDouble();
double discriminant = b*b - 4*a*c;
if (discriminant > 0) {
double root1 = (-b + Math.sqrt(discriminant)) /
                                        (2*a);
double root2 = (-b + Math.sqrt(discriminant)) / (2*a);
System.out.println("Roots are real and distinct");
System.out.println("Root 1 = " + root1);
System.out.println("Root 2 = " + root2);
}
else if (discriminant == 0) {
double root = -b / (2*a);
System.out.println("Roots are real and equal");
System.out.println("Root = " + root);
}
else {
System.out.println("There are no real solutions");
}
}
}
```

Output :-

Enter the coefficient a :

1

Enter the coefficient b:-

-3

Enter the coefficient c:-

2

The equation has two real & distinct roots:-

Root 1 = 2.0

Root 2 = 1.0

Enter the coefficient a:

1

Enter the coefficient b:

-6

Enter the coefficient c:

9

Roots are real and equal

Root = 3.0

Enter the coefficient a:

1

Enter the coefficient b:-

2

Enter the coefficient c:

5

There are no real solutions.

26/9/24

LAB 2 Develop a Java program to create student class with members USN, name, array of credits, & marks method to calculate SGPA of a student

3|10

SGPA calculation:-

```
import java.util.Scanner;

class Student{
    String USN;
    String name;
    int no_of_subjects;
    int[] credits;
    int[] marks;
    Public void acceptDetails(){
        Scanner SC = new Scanner(System.in);
        System.out.println("Enter USN");
        USN = SC.nextLine();
        system.out.println("Enter name");
        name = sc.nextLine();
        System.out.println("Enter number of subjects:");
        no_of_subjects = SC.nextInt();
        credits = new int[no_of_subjects];
        marks = new int[no_of_subjects];
System.out.println("Enter credits and marks for each
                                    subject:");
for(int i=0; i<no_of_subjects; i++){
```

```
        System.out.println("credits:");
            credits[i], sc.nextInt();
            sout("marks");
            marks[i] = sc.nextInt();
            marks[i] = sc.nextInt();
        }
    }

public void display(){
        System.out.println("Student details:");
        System.out.println("USN:" + USN);
        System.out.println("Name:" + name);
        System.out.println("credits and marks:");
    for(int i=0; i<=no-of-students ; i++){
        System.out.println("subject" + (i) + "" + "credits:
    = " + credits[i] + "marks" + marks[i]);}

}

private int gradepoints (int marks) {
    if (marks >=90) {
        return10;
    }
```

```
elsif (marks >= 80) {
    return 9;
}
else if (marks >= 70) {
    return 8;
}
else if (marks >= 60) {
    return 7;
}
else if (marks >= 50) {
    return 6;
}
else if (marks >= 40) {
    return 5;
}
else {
    return 0;
}
}
```

```
public double calculateSGPA(){
    int totalcredits = 0;
    int sum = 0;
    for(int i=0; i<subjects; i++){
        int gradepoint = getgradepoints(marks[i]);
        sum += gradepoint * credits[i];
    }
    return (double) sum / totalcredits;
  }
}

Public class main {
    public static void main(String[] args){
    Student student = new Student();
    student.acceptDetails();
      student.display();
  double SGPA = student.calculateSGPA();
  System.out.println(SGPA);
  }
}
```

Output:-

Enter USN:

1BM23CS

Enter Nam

Sanjana

Enter cred

Credits

marks

Credits

mark

Credits

marks

Credits

mark

Output:-

Enter USN:

1BM23CS301

Enter Name:-

Sanjana Srinivas

Enter credits and marks for each subject:-

Credits : 4

marks : 96

Credits : 4

marks : 93

Credits : 3

marks : 89

Credits : 3

marks ~~Credits~~ : 86

Credits : 3

marks : 92

Credits : 1

marks : 95

Credits : 1

marks : 95

Credits : 1

marks : 95

Student Details

USN : 1BM23CS301

Name : Sanjana Srinivas

Credits and marks :-

Subject 1 : credits = 4 marks = 96

Subject 2 : credits = 4 marks = 93

subject 3 : credits = 3 marks = 89

Subject 4 : credits = 3 marks = 86

Subject 5 : credits = 3 marks = 92

Subject 6 : credit = 1 marks = 95

Subject 7 : credit = 1 marks = 95

Subject 8 : credit = 1 marks = 95

9.7

3/10/24

9.7

3/10/24

3. Create a class book containing four member: name, author, price, num_pages, include a constructor to set values for members, include methods to set & get details of objects. Include to string() method that could display complete details of book.

Lab 3

```
import java.util.Scanner;

Class Book {

    String name, author;
    double price;
    int numPages;

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

    void setDetails(){
        Scanner sc = new Scanner(System.in)
        System.out.println("Enter name of the Book:");
        name = sc.next();
        System.out.println("Enter author name;");
        author = sc.next();
        System.out.println("Enter price;");
        price = sc.nextInt();
        System.out.println("Enter no of pages;");
        numPages = sc.nextInt();}
```

```
void getDetails(){
sout("Name of the book"+name);
sout("Name of Author"+author);
sout("Price"+price);
sout("Number of pages in the book:"+numPages);}

Public String toString(){
return "Book name:"+name+"\n"+"Author"+
        author+"\n"+"price"+price+"\n"+"number
        of pages:"+numpages);
}

class myBook{
public static void main(String[] args){
Scanner sc; new Scanner(System.in);
sout("Enter the no of books:");
int n=sc.nextInt();
Book[]. books= new Book[n];
for(int i=0;i<n;i++){
books[i]=new book();
books[i].setDetails();
books[i].getDetails();
}
```

System.ou

}

Output

~~Enter USN~~

~~Name~~

~~Sanjana~~

Enter no

Enter

Name

19/10/24 Autho

~~Enter d~~

Numbe

Enter detai

Name : Be

Author :

Price : 40

Number

```
System.out.println("\n All Book Details");
    for (Book Books : Books){
        Sout (Book);
    }
}
```

Output

~~Enter USN & Name :~~

~~1BM23CS361~~

~~Sanjana~~

Enter ~~the subject~~ no of Books : 1

Enter details for Book 1

Name :- Silent Patient

Author : Alexander

19/10/24 ~~Enter det~~ Price : 500

Number of pages = 400

Enter details book 2:

Name : Be

Author : B

Price : 400

Number of pages : 200

Output:-

Book Name : Silent Patient

Author : Alexander

Price : 500

Pages : 400

Book Name : Be

Author : AB

Price : 400

Page : 200

LAB4 24/10

Q4. Develop a Java program 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.

```
abstract class shape {

    private int d1;
    private int d2;

}
Public shape (int d1 , int d2)

{
  this.d1 = d1;
  this.d2 = d2;
 }

~~public int getd1() {~~

 public abstract void printArea();

}

//class Rectangle extending shape

class Rectangle extends shape {
   public Rectangle (int width , int height) {
```

```
        super(width, height);
    }
    public void printArea(){
        int area = d1*d2;
        System.out.println("Area of Rectangle: " + area);
    }
}

class Triangle extends Shape{
    public Triangle(int base, int height){
        super(base, height);
    }
    public void printArea(){
        double area = 0.5 * d1 * d2.
        System.out.println("Area of Triangle" + area);
    }
}

class Circle extends Shape{
    public Circle(int radius) {
        super(int radius, 0);
    }
```

```
public void printArea() {
    double area = Math.PI * d1 * d1;
    System.out.println("Area of circle: " + area);
  }
}

public class main {
  public static void main (String[] args) {
    Shape rectangle = new Rectangle(5,10);
    Shape triangle = new Triangle(5,10);
    Shape circle = new Circle(7);

    rectangle.printArea();
    triangle.printArea();
    circle.printArea();
  }
}
```

24/10/24

Output:-

Area of Rectangle = 50
Area of Triangle = 10.0
Area of Circle = 50.2654

LAB programs :-

Develop a Java program create a class Bank that maintain two kinds of account for its customer, one called saving account and other current account. The savings ~~bank~~ account provides facility. The current account provides cheque book facility but no ~~cheque book~~ facility interest. Current account holders should also maintain a minimum balance and if balance is below that service charge is imposed.

Create a class Account that stores customer name, account no and type of account. Derive cur-acct & sav-act to make them more specific. Accept deposit from customer and update balance. Display balance compute deposit and interest. Permit withdrawal and update interest.

```
import java.util.Scanner

~~class~~
abstract class Account{
    String customerName, account_Number;
    double balance;
    Account( String customerName, String account
```

```
import java.util.Scanner;

Class Account {
    int accountNumber;
    String customerName;
    double balance;
Account (String name, int accNumber){
    customerName = name;
    accountNumber = accNumber;
    balance = 0.0;
}
void deposit (double amount){
    balance += amount;
System.out.println("Deposited" + amount + "\nUpdated
                    balance :" + balance);
}
void withdraw (double amount){
System.out.println ("withdrawal is specific, to account
                    type");
}
```

```
void compute
    System.o
}
}
class Savi
    do
    Savin
}
void
doubl
    bal
system.
void
System.o
```

```
void computeInterest ( ){
    System.out.println("Interest computation not applicable
                        for this account type");
}
}

class SavingAccount extends Account {
    double interestRate = 0.04;

    SavingsAccount (String name, int accno){
        super (name, accno);
    }
    void computeInterest (){
    double interest = balance * interestRate;
    balance += interest;
System.out.println("Interest added:" + account + "\n
                    updated Balance:" + balance);

void withdraw (double amount){
    if (balance >= amount){
        balance -= amount;
System.out.println("withdrawn:" + amount + "\n updated
                    balance" + balance); }
```

```
else {
 System.out.println ("Insufficient balance");
  }
 }
}

class CurrentAccount extends Account {
     double minimumBalance = 500.0;
     double serviceCharge = 50.0;
current Account ( string name, int accno);
{
    super (name, accno);
  void checkminimumBalance ( ) {
    if (balance < minimumBalance) {
     System.out.println ("Balance below minimum
     service charge imposed");
     balance -= serviceCharge;
System.out.println ("service charge :" + serviceCharge
+ "\n updated balance" + balance);
  }
}
```

```
void withdra
    if (bala
     balan
  }
System.out.p
"\n updat
 checkminim
  }
  else {
System.out.
  }
  }
}
public class B
public stat
public
    int acc
    SC.
```

```
void withdraw (double amount){
    if (balance >= amount){
      balance -= amount;
    }
System.out.println ("withdrawn:" + amount +
" \n updated balance :" + balance);

checkminimum balance();
 }
 else{
System.out.println(" Insufficient balance");
 }
 }
}
public class Bank{
public static void main(String [] args) {
        scanner sc = new scanner(system.in);
public
        system.out.print (" Enter customer name");
        String name = sc.nextline();
        system.out.print ("enter acc no");

int accno = sc.nextInt();
 sc.nextline();
```

```
System.out.print ("Enter account type (savings / account):");

String acctype = sc.nextLine();

account account;

if (acctype.equalsIgnoreCase ("current")) {

account = new SavingsAccount (name, accno);

}

else if {

acctype.equalsIgnoreCase ("current")) {

account = new currentaccount (name, aconumber);

}

else {

System.out.println ("Invalid account type");

sc.close();

return;

}
```

```
while(true)
    System.out.println("\n Menu");
    System.out.println("1. Deposit");
    System.out.println("2. withdraw");
    System.out.println("3. Display Balance");
    System.out.println("4. compute Interest");
System.out.print("5. Exit");
    System.out.print("Enter your choice");
int choice = sc.nextInt();

switch(choice){
  case 1:
      System.out.print("Enter deposit amount");
      double depositAmount = sc.nextDouble();
      account.deposit(depositAmount);
      break;

case 2: System.out.print("Enter the withdrawal
                          amount:");
    double withdrawAmount = sc.nextDouble();

      account.withdraw(withdraw);
    break;
```

```
case 3: account.displayBalance();
        break;
case 4: account.computeInterest();
        break;
case 5: system.out.println("Existing");
        sc.close();
        return;
default : system.out.println("Invalid choice");
}
}
}
}
```

output :-

Enter customer name: Sanjana

Enter account number : 101

Chick account type (savings/current): savings

menu;

1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest

5. Exit

Enter your ch

Enter deposit

deposited : 5

updated Bala

Enter your

Interest ac

Updated Ba

Enter you

Enter with

withdra

Updated B

Enter custo

Enter acco

Enter acco

Enter your

Enter depos

Deposit :

Balance :

Enter yo

Enter wi

withdra

5. Exit

Enter your choice : 1
Enter deposit amount : 500
deposited : 500
updated Balance : 500

Enter your choice : 4
Interest added : 20.0
Updated Balance : 520.0

Enter your choice : 2
Enter withdrawal amount : 10
withdrawn : 10.0
Updated Balance : 510.0

Enter customer name : xyz
Enter account number : 134
Enter account type : current

Enter your choice : 1
Enter deposit amount : 210
Deposit : 210.0
Balance : 210.0

Enter your choice : 2
Enter withdrawal amount : 10
withdrawn : 10

updated balance : 200.0

Balance is below minimum service

charge imposed

service charge : 50.0

updated Balance : 150.0

N

27/11/24

LAB - 6.

Creat a Package CIE which has two classes Student and Internals. The class Personal has members like USN, name, sem. The class internals has an array that stores the internal marks scored in five courses of the current sem 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 two packages in a file that declare the final marks of n students.

```
package CIE;
public class student {

    string USN;
    string name;
    int sem
    public [student (string usn, string name, int sem){

        this.usn = usn;
        this.name = name;
        this.sem = sem;

}
public void display() {

    System.out.println ( "USN" + usn);
    System.out.println ("name" + name);
    System.out.println ( "sem" + sem);
```

```
}
}

package CIE;

public class Internals {
    int[] internalmarks = new int[5];

    public Internals(int[] marks) {
        if (marks.length == 5) {
            for(int i=0; i<5; i++){
                internalmarks[i] = marks[i];
            }
        else {
            System.out.println("Please provide exactly 5 marks
                                for internals");
        }
    }

    public void displayInternalMarks() {
        System.out.print("Internal marks");
        for (int mark : internalmarks) {
            System.out.print(mark + " ");
        }
        System.out.println();
    }
}
```

```
package SEE;
import CIE.Student;
public class External extends Student{
    public int[] externalsmarks = new int[5];

public External(String usn, String name, int sem, int[]
                marks){
    super(usn, name, sem);
    if(marks.length==5){
    for(int i=0;i<5;i++){
        externalmarks[i] = marks[i];
    }
    else{
        System.out.println("please provide exactly
                5 marks for SEE");
    }
}
public void displayExternalmarks(){
    System.out.print("SEE marks:");
    for(int mark:externalmarks){
        System.out.print(mark+" ");
    }
    System.out.println();
}}
```

```
import CIE.Student;
import CIE.Internals;
import SEE.External;
import java.util.Scanner;

public class main {
    public static void main (String [] args){
        Scanner sx = new scanner(System.in);
        System.out.print ("Enter number of students");
        int n = sx.nextInt()
    Student [] students = new Student[n];
    Internals[] internals = new Internals[n];
    External[] externals = new External [n];
for (int i=0;i<n;i++){
    System. out.println("Enter details for student"+(i+1))
    system.out.print ("Enter USN");
    String usn = sx.nextline();
    System.out.print("Enter name:");
    String name = sx.nextLine();
    System.out.print("Enter semester");
    int sem = sx.nextInt();
    sx.nextLine();
```

```
System.out.p
int [] inter
for (int j=
    interna
}
Internals int
System.out.p
int [] exte
for (int j =
    external
}
External stud
Student.dis
internals.d
Student.di
Student.d
}
sx.close(
}
}
```

```
System.out.println("Enter internal marks for 5 courses:");
int[] internalMarks = new int[5]

for(int j=0; j<5; j++){
    internalMarks[j] = sx.nextInt();
}

Internals internals = new Internals(internalMarks);

System.out.println("Enter external marks for 5 courses:");
int[] externalMarks = new int[];

for(int j=0; j<5; j++){
    externalMarks[j] = sx.nextInt();
}

External student = new External(usn, sem, externalMarks);

student.displayStudentDetails();
internals.displayInternalMarks();
student.displayExternalMarks();
student.displayFinalMarks(internals);

}

sx.close();

}

}
```

output :-

Enter no of students: 1

Enter details for student# 1:

USN : 301

Name: ABC

Semester: 3

Enter internal marks for 5 course

45
49
50
48
47

Enter SEE marks for 5 courses :-

46
47
48
50
50

21/11/24

Final marks of students:-

USN: 301

Name: 301

Sem : 3

Internal marks: 45 49 50 48 47

SEE marks : 46 47 48 50 50

Final marks : 91 96 98 98 97

LAB PGM: 7

write a program

inheritance tre

derived class "

class Implement

throws the ex

con class, impl

father and

impor j'ava.u

class wrong

public

Sys

}

}

clas Invali

public

Syster

LAB PGM:-

write a program that demonstrates handling exception in inheritance tree. Create a base class called "Father" and derived class "son" which extends the base class. In Father class implement a constructor which takes the age and throws the exception WrongAge() when input age<0. son class, implement constructor that uses both father and son's age and throws an exception if son's age >= father's age.

```
import java.util.Scanner;

class WrongAgeException extends Exception{
    public WrongAgeException(int age){
        System.out.print("Age cannot be negative.
                          Invalid age:"+age);
    }
}

class InvalidSonAgeException extends Exception{
    public InvalidSonAgeException(int fatherAge,
                                  int sonAge){
        System.out.print("Exception: Son's age ("+sonAge+")
                          cannot be greater than or equal
                          to father's age ("+fatherAge+").
                          ");
```

```
class Father{
    int fatherAge;
    public Father(int age) throws wrongAgeException{
        if (age<0){
            throw new wrongAgeException(age);
        }
        this.fatherAge = Age;
    }
}
class son extends Father{
    int sonAge;
    public son(int fatherAge, int sonAge) throws
        wrongAgeException, InvalidSonAgeExcption{
        super(fatherAge);
        if (sonAge<0){
            throw new wrongAgeException(sonAge);
        }
        if (sonAge>=fatherAge){
            throw new InvalidSonAgeException(fatherAge
                sonAge);
        }
        this.sonAge = sonAg
    }
    public void display
        System.out.pr
        System.out.pr
    }
}
public class Exce
    public sta
    Scanner s
    try{
    System.ou
        int f
    system.ou
        int son
    son son =
        son.
    }
```

```
}
this.sonAge = sonAge;
}
public void displayAges(){
    System.out.println("Father's Age" + fatherAge);
    System.out.println("Son's Age" + sonAge);
}
}
public class ExceptionHandlingInheritance2{
    public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    try{
    System.out.print("Enter Father's Age:");
        int fatherAge = scanner.nextInt();

    System.out.print("Enter Son's Age:");
     int sonAge = scanner.nextInt();

    Son son = new Son(fatherAge, sonAge);
        son.displayAges();
    }
```

```
catch (WrongAgeException | InvalidSonAgeException e)
    {
     scanner.close();
      }
    }
}
```

Output :-

```
Enter father's age : -56
Enter son's age : -23
  Age cannot be negative. Invalid age : -56
Father's Age : 56
Son's age : -23
  Age cannot be negative. Invalid age : -23
Father's age : 40
Son's age : 45
Son's age (45) cannot be greater than or equal to father's age (23).
```

Father's Age :
son's Age : 23

N
21/11/24

ption
e)

Father's Age : 56

son's Age : 23.

21/11/24

Lab 8:-

write a program which creates two threads, one thread displaying "BMS college of Engineering") once every every ten seconds and another displaying CSE once every two seconds

```
class BmsCollege thread extends thread
{
public void run(){
    while(true){
        try{
        System.out.println(" BMS college of Engineering");
        thread.sleep(10000);
        }
    catch (InterruptedException e){
        System.out.println(e);
        }
    }
}
}

Class CSEThread extends Thread{
    public void run(){
        while(True){
            try{
```

System.ou
thread.
}
catch (I
}
}
}
}

public class ma
Public sta
BmsColl
CSE thre
bmsThrea
CSEThrea
}
}

```
            System.out.println("CSE");
            Thread.sleep(2000);
        }
        catch (InterruptedException e){
            System.out.println(e);
        }
    }
}

public class main{
    Public static void main (String[] args){
        BmsCollegeThread bmsThread = new BmsCollegeThread();
        CSEThread cseThread = new CSEThread();
        bmsThread.start();
        cseThread.start();
    }
}
```

Output:-

```
import java.util.Scanner;

class Thread1 extends Thread
{
 String x;
 int t;
 Thread1(String a, int t1)
 {
   x=a;
   t=t1;
 }
public void run()
 {
   try{
     while(true)
     {
       System.out.println(x);
       Thread.sleep(t);
     }
   }
  catch(InterruptedException e)
   {
    System.out.println("exit thread");
   }
  }
 }
```

```
class main {
public stati
 {
   Thread 1
   Thread1
     t1.star
      t2.st
   }
 }
```

output :-

BMS colle
CSE
CSE
CSE
CSE
CSE

```
class main {
 public static void main (String args[])
 {
   Thread1 t1 = new Thread1( "BMS college of Eng", 10000);
   Thread1 t2 = new Thread1("CSE", 2000);

    t1.start();
     t2.start();

 }
}
```

AL
28/11/24

Output :-

BMS college of Engineering
CSE
CSE
CSE
CSE
CSE

BMS college of Engineering
CSE
CSE
CSE
CSE
CSE

LAB Program - 9

Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text field 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 ArithmeticException 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 = "NumberFormatException!" + 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, 10, 0, 200);
    flag=0;
  }
```

Output :-

12 3 4

12 0

Zero Division Error ArithmeticException

Demonstrate Interprocess communication & deadlock

```
class Q{
int n;
boolean valueSet = false;
synchronized int get(){
  while(! valueSet)
   try {
    System.out.println("\n consumer waiting \n");
   wait();
   }
  catch (InterruptedException e){
   System.out.println("InterruptedException caught");
   }
system.out.println("Got" + n);
valueSet = false;
System.out.println("\n Intimate Producer \n");
 notify();
 return n;
 }
synchronized void put(int n){
 while(valueSet)
 try {
    System.out.println("\n Producer waiting \n");
 wait();
}
```

```
{
System.out.println("Interrupted exception caught");
}
this.n=n;
valueSet = true;
System.out.println("Put:"+n);
System.out.println("\n 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<15){
q.put(i++);
}
}
}
```

```
class Consumer implements Runnable{
Q q
this.q = q;
new Thread(this,"consumer").start();
}
public void run(){
int i=0;
while(i<15){
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.");
}
}
```

Demonstration of Deadlock

```
class A {
synchronized void foo(B b)
{
 String name = Thread.currentThred().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(); }

synchronized 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( name + "trying to call A.last()");
   a.last(); }
Synchronized void last(){
 System.out.println(" Inside A.last"); }
}
Class Deadlock implements Runnable {
 A a = new A();
 B b = new B();
Deadlock(){
  Thread.currentThred().setName("mainthread");
  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(); }
}
public static void main
public static void main(String[] args )
{
   Division_main1 dm = new Division_main1 ();
    dm.setSize (new Dimension(800,400));
   dm.setTitle ("Division of Integers");
   dm.setVisible (true);
}
}
```