

18/12/23

1] 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.

→ import java.util.Scanner;
class First {

public static void main (String args[]) {

int a, b, c;

double r1, r2, d;

System.out.println("Enter the coefficients of quadratic equation:");

Scanner s1 = new Scanner(System.in);

a = s1.nextInt();

b = s1.nextInt();

c = s1.nextInt();

if (a == 0) {

System.out.println("Coefficients are invalid:");

}

else {

d = b*b - (4*a*c);

if (d > 0) {

System.out.println("It has real and distinct roots");

r1 = ((b + Math.sqrt(d)) / (2*a));

r2 = ((-b - Math.sqrt(d)) / (2*a));

System.out.println("The roots are " + r1 + " and " + r2);

}

if
else if (d == 0) {

System.out.println("It has real and equal roots");

r1 = (-b) / (2*a);

System.out.println("The roots are " + r1 + " and " + r1);

}

```

    else {
        System.out.println("It has no real roots");
    }
}
}
}

```

* Output:

> javac First.java

> java First

Enter the coefficients of quadratic equation

1

2

1

It has real and equal roots

The roots are -1.0 and -1.0

> java First

Enter the coefficients of quadratic equation

1

1

4

It has no real roots.

> java First

Enter the coefficients of quadratic equation

1

4

1

It has real and distinct roots.

The roots are -0.2679491924311228 and -3.732050807568877

Signature

Q. 2] 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 student{

String usn;

String name;

int credits [7];

int marks[];

void acceptDetails(){

Scanner s = new Scanner(System.in);

System.out.print("Enter the usn: ");

usn = s.next();

s.nextLine();

System.out.print("Enter the Name: ");

name = s.next();

s.nextLine();

System.out.print("Enter the no. of Subjects: ");

int n = s.nextInt();

credits = new int [n];

marks = new int [n];

s.nextLine();

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

System.out.println("Enter the credits for subject "+
(i+1) + ":");

credits [i] = s.nextInt();

System.out.println("Enter the marks for subject "+
(i+1) + ":");

marks [i] = s.nextInt();

}
}

void displayDetails(){

System.out.println("USN:" + usn);

System.out.println("Name:" + name);

for (int i=0; i < credits.length; i++){

System.out.println("Subject " + (i+1) + ":" +
credits[i]);

}

}

double calculateSGPA(){

float sum = 0;

for (int i=0; i < 5; i++){

if (marks[i] >= 90){

sum = sum + (credits[i] * 10);

~~if (marks~~

else if (marks[i] >= 80){

sum = sum + (credits[i] * 9);

}

else if (marks[i] >= 70){

sum = sum + (credits[i] * 8);

}

else if (marks[i] >= 60){

sum = sum + (credits[i] * 7);

}

else if (marks[i] >= 50){

sum = sum + (credits[i] * 6);

}

else if (marks[i] >= 35){

sum = sum + (credits[i] * 5);

}

else{

sum = sum + (credits[i] * 0);

}

}


```
int cresum = 0;
for (int i = 0; i < 5; i++) {
    cresum += credits[i];
}
```

```
float sgpa = sum / (float) cresum;
return sgpa;
```

```
}
```

```
}
```

```
public class student_sgpa {
    public static void main (String args []) {
        student s = new student ();
        s.acceptdetails();
        s.displaydetails();
        System.out.println("SGPA: " + s.calculateSGPA());
    }
}
```

Output:

> javac student_sgpa.java

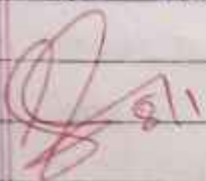
> java student_sgpa

Enter the USN: IBM28CS101

Enter the Name: Anrohi

Enter the no. of subjects: 5

Enter the

 8/1

8/1/24

3] Book program

```
public class Book {
```

```
    String name;
```

```
    String author;
```

```
    double price;
```

```
    int numPages;
```

```
    public Book (String name, String author, double price,  
                int numPages) {
```

```
        this.name = name;
```

```
        this.author = author;
```

```
        this.price = price;
```

```
        this.numPages = numPages;
```

```
    }
```

```
    public void setDetails (String name, String author,  
                            double price, int numPages) {
```

```
        this.name = name;
```

```
        this.author = author;
```

```
        this.price = price;
```

```
        this.numPages = numPages;
```

```
    }
```

```
    public String toString() {
```

```
        return "Book Details: \nName: " + name + " \nAuthor: "
```

```
        author + " \nPrice: " + price + " \nNumber of
```

```
        pages: " + numPages;
```

```
    }
```

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

```
        int Scanner s = new Scanner();
```

```
        int n;
```

```
        System.out.println ("Enter the no. of books:");
```



```
n = nextInt();
```

```
Book[] books = new Book[n];
```

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

```
    books[i] = new Book("Book " + (i+1), "Author " + (i+1), 200 +  
        i * 5, 100 + i * 50);
```

```
}
```

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

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

```
    System.out.println("-----");
```

```
}
```

```
}
```

```
}
```

Output:

Enter the no. of books:

3

Book Details:

Name: Book 1

Author: Author 1

Price: 200.0

Number of pages: 100

Book Details:

Name: Book 2

Author: Author 2

Price: 205.0

Number of pages: 150

Book Details:

Name: Book 3

Author: Author 3

Price: 3000.0

Number of pages: 200

Output →

Rectangle Area: 50

Triangle Area: 9.0

Circle Area: 50.2654

- 4] * 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 them classes contain only one method printArea() that prints the area of the given Shape.

→
abstract class Shape {

int d1;

int d2;

public Shape(int d1, int d2) {

this.d1 = d1;

this.d2 = d2;

}

public abstract void printArea();

}

class Rectangle extends Shape {

public Rectangle(int length, int width) {

super(length, width);

}

public void printArea() {

int area = d1 * d2;

System.out.println("Rectangle Area: " + 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 ("Triangle Area: " + area);  
    }  
}
```

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

```
    public void printArea() {  
        double area = Math.PI * d1 * d2;  
        System.out.println ("Triangle Circle Area: " + Area);  
    }  
}
```

```
public class ShapeMain {  
    public static void main (String[] args) {  
        Rectangle r = new Rectangle (5, 10);  
        rec r.printArea();  
    }  
}
```

```
    Triangle t = new Triangle (3, 6);  
    t.printArea();  
}
```

```
    Circle c = new Circle (4);  
    c.printArea();  
}
```


Name: Book 3

Author: Author 3

Price: 3000.0

Number of pages: 200

Output →

Rectangle Area: 50

Triangle Area: 9.0

Circle Area: 50.2654.

19/2/24

5* 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-acc and Sav-acc to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks:

1. Accept deposit from customer and update the balance.
 2. Display the balance.
 3. Compute and deposit interest.
 4. Permit withdrawal and update the balance.
- Check for the minimum balance, impose penalty if necessary and update the balance.

```
import java.util.Scanner;
class Account {
    String custName;
    long accNum;
    String accType;
    double balance;
```

```
    public Account(String custName, long accNum, String accType,
        double balance) {
        this.custName = custName;
        this.accNum = accNum;
```



```
this.acctype = acctype;
this.balance = balance;
}
```

```
public void deposit(double amount){
    balance += amount;
    System.out.println("Deposit successful. New Balance: " + balance);
}
```

```
public void displayBalance(){
    System.out.println("Balance: " + balance);
}
}
```

```
class CurrAccount extends Account{
    double minBalance;
    double serviceCharge;

    public CurrAccount(String custName, long accNum, String acctype,
        double balance, double minBalance, double serviceCharge){
        super(custName, accNum, acctype, balance);
        this.minBalance = minBalance;
        this.serviceCharge = serviceCharge;
    }
}
```

```
@Override
public void deposit(double amount){
    super.deposit(amount);
}
```

@Override

@Override

```
public void displayBalance() {  
    super.displayBalance();  
}
```

```
public void withdraw(double amount) {  
    if (balance - amount >= minBalance) {  
        balance -= amount;  
        System.out.println("Withdrawal successful. New Balance: " +  
            balance);  
    } else {  
        System.out.println("Insufficient funds. Withdrawal failed.");  
    }  
}
```

```
public void serviceCharge() {  
    balance -= serviceCharge;  
    System.out.println("Service charge imposed. New Balance: " +  
        balance);  
}
```

```
class SrvAccount extends Account {  
    double interestRate;
```

```
    public SrvAccount(String custName, long accNum, String accType,  
        double balance, double interestRate) {  
        super(custName, accNum, accType, balance);  
        this.interestRate = interestRate;  
    }
```


... @Override

```
public void deposit (double amount) {  
    super.deposit (amount);  
}
```

@Override

```
public void displayBalance () {  
    super.displayBalance ();  
}
```

```
public void computeAndDepositInterest () {  
    double interest = balance * interestRate / 100;  
    balance += interest;  
    System.out.println ("Interest computed and deposited.  
        New balance: " + balance);  
}
```

```
public void withdraw (double amount) {  
    if (balance - amount >= 0) {  
        balance -= amount;  
        System.out.println ("Withdrawal successful. New Balance: " +  
            balance);  
    } else {  
        System.out.println ("Insufficient funds. Withdrawal failed.");  
    }  
}
```

```
}
```

```
public class Bank {
```

```
    public static void main (String[] args) {  
        Scanner s = new Scanner (System.in);
```

Date _____
Page _____

```
CurAccount currentAccount = new CurAccount("Arni Khanna",  
123456789, "Current", 1000, 500, 10);
```

```
savAccount savingsAccount = new SavAccount("Neil Khanna",  
987654321, "Savings", 5000, 5);
```

```
System.out.println("1. Current Account");
```

```
System.out.println("2. Savings Account");
```

```
System.out.println("Select Account type: ");
```

```
int choice = s.nextInt();
```

```
switch (choice) {
```

```
    case 1:
```

```
        System.out.println("Current Account Selected.");
```

```
        operateCurAccount  
        CurAccount (currentAccount);
```

```
        break;
```

```
    case 2:
```

```
        System.out.println("Savings Account Selected.");
```

```
        operateSavAccount  
        SavAccount (savingsAccount);
```

```
        break;
```

```
    default:
```

```
        System.out.println("Invalid choice.");
```

```
}
```

```
}
```

```
public static void operateCurrentAccount (CurAccount currentAccount) {
```

```
    Scanner s = new Scanner(System.in);
```

```
    int option;
```

```
    do {
```

```
        System.out.println("1. Deposit");
```

```
        System.out.println("2. Withdraw");
```

```
        System.out.println("3. Display Balance");
```


System.out.println("4. Exit");
System.out.println("Select option: ");
option = s.nextInt();

switch (option) {

case 1:

System.out.print("Enter deposit amount: ");
double depositAmount = s.nextDouble();
currentAccount.deposit(depositAmount);
break;

case 2:

System.out.print("Enter withdrawal amount: ");
double withdrawAmount = s.nextDouble();
currentAccount.withdraw(withdrawAmount);
if (currentAccount.balance < currentAccount.minBalance) {
currentAccount.imposeServiceCharge();
}

break;

case 3:

currentAccount.displayBalance();
break;

case 4:

System.out.println("Exiting current account operations.");
break;

default:

System.out.println("Invalid option.");

}

while (option != 4);

}

public static void operateSavingsAccount(sqrAccount savingsAccount) {
Scanner s = new Scanner(System.in);

int option;

do {

System.out.println("\n1. Deposit ");

System.out.println("\n2. Withdraw \n3. Display Balance
\n4. Compute and Deposit Interest");

System.out.println("\n5. Exit ");

System.out.print("Select option: ");

option = s.nextInt();

switch(option){

case 1:

System.out.println("Enter deposit amount: ");

double depositAmt = s.nextDouble();

savingsAcc.deposit(depositAmt);

break;

case 2:

System.out.print("Enter withdrawal amount: ");

double withdrawAmt = s.nextDouble();

savingsAcc.withdraw(withdrawAmt);

break;

case 3:

savingsAcc.displayBalance();

break;

case 4:

savingsAcc.computeAndDepositInterest();

break;

case 5:

System.out.println("Exiting savings account operations");

break;

default:

System.out.println("Invalid option.");

while (option != 5);

Output:

1. Current Account

2. Savings Account

Select account type: 1

Current Account Selected.

1. Deposit

2. Withdraw

3. Display Balance

4. Exit

Select option: 1

Enter deposit amount: 1500

Deposit successful. New balance: 2500.0

Select option: 3

Balance: 2500.0

Select option: 2

Enter withdrawal amount: 1250

Withdrawal successful. New balance: 1250.0

Select option: 3

Balance: 1250.0

select option: 4

Exiting current account operations.

Select Account type: 2

Savings Account selected.

Select

1. Deposit
2. Withdraw
3. Display Balance
4. Compute and Deposit Interest
5. Exit

Select option: 4

Interest computed and deposited. New balance: 5250.0

Select option: 2

Enter withdrawal amount: 500

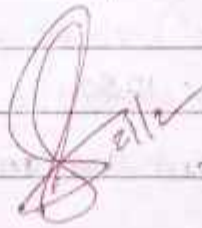
Withdrawal successful. New balance: 4750.0

Select option: 3

Balance: 4750.0

Select option: 5

Exiting Savings account operations.

A stylized handwritten signature in red ink, possibly reading "Sella".

22/12/23

6] * CIE, SEE Package programs.

Student.java

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;

}

}

Internals.java

package CIE;

public class Internals {

int[] internalMarks;

public Internals(int[] internalMarks) {

this.internalMarks = internalMarks;

}

}

External.java

package SEE;

public class External {

String usn;
String name;
int sem;
int[] internalMarks;
int[] externalMarks;

public External(String usn, String name, int sem, int[]
internalMarks, int[] externalMarks){
this.usn = usn;
this.name = name;
this.sem = sem;
this.internalMarks = internalMarks;
this.externalMarks = externalMarks;
}

public String getUsn(){
return usn;
}

public String getName(){
return name;
}

public String getSem(){
return sem;
}

public int[] getInternalMarks(){
return internalMarks;
}

public int[] getExternalMarks(){
return externalMarks;
}

public int getTotalMarks(){


```
int totalInternal=0;
```

```
int totalExternal=0;
```

```
for (int mark: internalMarks){
```

```
    totalInternal += mark;
```

```
}
```

```
for (int mark: externalMarks){
```

```
    totalExternal += mark;
```

```
}
```

```
return totalInternal + totalExternal;
```

```
}
```

```
public double getPercentage(){
```

```
    int totalMarks = getTotalMarks();
```

```
    int maxMarks = 5 * 200;
```

```
    return ((double) totalMarks / maxMarks) * 100;
```

```
}
```

```
public double getCGPA(){
```

```
    return getPercentage() / 10.0;
```

```
}
```

```
}
```

```
# MainProgram.java
```

```
import java.text.DecimalFormat;
```

```
import java.util.Scanner;
```

```
import SEE.External;
```

```
public class MainProgram {
```

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

Scanner s = new Scanner(System.in);

System.out.print("Enter the number of students:");
int n = s.nextInt();

External[] students = new External[n];

for (int i=0; i<n; i++){
System.out.println("\nEnter details for student: " +
(i+1) + ":");

System.out.print("Enter USN:");
String usn = s.next();

System.out.print("Enter Name:");
String name = s.next();

System.out.print("Enter semester:");
int sem = s.nextInt();

System.out.println("Enter Internal Marks for 6 courses");
int[] internalMarks = new int[6];

for (int j=0; j<6; j++){
System.out.print("Course " + (j+1) + ":");
internalMarks[j] = s.nextInt();
}

for (int i=0; i<n; i++){
System.out.println("Enter External Marks for student: " + (i+1) + ":");
int[] externalMarks = new int[5];

for (int j=0; j<5; j++){
System.out.print("Course " + (j+1) + ":");
externalMarks[j] = s.nextInt();
}

}


```
students[i] = new External(usn, name, sem, internalMarks,
                           externalMarks);
```

```
}
```

```
s.close();
```

```
displayFinalMarks(students);
```

```
}
```

```
public static void displayFinalMarks(External[] students) {
    System.out.println("\nFinal Marks, Percentage, and CGPA for
                        Students: ");
```

```
    DecimalFormat df = new DecimalFormat("#.###");
```

```
    for (int i=0; i<students.length; i++) {
```

```
        System.out.println("\nDetails for student " + (i+1) + ": ");
```

```
        System.out.println("USN: " + students[i].getUsn());
```

```
        System.out.println("Name: " + students[i].getName());
```

```
        System.out.println("Semester: " + students[i].getSem());
```

```
        System.out.println("Internal Marks: ");
```

```
        for (int j=0; j<5; j++) {
```

```
            System.out.println("Course " + (j+1) + ": " + st
```

```
            students[i].getInternalMarks(j));
```

```
        }
```

```
    } for (int i=0; i<students.length; i++) {
```

```
        System.out.println("Total Marks: " + students[i].getTotal
                            Marks());
```

```
        System.out.println("Percentage: " + df.format(students[i].
                                                        getPercentage()) + "%");
```

```
        System.out.println("CGPA: " + df.format(students[i].
                                                  getCGPA()));
```

```
}
```

```
}
```

```
}
```

Output:

1) Enter the number of students: 1

Enter details for student 1:

Enter USN: USN001

Enter Name: Anrohi

Enter Semester: 3

Enter Internal Marks for 5 courses:

Course 1: 75

Course 2: 80

Course 3: 75

Course 4: 80

Course 5: 90

Enter External marks for 5 courses:

Course 1: 80

Course 2: 85

Course 3: 80

Course 4: 85

Course 5: 90

Final Marks, Percentage & CGPA for students:

Details for Student 1:

USN: USN001

Name: Anrohi

Semester: 3

Internal Marks:

External Marks:

Total Marks: 820

Percentage: 82%

CGPA: 8.2%

7* Exception Handling Program

```
import java.util.Scanner;
```

```
class WrongAge extends Exception {  
    public WrongAge(String message) {  
        super(message);  
    }  
}
```

```
class Father {  
    int age;
```

```
    public Father(int age) throws WrongAge {
```

```
        if (age < 0) {
```

```
            throw new WrongAge("Age cannot be negative.");  
        }
```

```
        this.age = age;
```

```
    public int getAge() {
```

```
        return age;
```

```
    }  
  
class Son extends Father {
```

```
    int sonAge;
```

```
    public Son(int fatherAge, int sonAge) throws WrongAge {  
        super(fatherAge);
```

```
if (sonAge >= fatherAge) {
    throw new WrongAge("Son's age cannot be greater
    than or equal to the father's age.");
}
```

```
    this.sonAge = sonAge;
}
```

```
public int getSonAge() {
    return sonAge;
}
}
```

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

```
        try {
            System.out.println("Enter Father's Age: ");
            int fatherAge = s.nextInt();
            Father f = new Father(fatherAge);
```

```
            System.out.print("Enter Son's Age: ");
            int sonAge = s.nextInt();
            Son s1 = new Son(father.getAge(), sonAge);
```

```
            System.out.println("Father's Age: " + f.getAge());
            System.out.println("Son's Age: " + s1.getAge());
```

```
            System.out.print("Enter another Son's Age: ");
            int invalidSonAge = s.nextInt();
            int invalidSon = new Son(f.getAge(), invalidSonAge);
```


} catch (WrongAge e) {

System.out.println("Exception caught: " + e.getMessage());

} finally {

s.close();

}

}

}

Output:

Enter Father's Age: 40

Enter Son's Age: 20

Father's Age: 40

Son's Age: 20

Enter another Son's Age: 45

Exception caught: Son's age cannot be greater than or equal to the father's age.

8] Write a program that displays 2 Threads 1 Thread displaying BMS College of Engineering every 10 seconds and another thread displays CSE every 2 seconds

```
class BMSCEThread implements Runnable{
```

```
    Thread t;
```

```
    BMSCEThread(){
```

```
        t = new Thread(this, "BMSThread");
```

```
        System.out.println("CT: " + t);
```

```
        t.start();
```

```
    }
```

```
    public void run(){
```

```
        try{
```

```
            for(int i=5; i>0; i--){
```

```
                while(true){
```

```
                    System.out.println("BMS College of Engineering");
```

```
                    Thread.sleep(10);
```

```
                }
```

```
            }
```

```
            catch (InterruptedException ie){
```

```
                System.out.println("BMSThread Interrupted");
```

```
            }
```

```
            System.out.println("BMSThread is quitting");
```

```
        }
```

```
    }
```

```
class CSEThread{
```

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

```
        new BMSCEThread();
```

```
        try{
```

```
            for (int i=5; i>0; i--){
```

```
                System.out.println("CSE");
```



```

        Thread.sleep(2);
    }
}

catch (InterruptedException ie) {
    System.out.println ("CSE Thread is Interrupted");
}
System.out.println ("CSE Thread is quitting");
}
}

```

Output:

CT: Thread (21, BMSThread, 5, main)

CSE

BMS COLLEGE OF ENGINEERING

CSE

CSE

CSE

CSE

BMS COLLEGE OF ~~THREADING~~ ENGINEERING

BMS CSE Thread is quitting

BMS COLLEGE OF ENGINEERING

BMS COLLEGE OF ENGINEERING

BMS COLLEGE OF ENGINEERING

BMSThread is quitting

8/5/2

9] 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 or Num2 were not an integer, the program would throw an Arithmetic Exception. Display the exception in a message dialog box.

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

```
public class IntegerDivisionAWT extends Frame implements
    ActionListener {
```

```
    private TextField num1Field, num2Field, resultField;
    private Button divideButton;
```

```
    public IntegerDivisionAWT() {
```

```
        setTitle("IntegerDivision");
```

```
        setSize(300, 200);
```

```
        setLayout(new FlowLayout());
```

```
        addWindowListener(new WindowAdapter() {
```

```
            public void windowClosing(WindowEvent e) {
                dispose();
```

```
            }
```

```
        });
```

```
        label num1label = new Label("Num 1:");
```

```
        num1Field = new TextField(10);
```

```
        label num2label = new Label("Num 2:");
```

```
        num2Field = new TextField(10);
```

```
        label resultlabel = new Label("Result:");
```

```
        resultField = new TextField(10);
```

```
        resultField.setEditable(false);
```

```
        divideButton = new Button("Divide");
```



```
divideButton.addActionListener(this);  
  
add(num1Label);  
add(numField);  
add(num2Label);  
add(num2Field);  
add(resultLabel);  
add(resultField);  
add(divideButton);  
  
setVisible(true);  
}  
  
public void actionPerformed(ActionEvent e){  
    if (e.getSource() == divideButton){  
        try{  
            int num1 = Integer.parseInt(numField.getText());  
            int num2 = Integer.parseInt(num2Field.getText());  
            if (num2 == 0){  
                showMessageDialog("cannot divide by zero");  
                return;  
            }  
            int result = num1 / num2;  
            resultField.setText(String.valueOf(result));  
        }  
        catch (NumberFormatException ex){  
            showMessageDialog("Please Enter valid integers for  
            Num1 and Num2");  
        }  
    }  
}
```

```

private void ShowMessageDialog (String Message) {
    Dialog dialog = new Dialog (-this, "Error", true);
    dialog.setSize (400, 400);
    dialog.setLayout (new FlowLayout());
    Label label = new Label (message);
    dialog.add (label);
    Button okButton = new Button ("OK");
    okButton.addActionListener (new ActionListener() {
        public void actionPerformed (ActionEvent e) {
            dialog.dispose();
        }
    });
    dialog.add (okButton);
    dialog.setVisible (true);
}

public static void main (String [] args) {
    new IntegerDivisionAWT();
}
}

```

Output

Num1: Num2: Result:

En/2

10]

* Report

AWT - "ABSTRACT WINDOW TOOLKIT"

- The AWT is a package in Java which provides classes to create and manage graphical user interfaces (GUIs).
- AWT includes various components, event handlers, layout managers and other utilities.

Components:-

1. Frame:

A frame is a window with title & border. used as a main window in which other components are added.
Syntax: class IntegerDivision extends Frame;

2. Dialog:

- It takes some form of input from the user.
- Used to display modal dialog to interact with user
- Used to confirm actions, prompt the user & display messages.

3. Textfield:

- It is a single line text displayed.
- Used for guiding the user to accept input.

4. Label:

- Display area for short text.
- Display static text.
- Prompts the user to enter text/input.
- eg. Num1:

5. Button:

- Triggers the actions when clicked
- Used for performing specific actions
- Used for submitting forms, confirming etc.

6. CheckBox:

- Component which can be checked or unchecked.
- used to enable or disable.
- used in the form with multiple choices.

* Event Handling

1. Events:

- ~~Events~~ are generated by user actions.
- used for handling user input, such as mouse clicks, keypress or window events.
- Events are processed by event listeners attached to the relevant components.

2. Event Listener:

- Object that receives and handles events.
- This is for the implementation of event driven behavior in GUIs.
- Event Listeners are registered with specific components to listen for particular type events.

3. Action Event:

- Event that indicates, that a component defined action occurred.
- used for handling user actions.
- Action generated when user clicks on component like button, checkbox, etc.

4. Action Listener:

- It receives action events.
- It responds to the user action.

5. WindowEvent

- Event indicating change to the state of a window.
- Handles window related events such as window opening, closing, resizing etc.

6. WindowListener

- It receives window events.
- Used to implement methods to handle window related events.

* Layout Management

1. Flow Layout

- Arranges components in a centered flow one after another.
- Used for creating forms, etc.

2. Border Layout

- Layout manager that divides the container into 5 regions: north, south, east, west & center.
- Used for arranging components at the edges & center of the container.

3. Layout Manager

- Responsible for arranging components in container.
- Control size & position of components.

