B.M.S. COLLEGE OF ENGINEERING

(Autonomous college under VTU)

Bull Temple Rd, Basavanagudi, Bengaluru, Karnataka 560019 2024-2026

Department of Computer Applications

Report is submitted for fulfillment of AAT work in the subject

JAVA PROGRAMMING ("MMC204")

BY

YASHWANTH C H

(1BM24MC114)

Under the Guidance Prof.
R.V. Raghavendra Rao

(Assistant Professor)

CONTENTS

	6. Design a class to represent account, include the following members. Data Members: • Name of depositor—string • Account Number—int • Type of Account—boolean • Balance amount—double Methods • To assign initial values (using constructor) • To deposit an amount after checking balance and minimum balance 50. • To display the name and balance.	
AAT-2.	**1.** Write a program called SumAverageRunningInt to produce the sum of 1, 2, 3, ..., to 100. Store 1 and 100 in variables lowerbound and upperbound, so that we can change their values easily. Also compute and display the average. The output shall look like: The sum of 1 to 100 is 5050 The average is 50.5 **2.** Write a program called HarmonicSum to compute the sum of a harmonic series, as shown below, where n=50000. The program shall compute the sum from left-to-right as well as from the right-to-left. Are the two sums the same? Obtain the absolute difference between these two sums and explain the difference. Which sum is more accurate? **3.** Write a program called Fibonacci to print the first 20 Fibonacci numbers F(n), where F(n)=F(n–1) +F(n–2) and F (1) =F (2) =1. Also compute their average. The output shall look like: The first 20 Fibonacci numbers are: 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 The average is 885.5 4. Write a program called ExtractDigits to extract each digit from an int, in the reverse order. For example, if the int is 15423, the output shall be "3 2 4 5 1", with a space separating the digits.	24-31

AAT-3.

1. The progressive income tax rate is mandated as follows:

Taxable Income Rate (%)

First $20,000 0

Next $20,000 10

Next $20,000 20

The remaining 30

For example, suppose that the taxable income is $85000, the income tax payable is

$20000*0% + $20000*10% + $20000*20% + $25000*30%.

Write a program called IncomeTaxCalculator that reads the taxable income (in int). The

program shall calculate the income tax payable (in double); and print the result rounded to 2

decimal places. Program shall repeat the calculation until user enter -1.

For example,

Enter the taxable income: $41234

The income tax payable is: $2246.80

Enter the taxable income: $-1

bye!

2. Both the employer and the employee are mandated to contribute a certain percentage of the

employee's salary towards the employee's pension fund. The rate is tabulated as follows:

Employee's Age	Employee Rate (%)	Employer Rate (%)
55 and below	20	17
above 55 to 60	13	13
above 60 to 65	7.5	9
above 65	5	7.5

However, the contribution is subjected to a salary ceiling of $6,000. In other words, if an

employee earns $6800, only $6000 attracts employee's and employer's contributions, the

remaining $800 does not.

Write a program called PensionContributionCalculator that reads the monthly salary and age

(in int) of an employee. Your program shall calculate the employee's, employer's and total

contributions (in double); and print the results rounded to 2 decimal places.

For example,

32-41

	Enter the monthly salary: $3000 Enter the age: 30 The employee's contribution is: $600.00 The employer's contribution is: $510.00 The total contribution is: $1110.00 **3.** Write a program called ReverseString, which prompts user for a String, and prints the reverse of the String by extracting and processing each character. The output shall look like: Enter a String: abcdef The reverse of the String "abcdef" is "fedcba" **4.** Write a program called CountVowelsDigits, which prompts the user for a String, counts the number of vowels (a, e, i, o, u, A, E, I, O, U) and digits (0-9) contained in the string, and prints the counts and the percentages (rounded to 2 decimal places). For example, Enter a String: testing12345 Number of vowels: 2 (16.67%) Number of digits: 5 (41.67%) **5.** Write a program called Bin2Dec to convert an input binary string into its equivalent decimal number. Your output shall look like: Enter a Binary string: 1011 The equivalent decimal number for binary "1011" is: 11	
AAT-4.	**1.** Provided that you have a given number of small rice bags (1 kilo each) and big rice bags (5kilos each), write a method that returns true if it is possible to make a package with goal kilos of rice. **2.** Write a class called Employee, which models an employee with an ID, name and salary, is designed as shown in the following class diagram. The method raiseSalary(percent) increases the salary by the given percentage. Write the Employee class. 	42-49

3. Write a class called Author (as shown in the class diagram) is designed to model a book's author. It contains:

• Three private instance variables: name (String), email (String), and gender (char of either 'm' or 'f');

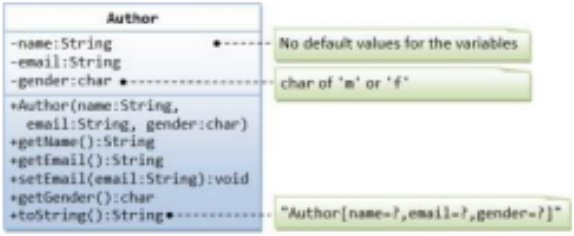
• One constructor to initialize the name, email and gender with the given values;

public Author (String name, String email, char gender) {. .... }

• public getters/setters: getName(), getEmail(), setEmail(), and getGender();

• A toString() method that returns "Author[name=?,email=?,gender=?]", e.g.,

"Author[name=Tan Ah Teck,email=ahTeck@somewhere.com,gender=m]".



4. In this exercise, a subclass called Cylinder is derived from the superclass Circle as shown in the

class diagram. Study how the subclass Cylinder invokes the superclass' constructors (via super() and super(radius)) and inherits the variables and methods from the superclass Circle.

• The subclass Cylinder inherits getArea() method from its superclass Circle. Try overriding the getArea() method in the subclass Cylinder to compute the surface area(=2π×radius×height + 2×base-area) of the cylinder instead of base area.

o That is, if getArea() is called by a Circle instance, it returns the area.

oIf getArea() is called by a Cylinder instance, it returns the surface area of the cylinder.

• If you override the getArea() in the subclass Cylinder, the getVolume() no longer works. This is because the getVolume() uses the overridden getArea() method found in the same class. Fix the getVolume().

	 **5.** Define a class CARRENTAL with the following details : • Class Members are: CarId of int type, CarType of string type and Rent of float type. • Define GetCar() method which accepts CarId and CarType. • GetRent() method which return rent of the car on the basis of car type, i.e. Small Car= 1000, Van = 800, SUV = 2500 • ShowCar() method which allow user to view the contents of cars i.e. id, type and rent.	
AAT-5.	**1.** Create an abstract class "BankAccount" with abstract methods "deposit()" and "withdraw()". Implement two subclasses "SavingsAccount" and"CheckingAccount" which extend "BankAccount" and implement the abstract methods. Create a"Customer" class which contains a list of "BankAccount" objects. Add methods to the "Customer" class to display account balances, deposit/withdraw money, etc. Create objects of all classes and test their behavior. **2.** An abstract class called "Marks" is needed to calculate the percentage of marks earned by students A in three subjects (with each subject out of 100) and student B in four subjects (with each subject out of 100). This class must contain the abstract method "getPercentage," which two other classes, "A" and "B," will inherit. The method "getPercentage," which provides the percentage of students, is shared by classes "A" and "B." The constructor of class 'A' will accept the marks obtained in three subjects as its parameters and the constructor of class 'B' will accept the marks obtained in four subjects as its parameters. To test the implementation, objects for both the classes need to be created and the percentage of marks for each student should be printed. **3.** Write a Java program using a function root to calculate and display all roots of the quadratic AX2 + BX + C = 0. **4.** Write a program to fi nd the volume of a box that has its sides w, h, d as width, height, and depth, respectively. Its volume is v = w * h * d and also fi nd the surface area given by the formula s = 2(wh + hd + dw). **5.** A class weight is having a data member pound, which will have the weight in pounds. Using a conversion function, convert the weight in pounds to weight in kilograms which is of double type. Write a program to do this. 1 pounds =1 kg/0.453592	50-63

	Use default constructor to initial assignment of 1000 pounds.	
AAT-6.	**1.** Write a package called Clear, it contains one public method clrscr() to clear the screen, import the package and use it in another programs. Add another public method starline(). It prints the line of 15 starts. **2.** Define an exception called " No Equal Exception " that is thrown when a float value is not equal to 3.14. Write a program that uses the above user defi ned exception. **3.** Write a java program using threads to simulate traffic lights switch between Red, Green, and Yellow with fixed delays.	46-50
AAT-7.	**1.** Write a package called Clear, it contains one public method clrscr() to clear the screen, import the package and use it in another programs. Add another public method starline(). It prints the line of 15 stars. **2.** Write a program in Java. A class Teacher contains two fi elds, Name and Qualifi cation. Extend the class to Department, it contains Dept. No and Dept. Name. An interface named as College contains one fi eld Name of the college. Using the above classes and Interface get the appropriate information and display it. **3.** Write and run the following Java program that does the following: a) Declare a string object named s1 containing the string "Object Oriented Programming- Java 5". b) Print the entire string. c) Use the length() method to find the length of the string. d) Use the charAt() method to find the first character in the string. e) Use charAt() and length() methods to print the last character in the string. f) Use the indexOf() and the substring() method to print the first word in the String	51-55

AAT-8.	**1.** Write a program that accepts a name list of five students from the command line and store in a vector. **2.** Write and run the following Java program that does the following: a) Declare a string object named s1 containing the string "Object Oriented Programming-Java 5". b) Print the entire string. c) Use the length() method to find the length of the string. d) Use the charAt() method to find the first character in the string. e) Use charAt() and length() methods to print the last character in the string. f) Use the indexOf() and the substring() method to print the first word in the String. **3.** Define an exception called " No Equal Exception " that is thrown when a fl oat value is not equal to 3.14. Write a program that uses the above user defi ned exception. **4.** Write a java program using threads to simulate traffic lights switch between Red, Green, and Yellow with fixed delays. **5.** Write a Java program to accept two parameters on the command line. If there are no command line arguments entered, the program should print the error message and exit. The program should check whether the first fi le exists and if it is an ordinary file. If it is so, then the content is copied to the second file.	75-82

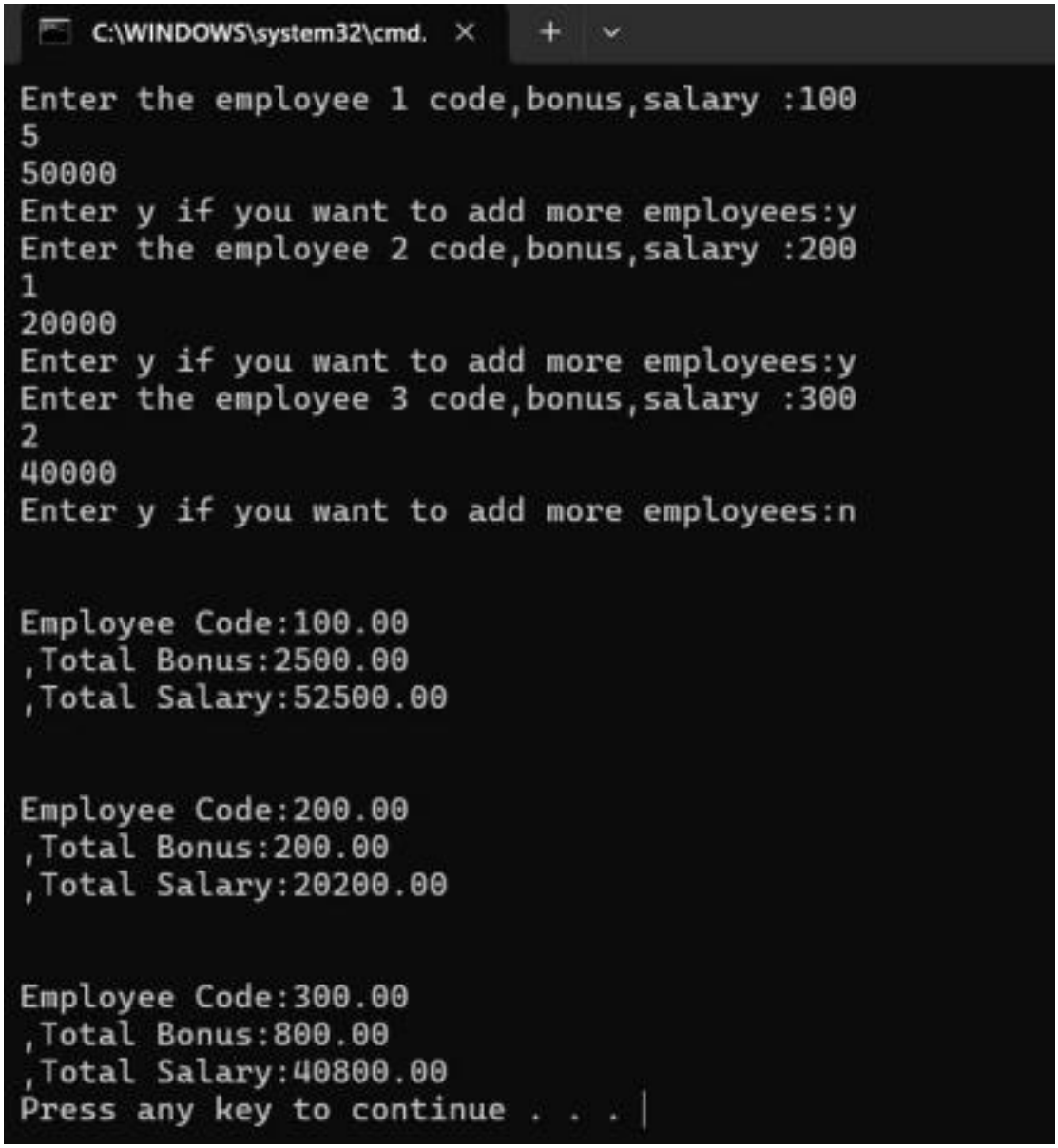
<table>
<tr><td>AAT-9.</td><td>1. Write a Java program to check whether the file is readable, writable and hidden.
2. Write a program using Lambda expression to filter the Employee objects. Each employee has
a name, department, and salary.
• Employees with salary > 50,000
• Employees in the "IT" department
3. You're building a system that handles different types of items in an online store. You want to
create an Inventory<T> class that can store any type of item (books, electronics, etc.).
4. Write a program using autoboxing to calculate the discounted prices for a shopping cart. The
system stores prices as Double, but the calculations are done using double.</td><td>83-88</td></tr>
<tr><td>AAT-10.</td><td>1. Write a program to create a calculator using event handling.
2. Create a menu is a frame as shown below:
Books Language Film Search
• The Books menu has the following items: C, C++, Java, Oracle, VB, and ASP.
• The language menu consists of French, German, English, Tamil, and Hindi.
• The File menu has the following: Titanic, Jurassic Park, Tomorrow never Dies, Jeans, and Slumdog.
• The search engine consists of Yahoo, Hotmail, Sify, Google, and Lycos.
3. Write a Java program to display the different prices of the books using the choice object.
4. Write Java program to display the different car names using list object.
5. Write a Java program to display the different IIT's names in India using the choice object and list object.
6. Write a Java program to display the following 3 × 3 magic square (total = 15) using JTable:
<table><tr><td>2</td><td>9</td><td>4</td></tr><tr><td>7</td><td>5</td><td>3</td></tr><tr><td>6</td><td>1</td><td>8</td></tr></table>
7. You're building a system that handles different types of items in an online store. You want tocreate an Inventory<T> class that can store any type of item (books, electronics, etc.).</td><td>89-100</td></tr>
</table>

AAT 1

```
/*1. The following table shows the employees code and the percentageof bonus
for the value of basic pay.
Employee code Bonus
 100 5
 200 1
 300 2
 400 25
 */

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
class Calculate{
        double code;
        double bonus;
        double salary;
Calculate(double code,double bonus,double salary){ this.code=code;
                this.bonus=bonus;
                this.salary=salary;
        }
         List<Double> calc(){
double totbonus=(salary*bonus)/100;
                double totsal=salary+totbonus;
List<Double> saldet=new ArrayList<>();
                saldet.add(totbonus);
                saldet.add(totsal);
                return saldet;
   }
}

public class Prg1{
public static void main(String args[]){
Scanner sc=new Scanner(System.in);
                char menu='y';
List<Calculate> obs=new ArrayList<>();
                int count=0;
                while (true){
if (menu=='y'){
count++;
System.out.printf("Enter the employee %d
```

```
code,bonus,salary :",count);
double code=sc.nextDouble();
double bonus=sc.nextDouble();
double salary=sc.nextDouble();
sc.nextLine();
Calculate c=new Calculate(code,bonus,salary);
obs.add(c);
System.out.print("Enter y if you want to add
more employees:");
menu=sc.next().charAt(0);
                        }
else { break;}
                }
                for (Calculate e:obs){
List<Double> results = e.calc();
double totalBonus = results.get(0);
double totalSalary = results.get(1);
System.out.println("\n");
System.out.printf("Employee Code:%.2f%n,Total
Bonus:%.2f%n,Total Salary:%.2f%n",e.code,totalBonus,totalSalary); }
        }
}
```

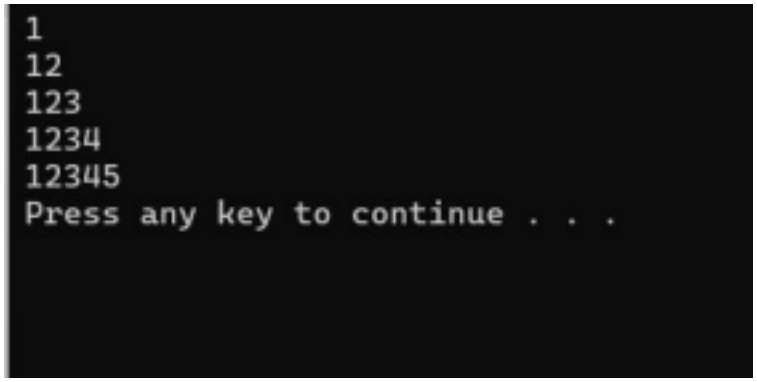
Output:-




```
/* Write a program to display the following output using for loop: ( a)
      1
      1 2
      1 2 3
      1 2 3 4
      1 2 3 4 5
 */

 class Pattern1{
 public static void main(String args[]){ int i;
                int j;
                for(i=1;i<=5;i++){
for(j=1;j<=i;j++){
System.out.print(j);
                        }
```

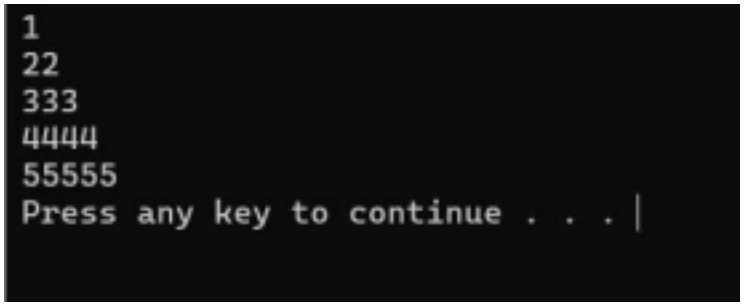
```
                System.out.println();
            }
        }
}
```

Output:-



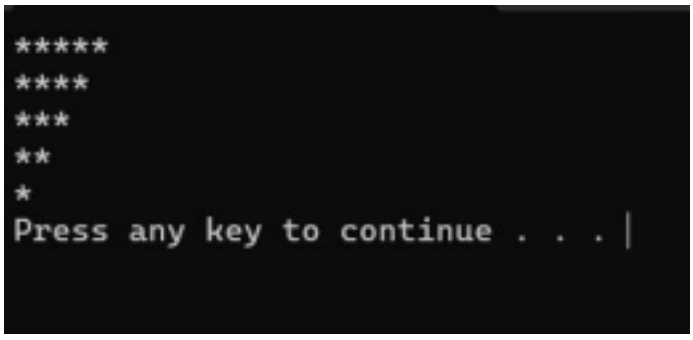

```
/* ( b)
      1
      2 2
      3 3 3
      4 4 4 4
      5 5 5 5 5
 */
class Pattern2{
public static void main(String args[]){ int i,j;
            for (i=1;i<=5;i++){
for (j=1;j<=i;j++){
System.out.print(i);
                }
                System.out.println();
            }
        }
}
```

Output:-



```
/* ( c)
      * * * * *
      * * * *
      * * *
      * *
      *
 */
class Pattern3{
public static void main(String args[]){ int i,j;
                for(i=5;i>=1;i--){
for(j=1;j<=i;j++){
System.out.print("*");
                        }
                        System.out.println();
                }
        }
}
```

Output:-




```
/* 4. Write and run a Java program that inputs three names and print themin
their alphabetical order.
*/

import java.util.Scanner;
class Str{
        String s1,s2,s3;
        Str(String s1,String s2,String s3){
                this.s1=s1;
                this.s2=s2;
                this.s3=s3;
        }
        void comp(){
                String f="",s="",t="";
        if(s1.compareTo(s2)<=0 && s1.compareTo(s3)<=0){ f=s1;
```

```
if(s2.compareTo(s3)<=0){
                        s=s2;
                        t=s3;
                }
                else{
                        s=s3;
                        t=s2;
                }
        }
    else if(s2.compareTo(s1)<=0 && s2.compareTo(s3)<=0){f=s2;
if(s1.compareTo(s3)<=0){
                        s=s1;
                        t=s3;
                }
                else{
                        s=s3;
                        t=s1;
                }
        }
    else if(s3.compareTo(s1)<=0 && s3.compareTo(s2)<=0){f=s3;
if(s1.compareTo(s2)<=0){
                        s=s1;
                        t=s2;
                }
                else{
                        s=s2;
                        t=s1;
                }
        }
System.out.printf("\nFirst String is: %s\n",f);
System.out.printf("Second String is: %s\n",s);
System.out.printf("Third String is: %s\n",t);
    }
}

public class Prg4{
public static void main(String args[]){
Scanner sc=new Scanner(System.in);
        String s1,s2,s3;
        outerLoop:
        while (true){
                String cont;
```

```
                    System.out.print("\nEnter the Strings 1,2 and 3 indifferent
lines:- ");
                    s1=sc.nextLine();
                    s2=sc.nextLine();
                    s3=sc.nextLine();
Str obj=new Str(s1,s2,s3);
                    obj.comp();
System.out.print("\nDo u want to Continue:");
                    cont=sc.nextLine();
while(true){
if(cont.equalsIgnoreCase("Yes") ||
cont.equalsIgnoreCase("y")){
break;
}
else if(cont.equalsIgnoreCase("no") ||
cont.equalsIgnoreCase("n")){
break outerLoop;
}
else{
System.out.print("\nEnter proper Input:");
cont=sc.nextLine();
}
                    }
                }
        }
}
```

Output:-

```
Enter the Strings 1,2 and 3 in different lines:- yash
sahana
aditya

First String is: aditya
Second String is: sahana
Third String is: yash

Do u want to Continue:y

Enter the Strings 1,2 and 3 in different lines:- mango
apple
banana

First String is: apple
Second String is: banana
Third String is: mango

Do u want to Continue:jsh

Enter proper Input:sdj

Enter proper Input:no
Press any key to continue . . .
```

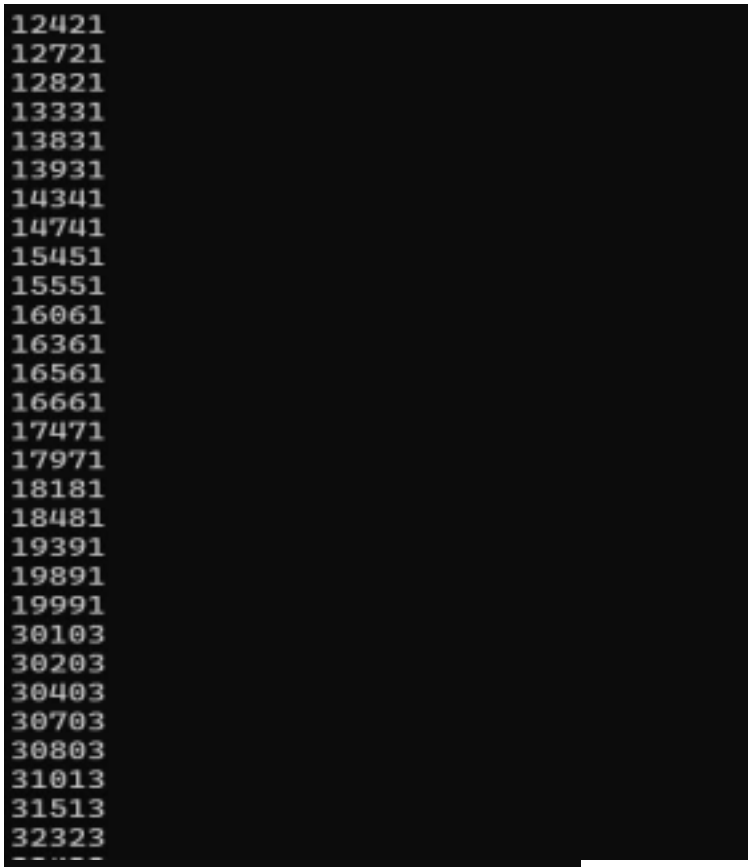
/*5. A number is said to be palindrome if it is invariant under reversion; that is, the number is the same if its digits are reversed. For example, 3456543 is palindromic.
Write a program that checks each of the first 10,000 prime numbers andprints those that are palindromic.
*/

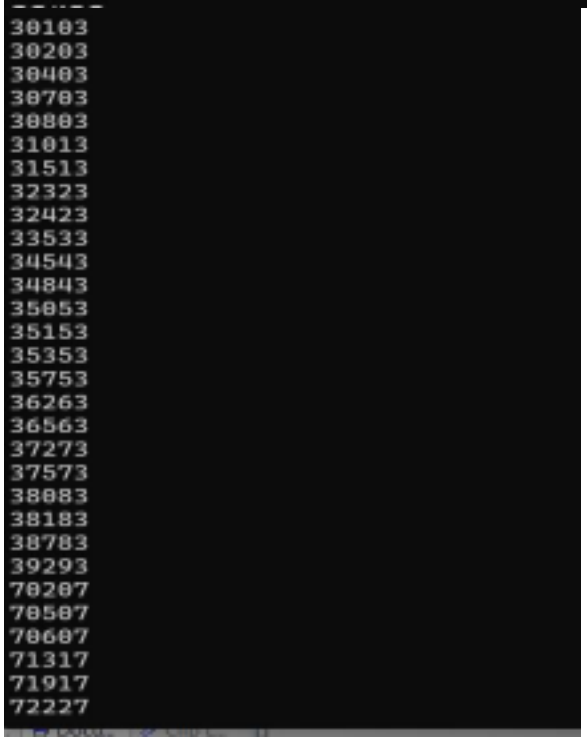
```
public class Prg5 {

   public static void main(String[] args) {
NumberChecker checker = new NumberChecker();

       System.out.println("Finding the first 10,000 prime palindromic numbers...");

       int countOfPrimesFound = 0;
       int numberToCheck = 2;
       while (countOfPrimesFound < 10000) {
if (checker.isPrime(numberToCheck)) {
            countOfPrimesFound++;

          if (checker.isPalindrome(numberToCheck)) {
             System.out.println(numberToCheck);
             }
```

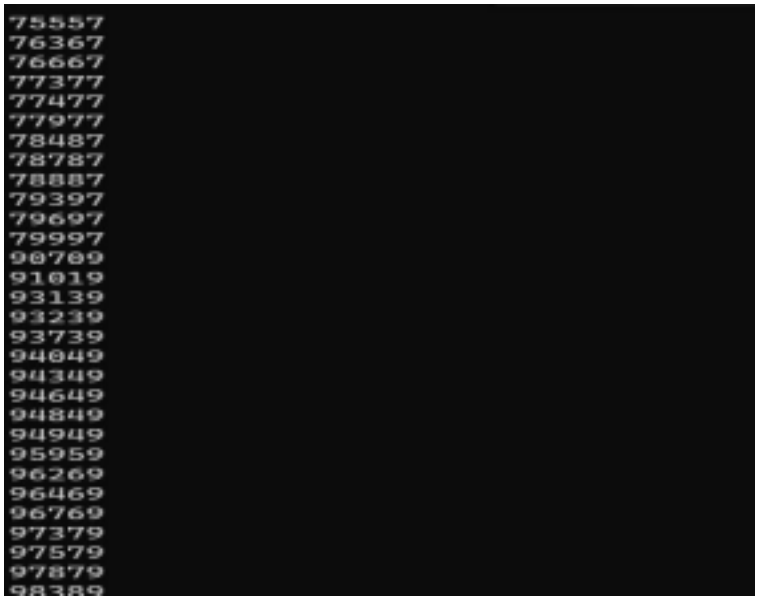
```
            }
            numberToCheck++;
        }

        System.out.println("\nFinished checking the first 10,000 prime numbers
for palindromes.");
    }
}

class NumberChecker {

    public boolean isPrime(int number) {
        if (number <= 1) {
            return false;
        }
        if (number == 2) {
            return true;
        }
        if (number % 2 == 0) {
            return false;
        }

    for (int i = 3; i <= Math.sqrt(number); i += 2) { if (number % i
        == 0) {
                return false;
            }
        }

        return true;
    }

    public boolean isPalindrome(int number) {
        if (number < 0) {
            return false;
        }
        int originalNumber = number;
        int reversedNumber = 0;

        while (number > 0) {
            int digit = number % 10; reversedNumber =
            reversedNumber * 10 + digit; number /= 10;
        }
```

```
        return originalNumber == reversedNumber;
    }
}
```

Output:-


12421
12721
12821
13331
13831
13931
14341
14741
15451
15551
16061
16361
16561
16661
17471
17971
18181
18481
19391
19891
19991
30103
30203
30403
30703
30803
31013
31513
32323


30103
30203
30403
30703
30803
31013
31513
32323
32423
33533
34543
34843
35053
35153
35353
35753
36263
36563
37273
37573
38083
38183
38783
39293
70207
70507
70607
71317
71917
72227



/*6. Design a class to represent account, include the following members. Data Members:

Name of depositor—string
Account Number—int
Type of Account—boolean
Balance amount—double
Methods
To assign initial values (using constructor)
To deposit an amount after checking balance and minimumbalance50. To display the name and balance.
*/

```
import java.util.*;

class Account {
    String name;
    int accNo;
    boolean accType;
    double balance;

public Account(String name, int accNo, boolean accType) { this.accNo =
    accNo;
        this.accType = accType;
        this.balance = 0.0;
        this.name = name;
    }
    public void deposit(double depo) {
        balance = balance + depo;
        if (balance >= 50)
```

```
System.out.println("deposited :" + depo);
        else
            System.out.println("please maintain a min balance of Rs.50"); }

    public void display() {
        String type;
        if (accType)
            type = "Savings Account";
        else
            type = "Current Account";
        System.out.println("name of account holder: " + name);
     System.out.println("account type :" + type + "account no: " +accNo);
     System.out.println("account balance: " + balance); }
}



public class Prg6 {
    public static void main(String[] args) {
        String name;
        int accNo;
        boolean accType;
        char cont = 'Y';
        Scanner in = new Scanner(System.in);
        System.out.println("welcome!");

        while (cont == 'Y' || cont == 'y') {
System.out.println("enter name of account holder");
            name = in.nextLine();
            System.out.println("enter acc type:");
            System.out.println("Enter \"true\" if Savings, \"false\" if Current
Account");
            accType = in.nextBoolean();
            System.out.println("enter account number");
            accNo = in.nextInt();
            in.nextLine(); // consume the newline after nextInt()

            Account ob = new Account(name, accNo, accType);

            System.out.println("Do you want to deposit amount? Y/N");
            char a = in.next().charAt(0);

            while (a == 'y' || a == 'Y') {
```

```
System.out.println("enter deposit amount:");
            double deposit = in.nextDouble();
            ob.deposit(deposit);
            System.out.println("Do you want to deposit amount? Y/N"); a =
            in.next().charAt(0);
          }

          ob.display();
          System.out.println("Do you want to continue? enter Y/y to continue...");
          cont = in.next().charAt(0);
in.nextLine(); // consume the leftover newline
       }
    }
}
```

Output:-

```
enter acc type:
Enter "true" if Savings, "false" if Current Account
true
enter account number
123
Do you want to deposit amount? Y/N
999
name of account holder: yash
account type :Savings Accountaccount no: 123
account balance: 0.0
Do you want to continue? enter Y/y to continue...
y
enter name of account holder
sahana
enter acc type:
Enter "true" if Savings, "false" if Current Account
false
enter account number
321
Do you want to deposit amount? Y/N
y
enter deposit amount:
49
please maintain a min balance of Rs.50
Do you want to deposit amount? Y/N
n
name of account holder: sahana
account type :Current Accountaccount no: 321
account balance: 49.0
Do you want to continue? enter Y/y to continue...
```

AAT 2

/*5. Write a program called SumAverageRunningInt to produce the sum of 1, 2, 3, ..., to 100. Store 1 and 100 in variables lowerbound and upperbound, so that we can change their values easily. Also compute and display the average.

The output shall look like: The sum of 1 to 100 is 5050 The average is 50.5*/

```
/* Calculator.java*/

class Calculator {

int lb, ub;


Calculator(int lb, int ub) {

this.lb = lb;

this ub = ub;

}


int sum() {

int s = 0;

for (int i = lb; i <= ub; i++) {

s += i;

}

return s;
}


double avg() {

return (double) sum() / (ub - lb + 1);

}
```

```
}
```

```
/*Main.java*/
public class Main {
public static void main(String[] args) {
int lowerbound = 1;
int upperbound = 100;

Calculator calc = new Calculator(lowerbound, upperbound);

System.out.println("The sum of " + lowerbound + " to " + upperbound
+ "  is " + calc.sum());
System.out.println("The average is " + calc.avg());
try {
System.in.read();
} catch (Exception e) {}
}
}
```

Output:

```
The sum of 1 to 100 is 5050
The average is 50.5
Press any key to continue . . .
```

/* 6. Write a program called HarmonicSum to compute the sum of a harmonic series, as shown below, where n=50000. The program shall compute the sum from left-to-right as well as from the right-to-left. Are the two sums the same? Obtain the absolute difference between these two sums and explain the difference. Which sum is more accurate?

$$Harmonic(n) = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$ */

```
/* HarmonicSum.java*/

public class HarmonicSum {

public static void main(String[] args) {

int n = 50000;

double leftToRightSum = computeLeftToRight(n);

double rightToLeftSum = computeRightToLeft(n);


System.out.println("Harmonic Sum (Left-to-Right): " +
leftToRightSum);  System.out.println("Harmonic Sum (Right-to-
Left): " + rightToLeftSum);


double absDifference = Math.abs(leftToRightSum -
rightToLeftSum);  System.out.println("Absolute Difference: " +
absDifference);


// Explanation of accuracy

System.out.println("\nExplanation:");

System.out.println("The Right-to-Left sum is generally more accurate.");

System.out.println("This is because smaller numbers are added
first,  reducing cumulative rounding errors.");

}


// Method to compute harmonic sum from left to right

private static double computeLeftToRight(int n) {
double sum = 0.0;

for (int i = 1; i <= n; i++) {

sum += 1.0 / i;

}

return sum;

}
```

```
// Method to compute harmonic sum from right to left
private static double computeRightToLeft(int n) {
double sum = 0.0;
for (int i = n; i >= 1; i--) {
sum += 1.0 / i;
}
return sum;
}
}
```

Output:

```
Harmonic Sum (Left-to-Right): 11.512925464970228
Harmonic Sum (Right-to-Left): 11.512925464970228
Absolute Difference: 0.0

Explanation:
The Right-to-Left sum is generally more accurate.
This is because smaller numbers are added first, reducing cumulative rounding errors.

Press any key to continue . . .
```

Explanation: In most cases, the two sums — one computed from left-to-right and the other from right-to-left — will be very close but not exactly the same due to the limitations of floating-point arithmetic in Java (and most programming languages). Floating-point numbers like `double` have limited precision (typically about 15 decimal digits), and when many small fractional values are added together repeatedly, tiny rounding errors can accumulate. These errors tend to occur more severely when adding smaller numbers to a large accumulated sum, which typically happens in the left-to-right approach. On the other hand, the right-to-left method starts by adding the smallest terms first, which helps preserve numerical accuracy because similar-sized small numbers are summed before larger ones are introduced. As a result, the right-to left summation is generally considered more accurate than the left-to-right approach for computing harmonic sums. The absolute difference between the two sums may be extremely small (often on the order of 1e-14 or less) for n = 50000, but it becomes more noticeable as n

increases, highlighting the impact of the order of operations in floating-point computations.

/*7. Write a program called Fibonacci to print the first 20 Fibonacci numbers F(n), where F(n)=F(n–1) +F(n–2) and F (1) =F (2) =1. Also compute their average. The output shall look like: The first 20 Fibonacci numbers are: 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 The average is 885.5*/

```
/* Fibonacci.java*/

class Fibonacci {

int n = 20;

long[] fib = new long[n];


void generate() {

fib[0] = 1;

fib[1] = 1;


for (int i = 2; i < n; i++) {

fib[i] = fib[i - 1] + fib[i - 2];
}

}


double avg() {

long sum = 0;

for (long f : fib) {

sum += f;

}
```

```
return (double) sum / n;
}

void print() {
System.out.println("The first 20 Fibonacci numbers
are:");  for (int i = 0; i < n; i++) {
System.out.print(fib[i]);
if (i < n - 1) System.out.print(" ");
}
System.out.println();
System.out.printf("The average is %.1f\n",
avg());  }
}

/* Main.java*/
public class Main {
public static void main(String[] args) {
Fibonacci f = new Fibonacci();
f.generate();
f.print();
try {
System.in.read();
} catch (Exception e) {}
}
}
```

Output:

```
The first 20 Fibonacci numbers are:
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765
The average is 885.5
Press any key to continue . . .
```

/*8. Write a program called ExtractDigits to extract each digit from an int, in the reverse order. For example, if the int is 15423, the output shall be "3 2 4 5 1", with a space separating the digits.*/

```
/*DigitExtractor.java*/

import java.util.Scanner;


class DigitExtractor {

int num;


void input() {

Scanner sc = new Scanner(System.in);

System.out.print("Enter an integer: ");

num = sc.nextInt();

}
String reverseDigits() {

StringBuilder rev = new
StringBuilder();  int n = num;


while (n > 0) {

rev.append(n % 10).append(" ");

n /= 10;

}
```

```
return rev.toString().trim(); // Remove trailing
space  }
}
```

```
/*Main.java*/

public class Main {

public static void main(String[] args) {

DigitExtractor d = new
DigitExtractor();  d.input();
String res = d.reverseDigits();


System.out.println("Digits in reverse: " + res);


try {

System.in.read();

} catch (Exception e) {}

}
}
```

Output:

```
Enter an integer: 15423
Digits in reverse: 3 2 4 5 1
Press any key to continue . . .
```

AAT 3

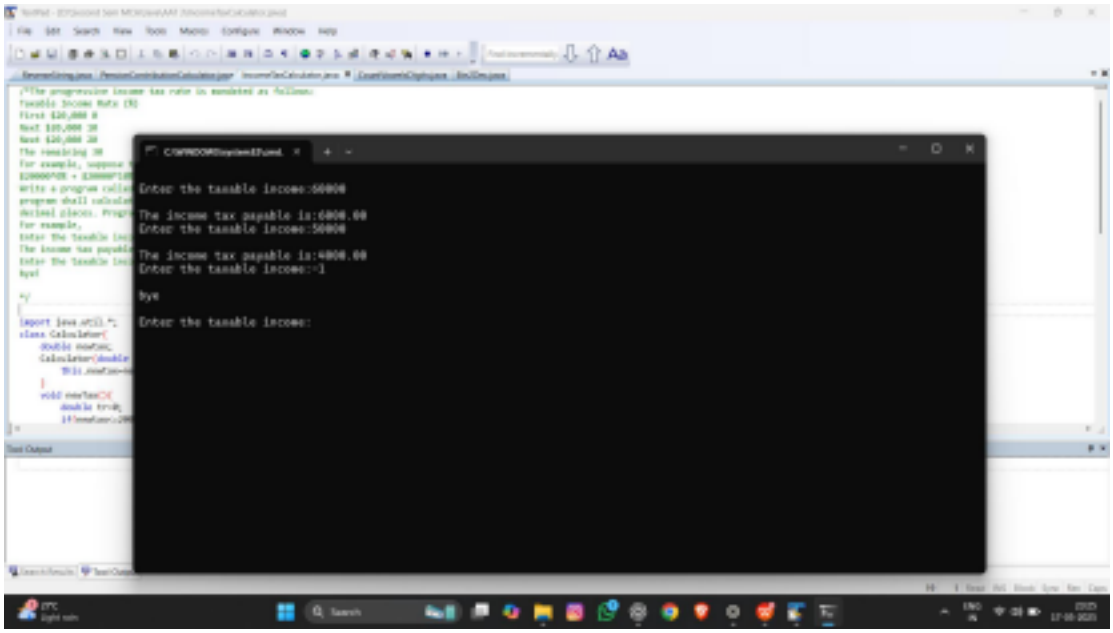
/*The progressive income tax rate is mandated as follows:
Taxable Income Rate (%)
First $20,000 0
Next $20,000 10
Next $20,000 20
The remaining 30
For example, suppose that the taxable income is $85000, the income taxpayable is
$20000*0% + $20000*10% + $20000*20% + $25000*30%. Write a program called IncomeTaxCalculator that reads the taxable income (in int). The program shall calculate the income tax payable (in double); and print theresult rounded to 2
decimal places. Program shall repeat the calculation until user enter -1.
For example,
Enter the taxable income: $41234
The income tax payable is: $2246.80
Enter the taxable income: $-1
bye!

*/

```
import java.util.*;
class Calculator{
	double newtax;
	Calculator(double newtax){
		this.newtax=newtax;
	}
	void newTax(){
		double tr=0;
		if(newtax<=20000){
			tr+=0;
	}else if(newtax>20000 && newtax<=40000){ tr+=newtax*0.10;
	}else if(newtax>40000 && newtax<=60000){ tr+=2000;
tr+=(newtax-40000)*0.20;
		}else if(newtax>60000){
			tr+=6000;
tr+=(newtax-60000)*0.30;
		}else{
System.out.println("Enter Correct Value:");
		}
		System.out.printf("\nThe income tax payable is:%.2f",tr); }
```

```
}

class IncomeTaxCalculator{
public static void main(String[] args){
Scanner sc=new Scanner(System.in);
                while(true){
System.out.print("\nEnter the taxable income:");
double tax=sc.nextDouble();
                        if(tax>-1){
Calculator c=new Calculator(tax);
c.newTax();
}else if(tax==-1){
System.out.println("\nbye");
                                break;
                        }
                }
        }
}
```

Output:-



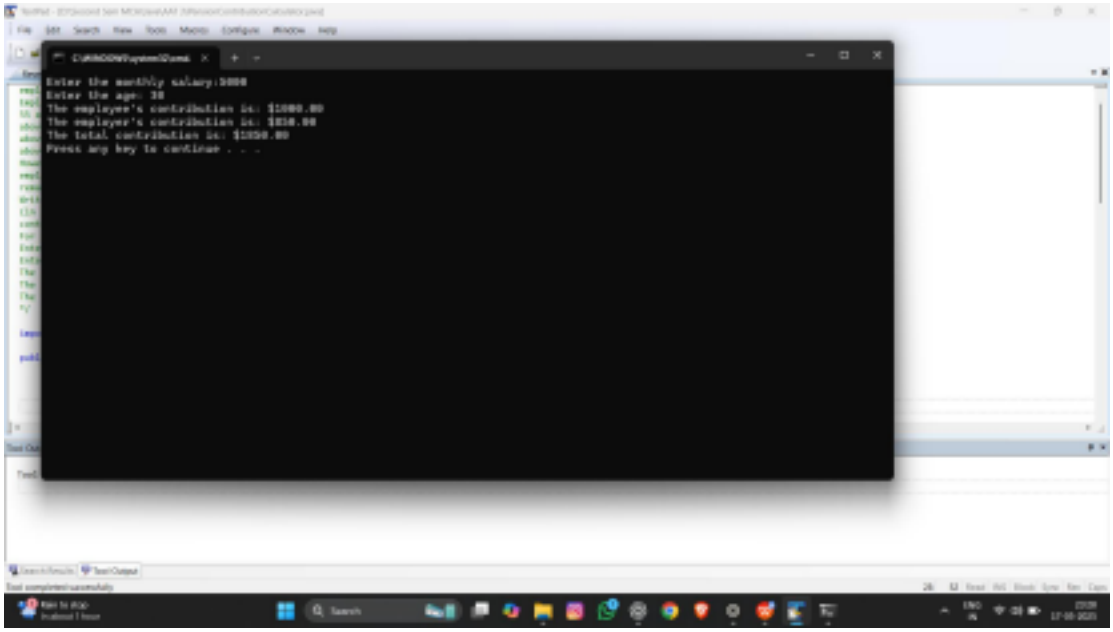
/*Both the employer and the employee are mandated to contribute a certain percentage of the
employee's salary towards the employee's pension fund. The rate is tabulated as follows:
Employee's Age Employee Rate (%) Employer Rate (%) 55
and below 20 17
above 55 to 60 13 13

above 60 to 65 7.5 9
above 65 5 7.5
However, the contribution is subjected to a salary ceiling of $6,000.
Inother words, if an
employee earns $6800, only $6000 attracts employee's and employer's contributions, the
remaining $800 does not.
Write a program called PensionContributionCalculator that reads the monthly salary and age
(in int) of an employee. Your program shall calculate the employee's, employer's and total
contributions (in double); and print the results rounded to 2 decimal places.
For example,
Enter the monthly salary: $3000
Enter the age: 30
The employee's contribution is: $600.00
The employer's contribution is: $510.00
The total contribution is: $1110.00
*/

```
import java.util.Scanner;

public class PensionContributionCalculator { public static
void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the monthly salary: $");
        double salary = scanner.nextDouble();

        System.out.print("Enter the age: ");
        int age = scanner.nextInt();

PensionCalculator calculator = new PensionCalculator(salary, age);
        double empContribution = calculator.getEmployeeContribution();
        double employerContribution =
calculator.getEmployerContribution();
double totalContribution = calculator.getTotalContribution();

        System.out.printf("The employee's contribution is: $%.2f%n",
empContribution);
        System.out.printf("The employer's contribution is: $%.2f%n",
employerContribution);
```

```
        System.out.printf("The total contribution is: $%.2f%n",
totalContribution);

        scanner.close();
    }
}

class PensionCalculator {
    private static final double SALARY_CEILING = 6000.0; private
    double salary;
    private int age;

    public PensionCalculator(double salary, int age) { this.salary =
Math.min(salary, SALARY_CEILING); // Applyceiling
        this.age = age;
    }

    public double getEmployeeRate() {
        if (age <= 55) return 0.20;
        else if (age <= 60) return 0.13;
        else if (age <= 65) return 0.075;
        else return 0.05;
    }

    public double getEmployerRate() {
        if (age <= 55) return 0.17;
        else if (age <= 60) return 0.13;
        else if (age <= 65) return 0.09;
        else return 0.075;
    }

public double getEmployeeContribution() { return salary *
    getEmployeeRate();
    }

    public double getEmployerContribution() {
        return salary * getEmployerRate();
    }

    public double getTotalContribution() {
        return getEmployeeContribution() + getEmployerContribution(); }
}
```

Output:-

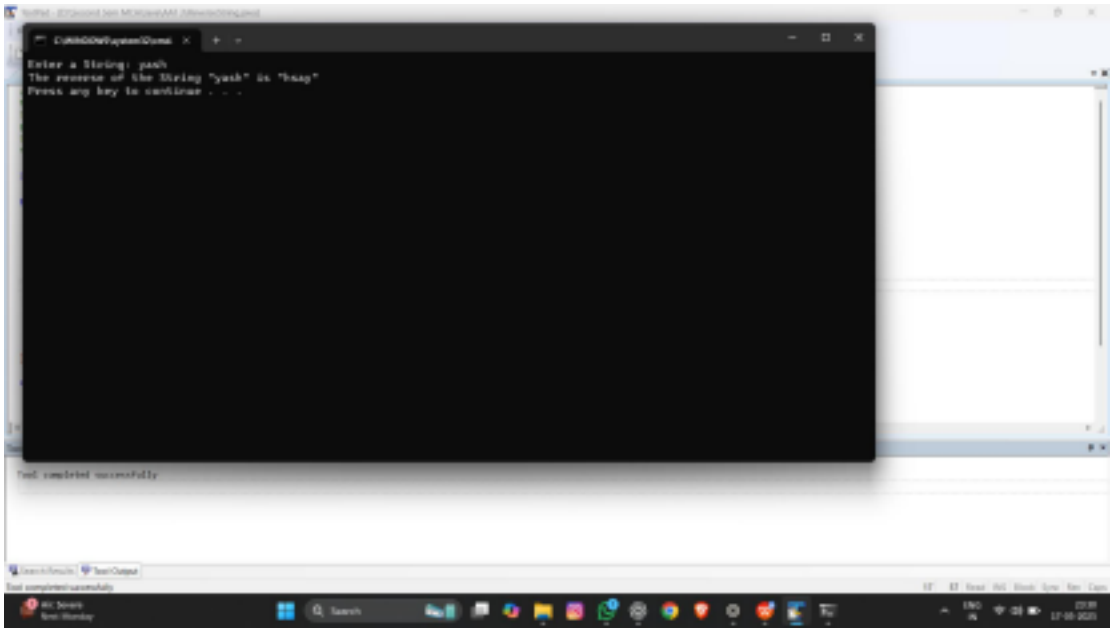


/* 3.Write a program called ReverseString, which prompts user for a String, and prints the reverse of
the String by extracting and processing each character.
The output shall look like:
Enter a String: abcdef
The reverse of the String "abcdef" is "fedcba"
*/

```
import java.util.Scanner;

public class ReverseString {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);

System.out.print("Enter a String: ");
String input = scanner.nextLine();

String reversed = StringReverser.reverse(input);

System.out.println("The reverse of the String \"" + input + "\" is \""+ reversed +
"\"");

scanner.close();
```

```
}
}

class StringReverser {
public static String reverse(String str) {
String result = "";
for (int i = str.length() - 1; i >= 0; i--) {
result += str.charAt(i);
}
return result;
}
}
```

Output:-



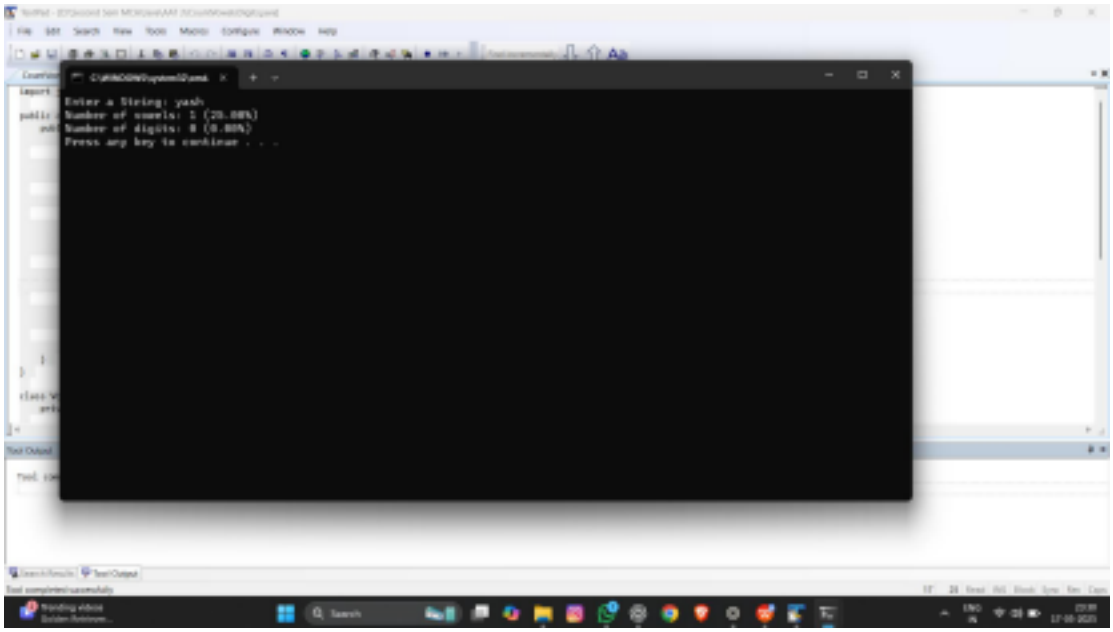
4/*Write a program called CountVowelsDigits, which prompts the user for a String, counts the
number of vowels (a, e, i, o, u, A, E, I, O, U) and digits (0-9) containedinthe string, and prints
the counts and the percentages (rounded to 2 decimal places). For example,
Enter a String: testing12345
Number of vowels: 2 (16.67%)
Number of digits: 5 (41.67%)
*/

```
import java.util.Scanner;

public class CountVowelsDigits {
```

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

        System.out.print("Enter a String: ");
        String input = scanner.nextLine();

VowelDigitCounter counter = new VowelDigitCounter(input);

        int vowelCount = counter.countVowels();
        int digitCount = counter.countDigits();
        int totalLength = input.length();
        double vowelPercent = (totalLength == 0) ? 0 : (vowelCount
*100.0)/ totalLength;
        double digitPercent = (totalLength == 0) ? 0 : (digitCount * 100.0) /
totalLength;

        System.out.printf("Number of vowels: %d (%.2f%%)%n",
vowelCount, vowelPercent);
        System.out.printf("Number of digits: %d (%.2f%%)%n", digitCount,
digitPercent);

        scanner.close();
    }
}

class VowelDigitCounter {
    private String str;

    public VowelDigitCounter(String str) {
        this.str = str;
    }

    public int countVowels() {
        int count = 0;
        for (char ch : str.toCharArray()) {
if ("aeiouAEIOU".indexOf(ch) != -1) {
                count++;
            }
        }
        return count;
    }

    public int countDigits() {
```

```
        int count = 0;
        for (char ch : str.toCharArray()) {
            if (Character.isDigit(ch)) {
                count++;
            }
        }
        return count;
    }
}
```

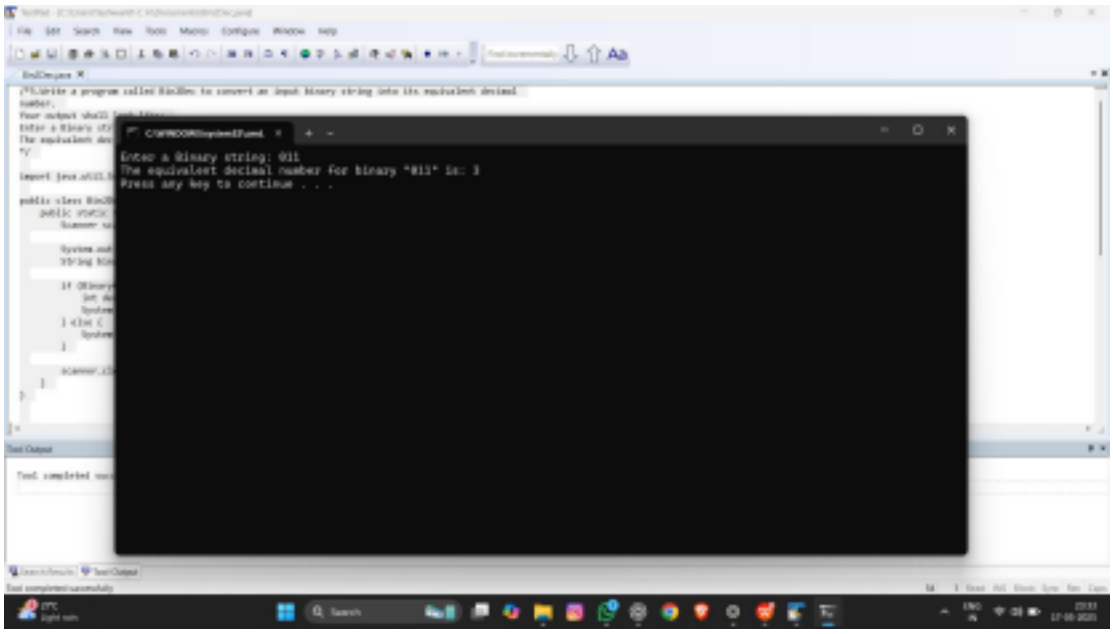
Output:-



```
/*5.Write a program called Bin2Dec to convert an input binary stringintoits
equivalent decimal
number.
Your output shall look like:
Enter a Binary string: 1011
The equivalent decimal number for binary "1011" is: 11 */

import java.util.Scanner;

public class Bin2Dec {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a Binary string: ");
        String binaryStr = scanner.nextLine();
```

```
    if (BinaryConverter.isBinary(binaryStr)) { int
        decimalValue =
BinaryConverter.convertToDecimal(binaryStr);
            System.out.println("The equivalent decimal number for
binary\""+ binaryStr + "\" is: " + decimalValue);
        } else {
            System.out.println("Error: \"" + binaryStr + "\" is not a
validbinary string.");
        }

        scanner.close();
    }
}



class BinaryConverter {
    public static boolean isBinary(String str) { for (char ch
    : str.toCharArray()) {
            if (ch != '0' && ch != '1') {
                return false;
            }
        }
        return true;
    }

public static int convertToDecimal(String binaryStr) { int decimal =
    0;
        int power = 0;

    for (int i = binaryStr.length() - 1; i >= 0; i--) { char bit =
        binaryStr.charAt(i);
            if (bit == '1') {
                decimal += Math.pow(2, power);
            }
            power++;
        }
        return decimal;
    }
}
```

Output:-


Enter a Binary string: 011
The equivaleet decimal number for binary "011" is: 3
Press any key to continue . . .

AAT4

/*Provided that you have a given number of small rice bags (1 kilo each) and big ricebags (5 kilos each), write a method that returns true if it is possible to make a package withgoal kilos
of rice.*/

```
public class RicePacker {

public static boolean canPack(int s, int b, int g) {
if (s < 0 || b < 0 || g < 0) return false;
int maxBigNeeded = g / 5;
int usedBig = Math.min(maxBigNeeded, b);
int remKg = g - usedBig * 5;
return s >= remKg;
}

public static void main(String[] args) {
System.out.println(canPack( 4, 1, 9));
System.out.println(canPack( 4, 1, 10));
System.out.println(canPack( 6, 2, 11));
System.out.println(canPack(-1, 2, 5));
System.out.println(canPack( 5, 0, 5));
}
}
```

OUTPUT

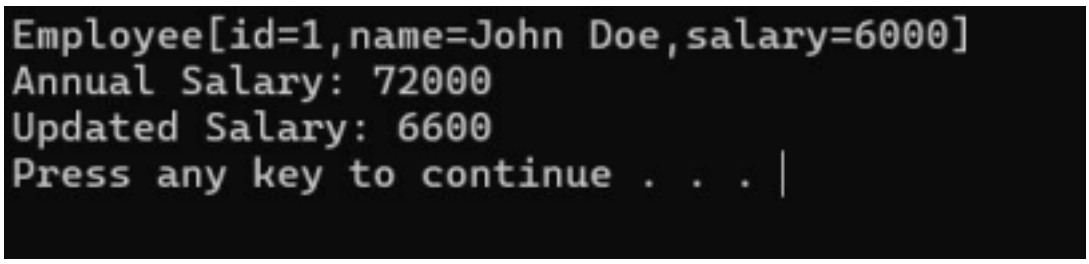
```
true
false
true
false
true
Press any key to continue . . .
```

/*Write a class called Employee, which models an employee with an ID, name andsalary, is
designed as shown in the following class diagram. The method raiseSalary(percent) increases
the salary by the given percentage. Write the Employee class.*/

```
public class Employee {
private int id;
private String fn;
private String ln;
private int sal;

public Employee(int id, String fn, String ln, int sal) { this.id = id;
this.fn = fn;
this.ln = ln;
this.sal = sal;
}

public int getId() {
return id;
}

public String getFn() {
return fn;
}

public String getLn() {
return ln;
}

public int getSal() {
return sal;
}

public void setSal(int sal) {
this.sal = sal;
}

public int getAnnualSal() {
return sal * 12;
}

public int raiseSal(int p) {
sal += sal * p / 100;
return sal;
}

@Override
public String toString() {
   return String.format("Employee[id=%d,name=%s %s,salary=%d]", id, fn, ln, sal); }

public static void main(String[] args) {
Employee e = new Employee(1, "John", "Doe", 6000);
System.out.println(e);
System.out.println("Annual Salary: " + e.getAnnualSal());
e.raiseSal(10);
```

```
System.out.println("Updated Salary: " + e.getSal());
}
}
```

OUTPUT

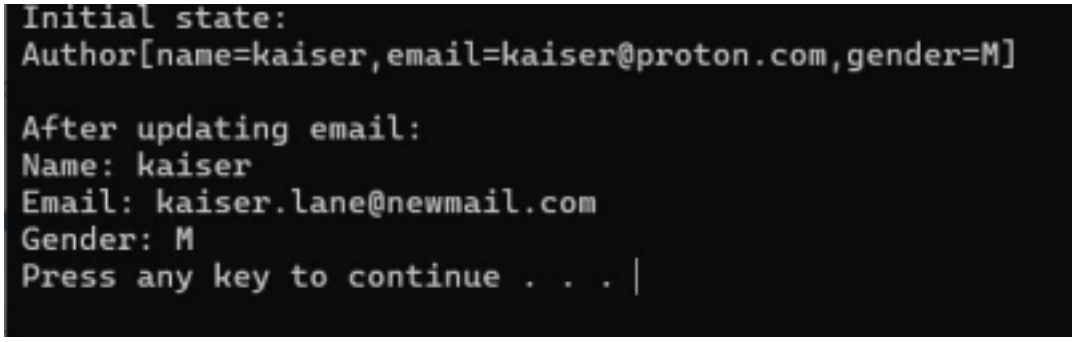


/*3. Write a class called Author (as shown in the class diagram) is designed to model a book's
author. It contains:
• Three private instance variables: name (String), email (String), and gender (char of either 'm' or 'f');
• One constructor to initialize the name, email and gender with the given values;
public Author (String name, String email, char gender) {.....} • public getters/setters: getName(), getEmail(), setEmail(), and getGender(); • A toString() method that returns "Author[name=?,email=?,gender=?]", e.g., "Author[name=Tan Ah Teck,email=ahTeck@somewhere.com,gender=m]". */

```
public class Author {
private String n;
private String e;
private char g;

public Author(String n, String e, char g) {
this.n = n;
this.e = e;
this.g = g;
}

public String getName() {
return n;
}
public String getEmail() {
return e;
}

public void setEmail(String e) {
this.e = e;
}

public char getGender() {
return g;
}
```

```
@Override
public String toString() {
   return String.format("Author[name=%s,email=%s,gender=%c]", n, e, g); }

public static void main(String[] args) {
Author author = new Author("kaiser", "kaiser@proton.com", 'M');

System.out.println("Initial state:");
System.out.println(author);

author.setEmail("kaiser.lane@newmail.com");

System.out.println("\nAfter updating email:");
System.out.println("Name: " + author.getName());
System.out.println("Email: " + author.getEmail());
System.out.println("Gender: " + author.getGender());
}
}
```

OUTPUT



/*4. In this exercise, a subclass called Cylinder is derived from the superclass Circle as shown in the
class diagram. Study how the subclass Cylinder invokes the superclass' constructors (via super
() and super(radius)) and inherits the variables and methods from the superclass Circle. • The subclass Cylinder inherits getArea() method from its superclass Circle. Try overriding the getArea() method in the subclass Cylinder to compute the surface area(=2π×radius×height + 2×base-area) of the cylinder instead of base area. o That is, if getArea() is called by a Circle instance, it returns the area.

o If getArea() is called by a Cylinder instance, it returns the surface area of the cylinder.

• If you override the getArea() in the subclass Cylinder, the getVolume() no longer works. This is because the getVolume() uses the overridden getArea() method foundin the same class. Fix the getVolume().

*/

```
class Circle {
private double radius;
private String color;
```

```
    public Circle() {
    this.radius = 1.0;
    this.color = "red";
    }

    public Circle(double radius) {
    this.radius = radius;
    this.color = "red";
    }

    // Constructor with radius and color
    public Circle(double radius, String color) {
    this.radius = radius;
    this.color = color;
    }

    public double getRadius() {
    return radius;
    }

    public String getColor() {
    return color;
    }

    public double getArea() {
    return Math.PI * radius * radius;
    }
    }

    public class Cylinder extends Circle {
    private double height;

    public Cylinder() {
    super();
    this.height = 1.0;
    }

    public Cylinder(double radius) {
    super(radius);
    this.height = 1.0;
    }

    public Cylinder(double radius, double height) {
    super(radius);
    this.height = height;
    }

public Cylinder(double radius, double height, String color) { super(radius,
    color);
    this.height = height;
```

```
}

public double getHeight() {
return height;
}

public void setHeight(double height) {
this.height = height;
}

@Override
public double getArea() {
   return 2 * Math.PI * getRadius() * height + 2 * super.getArea(); }

public double getVolume() {
return super.getArea() * height;
}

public static void main(String[] args) {
Circle circle = new Circle(5);
System.out.println("Circle Area (radius 5): " + circle.getArea());

Cylinder cylinder = new Cylinder(3, 4);
      System.out.println("Cylinder Surface Area (radius 3, height 4): " +
cylinder.getArea());
System.out.println("Cylinder Volume: " + cylinder.getVolume());

Circle polyCylinder = new Cylinder(2, 5);
      System.out.println("Polymorphic getArea() (radius 2, height 5): " +
polyCylinder.getArea());
}
}
```

OUTPUT

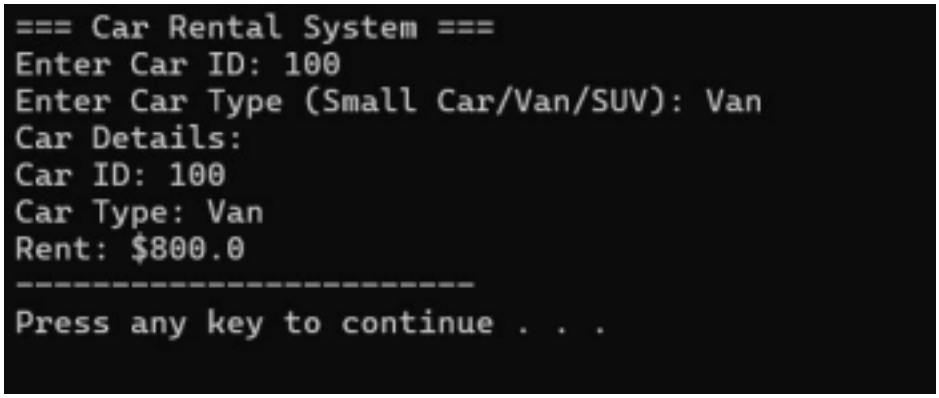
```
Circle Area (radius 5): 78.539816339744 83
Cylinder Surface Area (radius 3, height 4): 131.94689145077132
Cylinder Volume: 113.09733552923255
Polymorphic getArea() (radius 2, height 5): 87.96459430051421
Press any key to continue . . .
```

```
/*
5. Define a class CARRENTAL with the following details : • Class Members are:
CarId of int type, CarType of string type and Rent of float type. • Define GetCar()
method which accepts CarId and CarType.
• GetRent() method which return rent of the car on the basis of car type, i.e. Small Car =
1000, Van = 800, SUV = 2500
• ShowCar() method which allow user to view the contents of cars i.e. id, type andrent.
*/
```

```
import java.util.Scanner;

public class CARRENTAL {
private int CarId;
private String CarType;
private float Rent;

public CARRENTAL() {
this.CarId = 0;
this.CarType = "";
this.Rent = 0.0f;
}

public void GetCar(int carId, String carType) {
this.CarId = carId;
this.CarType = carType;
this.Rent = GetRent();
}

public float GetRent() {
switch (CarType.toLowerCase()) {
case "small car":
return 1000.0f;
case "van":
return 800.0f;
case "suv":
return 2500.0f;
default:
return 0.0f;
}
}

public void ShowCar() {
System.out.println("Car Details:");
System.out.println("Car ID: " + CarId);
System.out.println("Car Type: " + CarType);
System.out.println("Rent: $" + Rent);
System.out.println("------------------------");
}

public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
CARRENTAL car = new CARRENTAL();

System.out.println("=== Car Rental System ===");

System.out.print("Enter Car ID: ");
int id = scanner.nextInt();
scanner.nextLine();
```

```
System.out.print("Enter Car Type (Small Car/Van/SUV): "); String
type = scanner.nextLine();

car.GetCar(id, type);

car.ShowCar();

scanner.close();
}
}
```

OUTPUT



AAT5

/*Create an abstract class “BankAccount” with abstract methods “deposit()” and “withdraw()”. Implement two subclasses “SavingsAccount” and “CheckingAccount”which extend “BankAccount” and implement the abstract methods. Create a “Customer” class which contains a list of “BankAccount” objects. Add methods tothe “Customer” class to display account balances, deposit/withdraw money, etc. Create objects of all classes and test their behavior.*/

```
// Abstract BankAccount class

import java.util.*;

abstract class BankAccount {
   protected String accountNumber;
   protected double balance;
   protected String accountType;

   public BankAccount(String accountNumber, double initialBalance, String
accountType) {
      this.accountNumber = accountNumber;
      this.balance = initialBalance;
      this.accountType = accountType;
   }

   // Abstract methods to be implemented by subclasses
   public abstract boolean deposit(double amount);
   public abstract boolean withdraw(double amount);

   // Concrete methods
   public double getBalance() {
      return balance;
   }

   public String getAccountNumber() {
      return accountNumber;
   }

   public String getAccountType() {
      return accountType;
   }

   @Override
   public String toString() {
      return String.format("%s Account: %s, Balance: $%.2f",
```

```
accountType, accountNumber, balance);
    }
}

// SavingsAccount implementation
class SavingsAccount extends BankAccount {
    private static final double MINIMUM_BALANCE = 100.0; private static final double
    INTEREST_RATE = 0.02; // 2% annual interest

public SavingsAccount(String accountNumber, double initialBalance) {
    super(accountNumber, initialBalance, "Savings");
    }

    @Override
    public boolean deposit(double amount) {
        if (amount <= 0) {
            System.out.println("Deposit amount must be positive.");
            return false;
        }
        balance += amount;
        System.out.printf("Deposited $%.2f to Savings Account %s. New balance: $%.2f%n",
                amount, accountNumber, balance);
        return true;
    }

    @Override
    public boolean withdraw(double amount) {
        if (amount <= 0) {
System.out.println("Withdrawal amount must be positive.");
            return false;
        }
        if (balance - amount < MINIMUM_BALANCE) {
            System.out.printf("Withdrawal denied. Minimum balance of $%.2f
required.%n",
                    MINIMUM_BALANCE);
            return false;
        }
        balance -= amount;
        System.out.printf("Withdrew $%.2f from Savings Account %s. Newbalance:
$%.2f%n",
                amount, accountNumber, balance);
        return true;
    }

    public void addInterest() {
        double interest = balance * INTEREST_RATE / 12; // Monthly interest
        balance += interest;
System.out.printf("Monthly interest of $%.2f added to account %s%n", interest,
        accountNumber);
    }
```

```
}

// CheckingAccount implementation
class CheckingAccount extends BankAccount {
   private static final double OVERDRAFT_LIMIT = 500.0; private
   static final double OVERDRAFT_FEE = 35.0;

public CheckingAccount(String accountNumber, double initialBalance) {
   super(accountNumber, initialBalance, "Checking");
   }

   @Override
   public boolean deposit(double amount) {
      if (amount <= 0) {
         System.out.println("Deposit amount must be positive.");
         return false;
      }
      balance += amount;
      System.out.printf("Deposited $%.2f to Checking Account %s. Newbalance:
$%.2f%n",
                  amount, accountNumber, balance);
      return true;
   }

   @Override
   public boolean withdraw(double amount) {
      if (amount <= 0) {
System.out.println("Withdrawal amount must be positive.");
         return false;
      }

      if (balance >= amount) {
         // Sufficient funds
         balance -= amount;
         System.out.printf("Withdrew $%.2f from Checking Account %s. Newbalance:
$%.2f%n",
                     amount, accountNumber, balance);
         return true;
   } else if (balance + OVERDRAFT_LIMIT >= amount) { // Overdraft
      allowed
         balance -= amount;
         balance -= OVERDRAFT_FEE; // Apply overdraft fee
         System.out.printf("Overdraft withdrawal of $%.2f from account %s. "
         +"Overdraft fee: $%.2f. New balance: $%.2f%n",
amount, accountNumber, OVERDRAFT_FEE, balance);
         return true;
      } else {
         System.out.printf("Withdrawal denied. Insufficient funds and overdraft limit
exceeded.%n");
         return false;
```

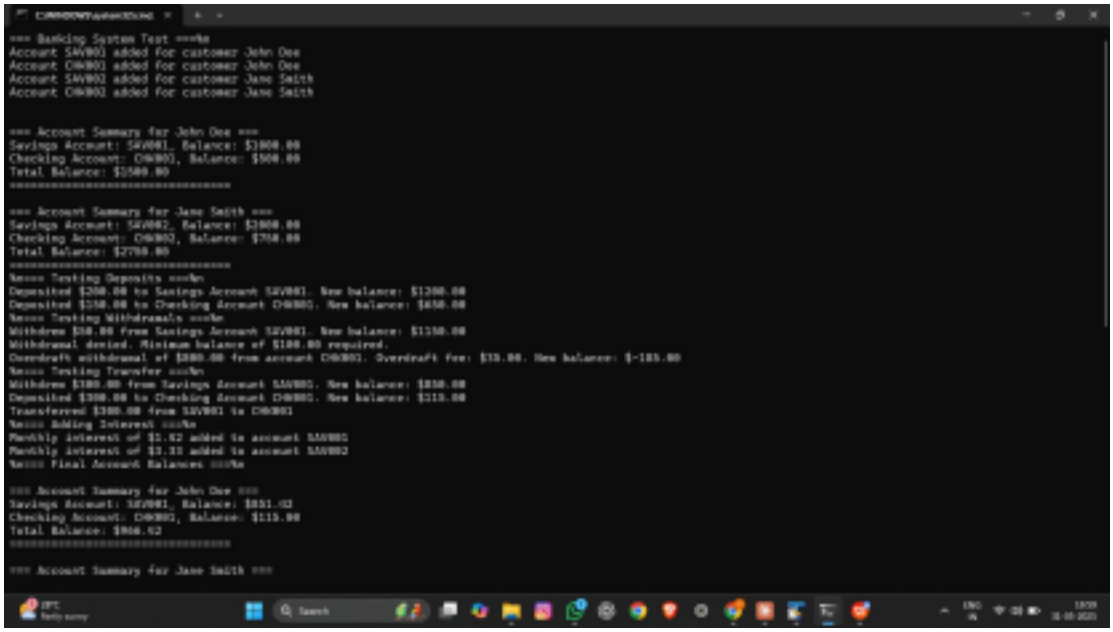
```
        }
    }
}

// Customer class

class Customer {
    private String name;
    private String customerId;
    private List<BankAccount> accounts;

    public Customer(String name, String customerId) {
        this.name = name;
        this.customerId = customerId;
        this.accounts = new ArrayList<>();
    }

    public void addAccount(BankAccount account) {
        accounts.add(account);
        System.out.printf("Account %s added for customer %s%n",
                account.getAccountNumber(), name);
    }

    public void displayAllBalances() {
        System.out.printf("%n=== Account Summary for %s ===%n", name);
        double totalBalance = 0;
        for (BankAccount account : accounts) {
            System.out.println(account);
            totalBalance += account.getBalance();
        }
        System.out.printf("Total Balance: $%.2f%n", totalBalance);
        System.out.println("=============================="); }

public boolean depositToAccount(String accountNumber, double amount) {
    BankAccount account = findAccount(accountNumber);
        if (account != null) {
            return account.deposit(amount);
        } else {
System.out.printf("Account %s not found for customer %s%n",
                        accountNumber, name);
            return false;
        }
    }

public boolean withdrawFromAccount(String accountNumber, double amount)
    {BankAccount account = findAccount(accountNumber);
        if (account != null) {
            return account.withdraw(amount);
        } else {
```

```
            System.out.printf("Account %s not found for customer %s%n",
            accountNumber, name);
            return false;
        }
    }

    public boolean transferFunds(String fromAccount, String toAccount, double amount)
{
        BankAccount from = findAccount(fromAccount);
        BankAccount to = findAccount(toAccount);

        if (from == null || to == null) {
            System.out.println("One or both accounts not found.");
            return false;
        }

        if (from.withdraw(amount)) {
            to.deposit(amount);
            System.out.printf("Transferred $%.2f from %s to %s%n", amount,
            fromAccount, toAccount);
            return true;
        }
        return false;
    }

private BankAccount findAccount(String accountNumber) { for
    (BankAccount account : accounts) {
        if (account.getAccountNumber().equals(accountNumber)) { return
            account;
            }
        }
        return null;
    }

    public String getName() {
        return name;
    }

    public String getCustomerId() {
        return customerId;
    }

    public List<BankAccount> getAccounts() {
        return new ArrayList<>(accounts); // Return copy to prevent external
modification
    }
}
// Main class to test the banking system public class
BankingSystemTest {
    public static void main(String[] args) {
```

```
System.out.println("=== Banking System Test ===%n");

        // Create customers
        Customer customer1 = new Customer("John Doe", "CUST001");
        Customer customer2 = new Customer("Jane Smith", "CUST002");

        // Create accounts
        SavingsAccount savings1 = new SavingsAccount("SAV001", 1000.0);
        CheckingAccount checking1 = new CheckingAccount("CHK001", 500.0);
        SavingsAccount savings2 = new SavingsAccount("SAV002", 2000.0);
        CheckingAccount checking2 = new CheckingAccount("CHK002", 750.0);

        // Add accounts to customers
        customer1.addAccount(savings1);
        customer1.addAccount(checking1);
        customer2.addAccount(savings2);
        customer2.addAccount(checking2);

        System.out.println();

        // Display initial balances
        customer1.displayAllBalances();
        customer2.displayAllBalances();

        // Test deposits
        System.out.println("%n=== Testing Deposits ===%n");
        customer1.depositToAccount("SAV001", 200.0);
        customer1.depositToAccount("CHK001", 150.0);

        // Test withdrawals
        System.out.println("%n=== Testing Withdrawals ===%n");
        customer1.withdrawFromAccount("SAV001", 50.0); // Should work
customer1.withdrawFromAccount("SAV001", 1500.0); // Should fail
(belowminimum)
        customer1.withdrawFromAccount("CHK001", 800.0); // Should trigger
overdraft

        // Test transfer
        System.out.println("%n=== Testing Transfer ===%n");
        customer1.transferFunds("SAV001", "CHK001", 300.0);

        // Test interest on savings account
        System.out.println("%n=== Adding Interest ===%n");
        savings1.addInterest();
        savings2.addInterest();
        // Display final balances
        System.out.println("%n=== Final Account Balances ===%n");
        customer1.displayAllBalances();
        customer2.displayAllBalances();
```

```
        // Test error cases
        System.out.println("%n=== Testing Error Cases ===%n");
        customer1.depositToAccount("INVALID", 100.0); // Invalid account
        customer1.withdrawFromAccount("CHK001", -50.0); // Negative amount
        customer1.depositToAccount("SAV001", 0.0); // Zero deposit

        // Demonstrate polymorphism
        System.out.println("%n=== Demonstrating Polymorphism ===%n");
        List<BankAccount> allAccounts = new ArrayList<>();
        allAccounts.addAll(customer1.getAccounts());
        allAccounts.addAll(customer2.getAccounts());

        System.out.println("All accounts in the system:");
        for (BankAccount account : allAccounts) {
            System.out.println(account);
        }
    }
}
```

Output:-



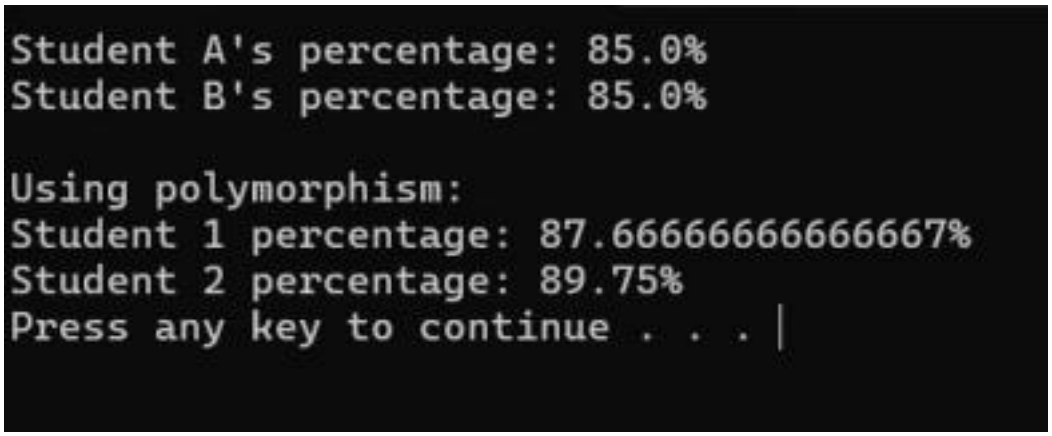
/*An abstract class called “Marks” is needed to calculate the percentage of marks earned by students A in three subjects (with each subject out of 100) and student B in four subjects (with each subject out of 100). This class must contain the abstract method “getPercentage,” which two other classes, “A” and “B,” will inherit. The method “getPercentage,” which provides the percentage of students, is shared by classes “A” and “B.”*/

```
// Abstract class Marks
abstract class Marks {
    // Abstract method to calculate percentage
    public abstract double getPercentage();
```

```
}

// Class A for student with 3 subjects
class A extends Marks {
    private double subject1, subject2, subject3;

    // Constructor
public A(double subject1, double subject2, double subject3) {
    this.subject1 = subject1;
        this.subject2 = subject2;
        this.subject3 = subject3;
    }

    // Implementation of getPercentage method for 3 subjects
    @Override
    public double getPercentage() {
        double totalMarks = subject1 + subject2 + subject3;
        double maxMarks = 3 * 100; // 3 subjects, each out of 100 return
        (totalMarks / maxMarks) * 100;
    }
}

// Class B for student with 4 subjects
class B extends Marks {
    private double subject1, subject2, subject3, subject4;

    // Constructor
public B(double subject1, double subject2, double subject3, double subject4) {
    this.subject1 = subject1;
        this.subject2 = subject2;
        this.subject3 = subject3;
        this.subject4 = subject4;
    }

    // Implementation of getPercentage method for 4 subjects
    @Override
    public double getPercentage() {
        double totalMarks = subject1 + subject2 + subject3 + subject4;
        double maxMarks = 4 * 100; // 4 subjects, each out of 100 return
        (totalMarks / maxMarks) * 100;
    }
}

// Main class to test the program
public class MarksCalculator {
    public static void main(String[] args) {
        // Create student A with marks in 3 subjects
        A studentA = new A(85, 92, 78);
        System.out.println("Student A's percentage: " + studentA.getPercentage() + "%"); //

        Create student B with marks in 4 subjects

        B studentB = new B(88, 76, 94, 82);
System.out.println("Student B's percentage: " + studentB.getPercentage() + "%");

        // Demonstrating polymorphism
        System.out.println("\nUsing polymorphism:");
        Marks[] students = {
            new A(90, 85, 88),
            new B(92, 87, 91, 89)
```

```
        };

        for (int i = 0; i < students.length; i++) {
            System.out.println("Student " + (i + 1) + " percentage: " +
            students[i].getPercentage() + "%");
        }
    }
}
```

Output:-



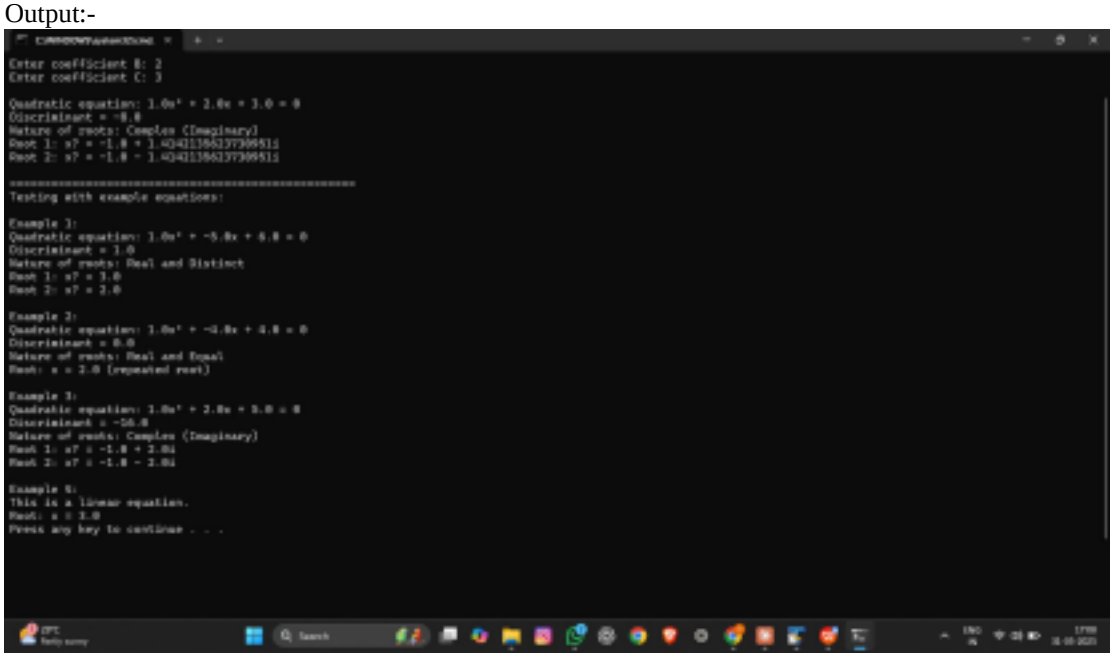
/*3. Write a Java program using a function root to calculate and display all roots of the quadratic
AX2 + BX + C = 0.*/

```
import java.util.Scanner;

public class QuadraticRoots {

    // Function to calculate and display roots of quadratic equation public
    static void root(double a, double b, double c) {
        // Check if it's a valid quadratic equation
        if (a == 0) {
            if (b == 0) {
                if (c == 0) {
                    System.out.println("Infinite solutions (0 = 0)");
                } else {
                    System.out.println("No solution (inconsistent equation)");
                }
            } else {
                // Linear equation: bx + c = 0
                double linearRoot = -c / b;
                System.out.println("This is a linear equation.");
                System.out.println("Root: x = " + linearRoot);
            }
            return;
        }

        // Calculate discriminant
        double discriminant = (b * b) - (4 * a * c);

        System.out.println("Quadratic equation: " + a + "x² + " + b + "x + " + c + " = 0");
        System.out.println("Discriminant = " + discriminant);

        // Check nature of roots based on discriminant
```

```
    if (discriminant > 0) {
        // Two distinct real roots
        double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
        double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);

        System.out.println("Nature of roots: Real and Distinct");
        System.out.println("Root 1: x₁ = " + root1);
        System.out.println("Root 2: x₂ = " + root2);

    } else if (discriminant == 0) {
        // One repeated real root
        double root1 = -b / (2 * a);

        System.out.println("Nature of roots: Real and Equal");
        System.out.println("Root: x = " + root1 + " (repeated root)");

    } else {
        // Complex roots
        double realPart = -b / (2 * a);
        double imaginaryPart = Math.sqrt(-discriminant) / (2 * a);

        System.out.println("Nature of roots: Complex (Imaginary)");
        System.out.println("Root 1: x₁ = " + realPart + " + " + imaginaryPart + "i");
    System.out.println("Root 2: x₂ = " + realPart + " - " + imaginaryPart + "i"); }
}

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

    System.out.println("Quadratic Equation Solver: AX² + BX + C = 0");
    System.out.println("==============================================")
    ;

    // Get coefficients from user
    System.out.print("Enter coefficient A: ");
    double a = scanner.nextDouble();

    System.out.print("Enter coefficient B: ");
    double b = scanner.nextDouble();

    System.out.print("Enter coefficient C: ");
    double c = scanner.nextDouble();

    System.out.println();

    // Call the root function
    root(a, b, c);

    System.out.println("\n" + "=".repeat(50));
    // Test with predefined examples
    System.out.println("Testing with example equations:");
    System.out.println();

    // Example 1: Two distinct real roots
    System.out.println("Example 1:");
    root(1, -5, 6); // x² - 5x + 6 = 0

    System.out.println();

    // Example 2: Equal roots
```

```
        System.out.println("Example 2:");
        root(1, -4, 4); // x² - 4x + 4 = 0

        System.out.println();

        // Example 3: Complex roots
        System.out.println("Example 3:");
        root(1, 2, 5); // x² + 2x + 5 = 0

        System.out.println();

        // Example 4: Linear equation
        System.out.println("Example 4:");
        root(0, 2, -6); // 2x - 6 = 0

        scanner.close();
    }
}
```

Output:-



/*Write a program to find the volume of a box that has its sides w, h, d as width, height, and depth, respectively. Its volume is v = w * h * d and also fi nd the surface area given by the formula s = 2(wh + hd + dw).*/

```
import java.util.Scanner;

public class BoxCalculator {
    // Method to calculate volume
public static double calculateVolume(double width, double height, double depth) { return
    width * height * depth;
    }

    // Method to calculate surface area
public static double calculateSurfaceArea(double width, double height, double depth) { return 2 *
    (width * height + height * depth + depth * width);
    }

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

```
        System.out.println("Box Volume and Surface Area Calculator");
        System.out.println("====================================");

        // Get input from user
        System.out.print("Enter the width (w): ");
        double w = scanner.nextDouble();

        System.out.print("Enter the height (h): ");
        double h = scanner.nextDouble();

        System.out.print("Enter the depth (d): ");
        double d = scanner.nextDouble();

        // Calculate volume and surface area
        double volume = calculateVolume(w, h, d);
        double surfaceArea = calculateSurfaceArea(w, h, d);

        // Display results
        System.out.println("\nResults:");
        System.out.println("--------");
        System.out.printf("Width: %.2f\n", w);
        System.out.printf("Height: %.2f\n", h);
        System.out.printf("Depth: %.2f\n", d);
        System.out.printf("Volume (v = w * h * d): %.2f\n", volume);
        System.out.printf("Surface Area (s = 2(wh + hd + dw)): %.2f\n", surfaceArea);

        scanner.close();
    }
}
```

Output:-

```
Box Volume and Surface Area Calculator
====================================
Enter the width (w): 5
Enter the height (h): 10
Enter the depth (d): 3

Results:
--------
Width: 5.00
Height: 10.00
Depth: 3.00
Volume (v = w * h * d): 150.00
Surface Area (s = 2(wh + hd + dw)): 190.00
Press any key to continue . . .
```

/*A class weight is having a data member pound, which will have the weight in pounds. Using a conversion function, convert the weight in pounds to weight in kilograms which is of double type. Write a program to do this. 1 pounds =1 kg/0.453592
Use default constructor to initial assignment of 1000 pounds.*/

```
public class Weight {
    // Data member to store weight in pounds
    private double pound;

    // Default constructor - initializes weight to 1000 pounds
    public Weight() {
        this.pound = 1000.0;
    }
```

```
// Parameterized constructor
public Weight(double pound) {
    this.pound = pound;
}

// Conversion function to convert pounds to kilograms
public double convertToKilograms() {
    // Using the conversion formula: 1 pound = 1 kg / 0.453592
    // Therefore, kilograms = pounds / 0.453592
    return this.pound / 0.453592;
}

// Getter method for pound
public double getPound() {
    return this.pound;
}

// Setter method for pound
public void setPound(double pound) {
    this.pound = pound;
}

// Method to display weight information
public void displayWeight() {
    System.out.println("Weight in Pounds: " + this.pound);
    System.out.println("Weight in Kilograms: " + String.format("%.2f",
convertToKilograms())); }

// Main method to test the Weight class
public static void main(String[] args) {
    System.out.println("=== Weight Conversion Program ===\n");

    // Using default constructor (1000 pounds)
    Weight defaultWeight = new Weight();
    System.out.println("Using Default Constructor:");
    defaultWeight.displayWeight();

    System.out.println("\n" + "=".repeat(40) + "\n");

    // Using parameterized constructor with different weights
    Weight weight1 = new Weight(100.0);
    System.out.println("Weight Example 1 (100 pounds):");
    weight1.displayWeight();
    System.out.println("\n" + "-".repeat(30) + "\n");

    Weight weight2 = new Weight(250.5);
    System.out.println("Weight Example 2 (250.5 pounds):");
    weight2.displayWeight();

    System.out.println("\n" + "-".repeat(30) + "\n");

    // Demonstrating setter method
    Weight weight3 = new Weight();
    weight3.setPound(75.0);
    System.out.println("Weight Example 3 (modified to 75
    pounds):"); weight3.displayWeight();

    System.out.println("\n" + "=".repeat(40) + "\n");

    // Verification of conversion formula
```

```
        System.out.println("Conversion Formula Verification:");
        System.out.println("1 pound = " + String.format("%.6f", 1.0/0.453592) + "
        kilograms"); System.out.println("1 kilogram = 0.453592 pounds");
    }
}
```

Output:-

```
=== Weight Conversion Program ===

Using Default Constructor:
Weight in Pounds: 1000.0
Weight in Kilograms: 2204.62

========================================

Weight Example 1 (100 pounds):
Weight in Pounds: 100.0
Weight in Kilograms: 220.46

----------------------------

Weight Example 2 (250.5 pounds):
Weight in Pounds: 250.5
Weight in Kilograms: 552.26

----------------------------

Weight Example 3 (modified to 75 pounds):
Weight in Pounds: 75.0
Weight in Kilograms: 165.35

========================================

Conversion Formula Verification:
1 pound = 2.204624 kilograms
1 kilogram = 0.453592 pounds
Press any key to continue . . .
```

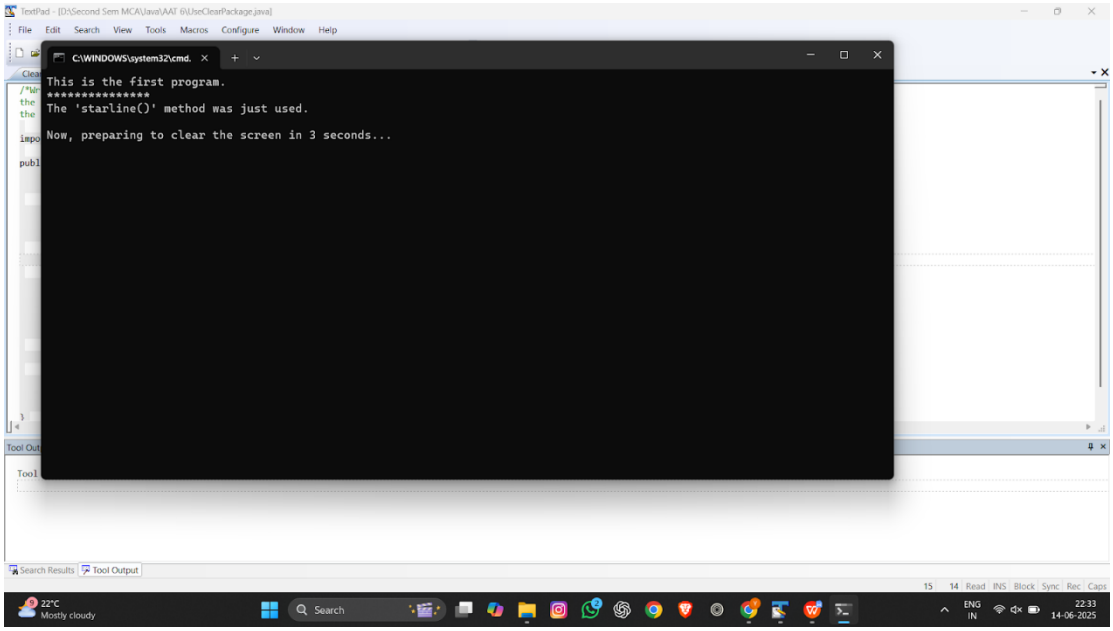
AAT6

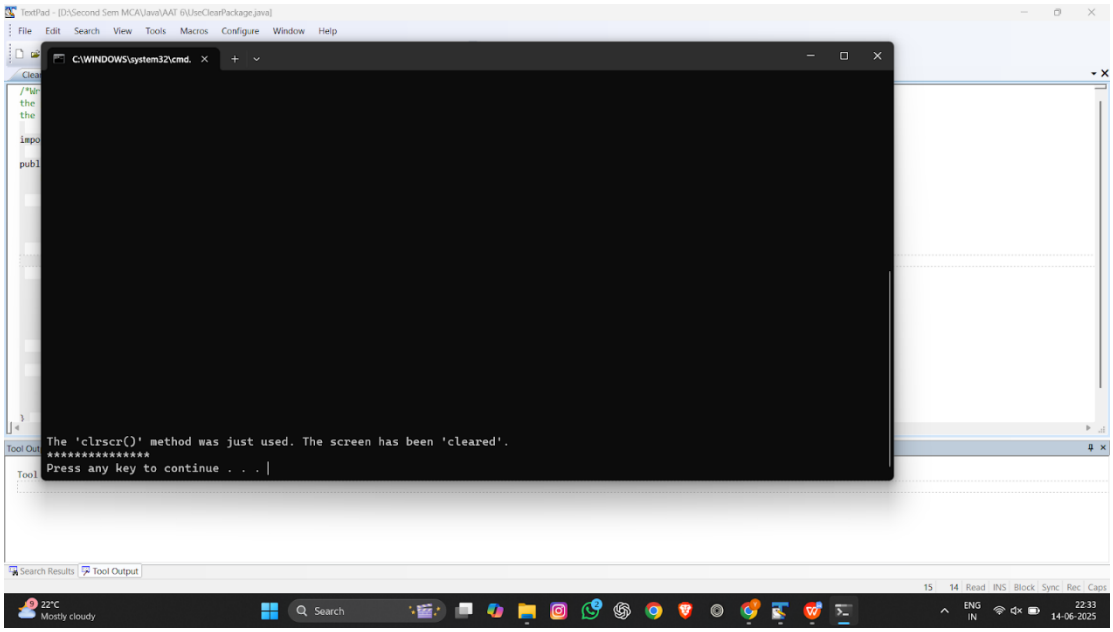
/*Write a package called Clear, it contains one public method clrscr() to clear the screen, import
the package and use it in another programs. Add another public method starline(). It prints
the line of 15 starts.*/

```
package Clear;

public class Clear {
/**
* This method attempts to clear the console screen by printing multiple new lines.
* Note: This is a simple, platform-independent approach and may not work perfectly in all console environments.
*/
public void clrscr() {
for (int i = 0; i < 50; ++i) {
System.out.println();
}
}

/**
* This method prints a line of 15 asterisks (*) to the console.
*/
public void starline() {
System.out.println("***************");
}
}
```

```
import Clear.Clear;

public class UseClearPackage {
public static void main(String[] args) {
Clear myClear = new Clear();

System.out.println("This is the first program.");
myClear.starline();
System.out.println("The 'starline()' method was just used.");
```

```
System.out.println("\nNow, preparing to clear the screen in 3 seconds...");

try {
Thread.sleep(3000);
} catch (InterruptedException e) {
Thread.currentThread().interrupt();
}

myClear.clrscr();

System.out.println("The 'clrscr()' method was just used. The screen has been
'cleared'.");
myClear.starline();
}
}
```





/*Defi ne an exception called " No Equal Exception " that is thrown when a float value is not equal to 3.14. Write a program that uses the above user defi ned exception.*/

```
import java.util.Scanner;

public class CheckFloatValue {

/**
* Checks if the given float value is equal to 3.14.
* @param value The float value to check.
* @throws NoEqualException if the value is not 3.14.
*/
public static void checkValue(float value) throws NoEqualException {
if (Math.abs(value - 3.14f) > 0.00001) { // Use a tolerance for float comparison
throw new NoEqualException("The float value (" + value + ") is not equal to 3.14");
}
System.out.println("Success! The float value " + value + " is equal to 3.14");
}

public static void main(String[] args) {
// --- Test with hardcoded values ---
System.out.println("--- Testing with hardcoded values ---");
try {
System.out.println("\nTesting with value: 5.5f");
checkValue(5.5f);
} catch (NoEqualException e) {
System.out.println("Caught expected exception: " + e.getMessage());
}

try {
System.out.println("\nTesting with value: 3.14f");
checkValue(3.14f);
} catch (NoEqualException e) {
```

```
System.out.println("Caught unexpected exception: " + e.getMessage());
}

// --- Test with user input ---
System.out.println("\n--- Testing with user input ---");
Scanner scanner = new Scanner(System.in);
System.out.print("Enter a float value (e.g., 3.14): ");

try {
float inputValue = scanner.nextFloat();
checkValue(inputValue);
} catch (NoEqualException e) {
System.out.println("Caught the exception: " + e.getMessage());
} catch (Exception e) {
System.out.println("Invalid input. Please enter a valid float value.");
} finally {
scanner.close();
}
}
}

/**
* A custom exception that is thrown when a float value is not equal to 3.14.
*/
public class NoEqualException extends Exception {
public NoEqualException(String message) {
super(message);
}
}
```

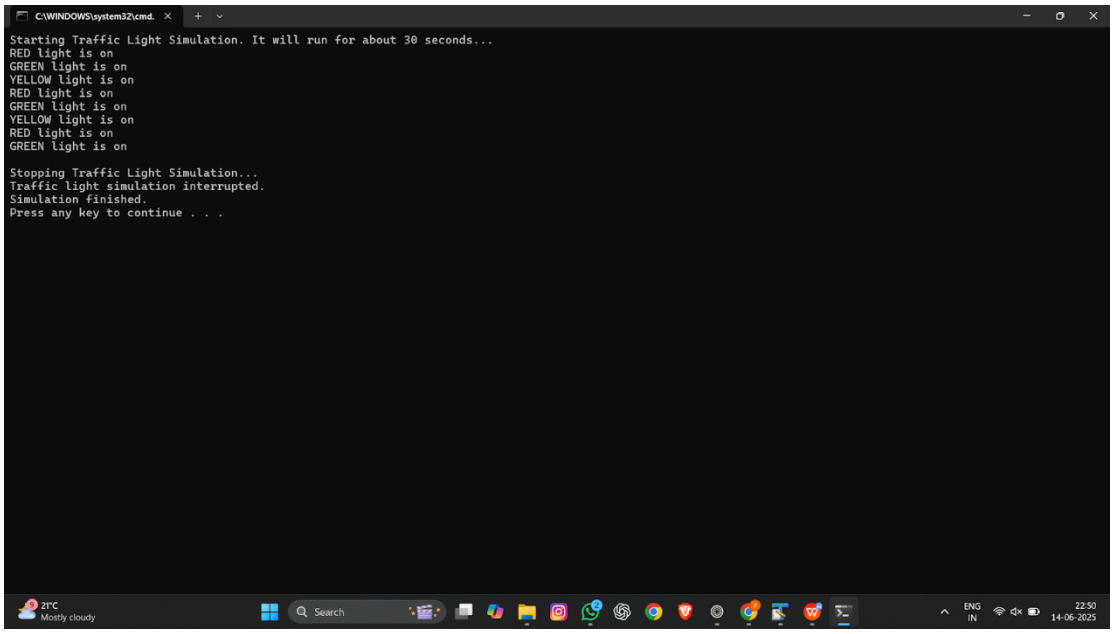
```
--- Testing with hardcoded values ---

Testing with value: 5.5f
Caught expected exception: The float value (5.5) is not equal to 3.14

Testing with value: 3.14f
Success! The float value 3.14 is equal to 3.14

--- Testing with user input ---
Enter a float value (e.g., 3.14): 1.5
Caught the exception: The float value (1.5) is not equal to 3.14
Press any key to continue . . .
```

/*Write a java program using threads to simulate traffic lights switch between Red, Green, and
Yellow with fixed delays.*/

```
enum LightColor {
RED, GREEN, YELLOW
}

class TrafficLightThread extends Thread {
private LightColor currentColor = LightColor.RED;
private volatile boolean running = true; // Use volatile for thread safety
```

```
@Override
public void run() {
while (running) {
try {
switch (currentColor) {
case RED:
System.out.println("RED light is on");
Thread.sleep(5000); // Red for 5 seconds
currentColor = LightColor.GREEN;
break;
case GREEN:
System.out.println("GREEN light is on");
Thread.sleep(5000); // Green for 5 seconds
currentColor = LightColor.YELLOW;
break;
case YELLOW:
System.out.println("YELLOW light is on");
Thread.sleep(2000); // Yellow for 2 seconds
currentColor = LightColor.RED;
break;
}
} catch (InterruptedException e) {
// If the thread is interrupted, stop the loop gracefully
running = false;
System.out.println("Traffic light simulation interrupted.");
Thread.currentThread().interrupt(); // Preserve the interrupted status
}
}
System.out.println("Simulation finished.");
}

public void stopSimulation() {
running = false;
// Interrupt the thread to make it stop sleeping immediately
this.interrupt();
}
}

public class TrafficLightSimulator {
public static void main(String[] args) {
TrafficLightThread trafficLight = new TrafficLightThread();
```

```
System.out.println("Starting Traffic Light Simulation. It will run for about 30 seconds...");
trafficLight.start();

try {
// Let the simulation run for 30 seconds
Thread.sleep(30000);
} catch (InterruptedException e) {
e.printStackTrace();
}

System.out.println("\nStopping Traffic Light Simulation...");
trafficLight.stopSimulation();
}
}
```

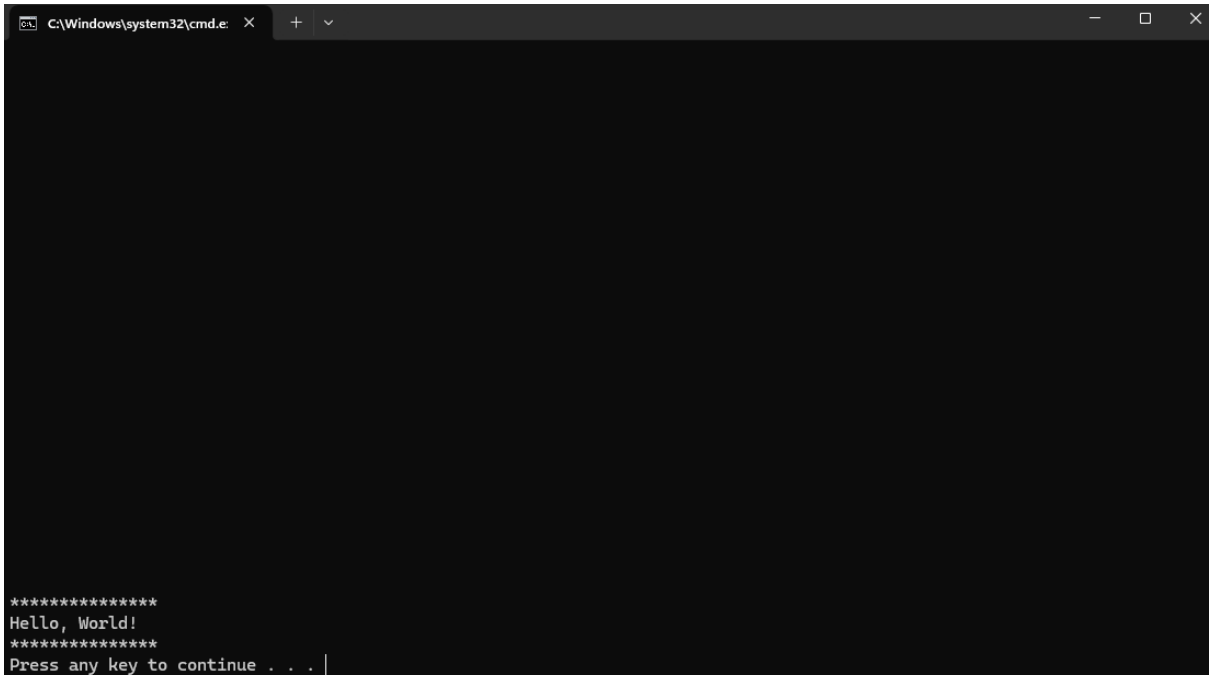


AAT 7

/*Write a package called Clear, it contains one public method clrscr() to clear the screen, import the package and use it in another programs. Add another public method starline(). It prints the line of 15 stars.*/

```
package Clear;

public class ClearScreen {
// Method to clear the screen (simulated by printing new lines)
public static void clrscr() {
for (int i = 0; i < 50; i++) {
System.out.println();
}
}

// Method to print a line of 15 stars
public static void starline() {
System.out.println("***************"); // 15 stars
}
}

import Clear.ClearScreen;

public class MainProgram {
public static void main(String[] args) {
ClearScreen.clrscr();        // Clear screen
ClearScreen.starline();      // Print 15 stars
System.out.println("Hello, World!");
ClearScreen.starline();      // Again print 15 stars
}
}
```

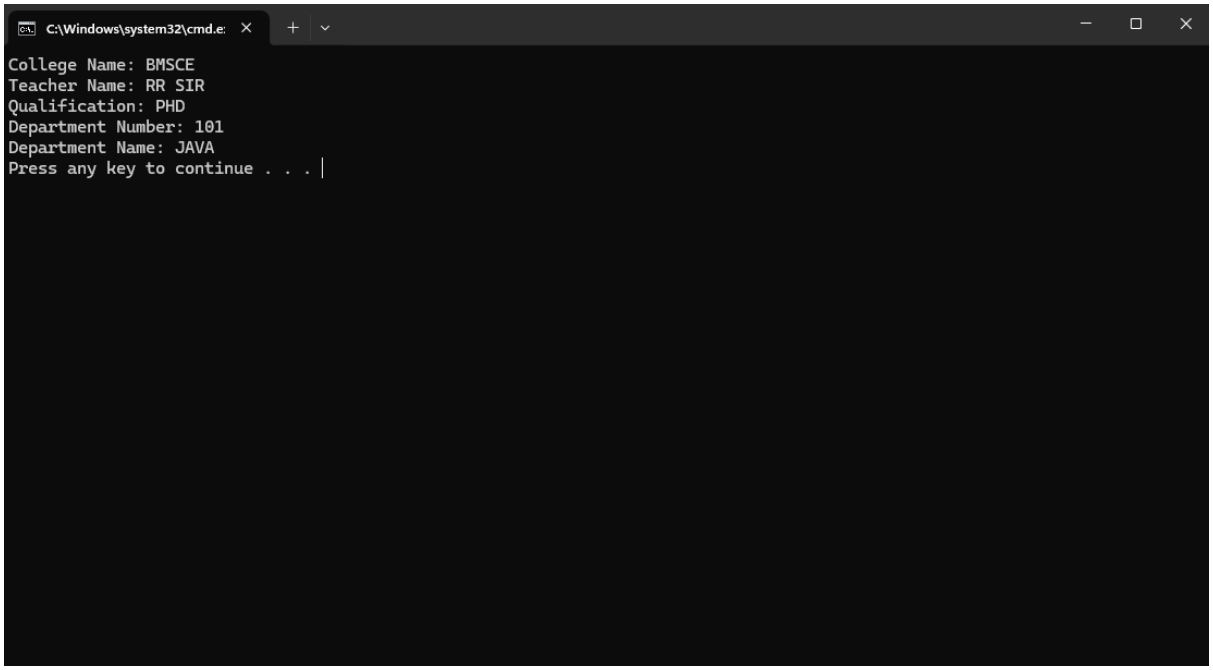
Output:-



/*Write a program in Java. A class Teacher contains two fi elds, Name and Qualifi cation. Extend the class to Department, it contains Dept. No and Dept. Name. An interface named as College contains one fi eld Name of the college. Using the above classes and Interface get the appropriate information and display it.*/

```
interface College {
String collegeName = "BMSCE"; // constant by default
}

// Base Class
class Teacher {
String name;
String qualification;

Teacher(String name, String qualification) {
this.name = name;
this.qualification = qualification;
}
}
```

```
// Derived Class
class Department extends Teacher implements College {
int deptNo;
String deptName;

Department(String name, String qualification, int deptNo, String deptName) {
super(name, qualification);
this.deptNo = deptNo;
this.deptName = deptName;
}

void displayInfo() {
System.out.println("College Name: " + collegeName);
System.out.println("Teacher Name: " + name);
System.out.println("Qualification: " + qualification);
System.out.println("Department Number: " + deptNo);
System.out.println("Department Name: " + deptName);
}
}

// Main class
public class Main {
public static void main(String[] args) {
Department obj = new Department("RR SIR", "PHD", 101, "JAVA");
obj.displayInfo();
}
}
```



/*Write and run the following Java program that does the following:

a) Declare a string object named s1 containing the string "Object Oriented Programming-Java 5".

b) Print the entire string.

c) Use the length() method to find the length of the string.

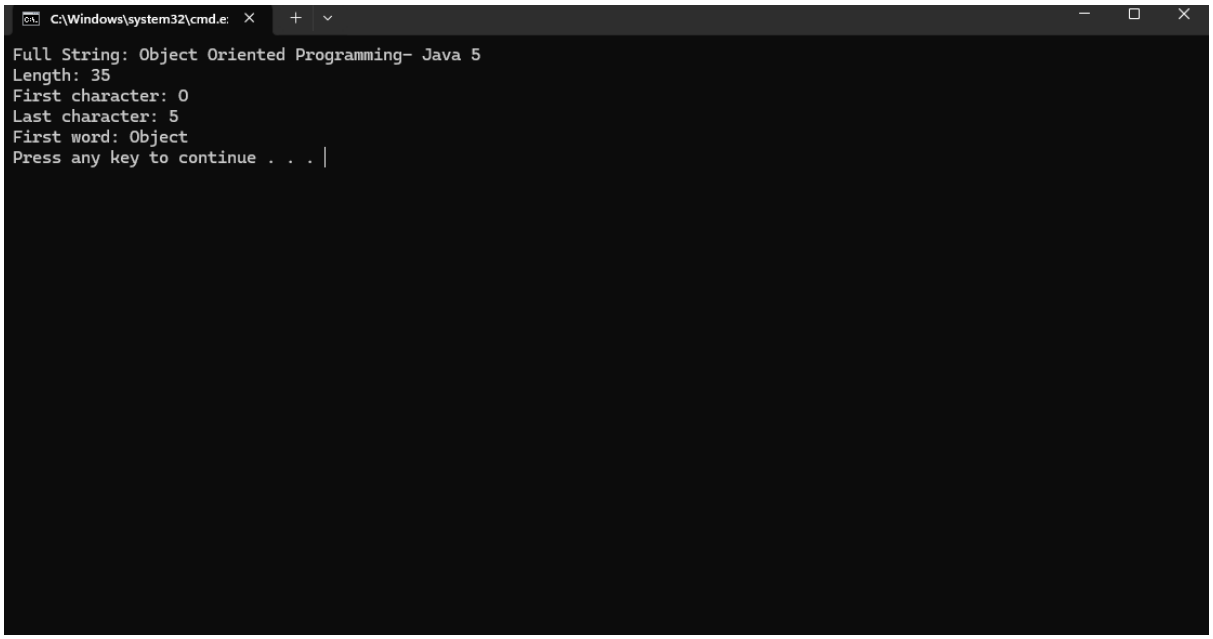
d) Use the charAt() method to find the first character in the string.

e) Use charAt() and length() methods to print the last character in the string.

f) Use the indexOf() and the substring() method to print the first word in the String.*/

```
public class StringOperations {
public static void main(String[] args) {
// a) Declare string object s1
String s1 = "Object Oriented Programming- Java 5";

// b) Print the entire string
System.out.println("Full String: " + s1);

// c) Length of the string
System.out.println("Length: " + s1.length());
```

```
// d) First character using charAt()
System.out.println("First character: " + s1.charAt(0));

// e) Last character using charAt() and length()
System.out.println("Last character: " + s1.charAt(s1.length() - 1));

// f) First word using indexOf() and substring()
int spaceIndex = s1.indexOf(" ");
String firstWord = s1.substring(0, spaceIndex);
System.out.println("First word: " + firstWord);
}
}
```



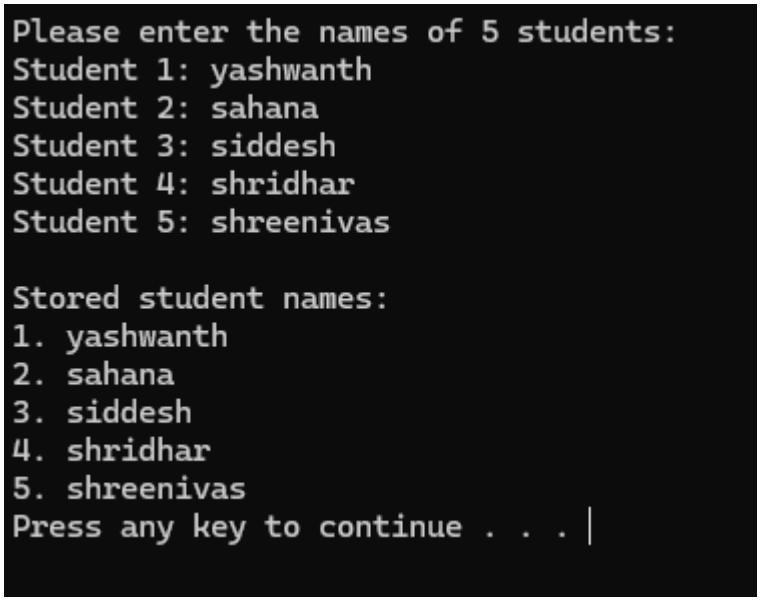
AAT 8

1. Write a program that accepts a name list of five students from the command line and store in a vector.

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

public class StudentList {
public static void main(String[] args) {
Vector<String> studentNames = new Vector<>();
Scanner input = new Scanner(System.in);

System.out.println("Please enter the names of 5 students:");

int count = 0;
while (count < 5) {
System.out.print("Student " + (count + 1) + ": ");
String student = input.nextLine();
studentNames.addElement(student);
count++;
}

System.out.println("\nStored student names:");
int index = 1;
for (String s : studentNames) {
System.out.println(index + ". " + s);
index++;
}

input.close();
}
```

}

OUTPUT:-



/*3. Define an exception called " No Equal Exception " that is thrown when a float value is not equal to 3.14. Write a program that uses the above user defined exception.*/

```
import java.util.Scanner;

/*  1. Custom checked exception  */
class NotEqualException extends Exception {
public NotEqualException(String detail) {
super(detail);
}
}

/*  2. Demo class with main()  */
public class FloatChecker {
```

```
// Utility method that throws the user-defined exception
private static void verify(float n) throws NotEqualException {
final float TARGET = 3.14f;

// Use Float.compare for clarity
if (Float.compare(n, TARGET) != 0) {
throw new NotEqualException("?  Input " + n + " ? " + TARGET);
}
}

public static void main(String[] args) {
try (Scanner kb = new Scanner(System.in)) {
System.out.print("Enter a float value ? ");
float userInput = kb.nextFloat();

/* Attempt verification */
verify(userInput);
System.out.println("?  Great! The number equals 3.14");
}
/* Handle the custom exception */
catch (NotEqualException ex) {
System.out.println("Exception caught ? " + ex.getMessage());
}
}
}
```

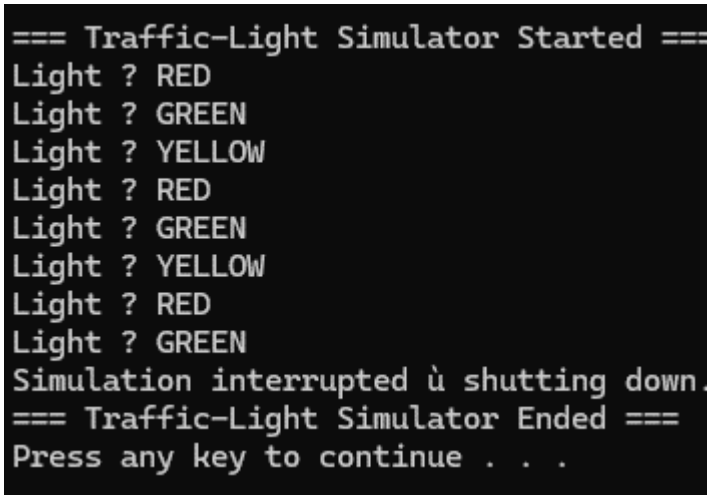
```
Enter a float value ? 1.1
Exception caught ? ?  Input 1.1 ? 3.14
Press any key to continue . . .
```

/*4. Write a java program using threads to simulate traffic lights switch between Red, Green,and Yellow with fixed delays.

*/

```
public class TrafficSignalDemo {

/** Enum stores both the colour name and its delay (ms). */
private enum Light {
RED(3_000),
GREEN(3_000),
YELLOW(1_000);

final int delay;
Light(int delay) { this.delay = delay; }
}

/** Runnable that loops through the lights until interrupted. */
private static class SignalTask implements Runnable {
@Override
public void run() {
Light[] sequence = Light.values();
int idx = 0;

System.out.println("=== Traffic-Light Simulator Started ===");
while (!Thread.currentThread().isInterrupted()) {
Light current = sequence[idx];
System.out.println("Light ? " + current);
idx = (idx + 1) % sequence.length;
```

```
try { Thread.sleep(current.delay); }
catch (InterruptedException ex) {
System.out.println("Simulation interrupted — shutting down.");
Thread.currentThread().interrupt();      // preserve flag
}
}
}
}

/** Main method: spin the thread for ~20 s then stop it. */
public static void main(String[] args) {
Thread worker = new Thread(new SignalTask(), "Signal-Thread");
worker.start();

try { Thread.sleep(20_000); }              // run for 20 s
catch (InterruptedException ignored) {}

worker.interrupt();                        // ask it to stop
try { worker.join(); } catch (InterruptedException ignored) {}

System.out.println("=== Traffic-Light Simulator Ended ===");
}
}
```




```
/*
5. Write a Java program to accept two parameters on the command line.
If there are no command line arguments entered, the program should print the error message and exit.
The program should check whether the first file exists and if it is an ordinary file.
If it is so, then the content is copied to the second file.
*/
import java.io.*;
import java.nio.file.*;


public class SimpleFileCopier {


public static void main(String[] args) {
/* 1. Resolve file names (CLI or default) */
String srcName, dstName;


if (args.length == 2) {
srcName = args[0];
dstName = args[1];
} else if (args.length == 0) {       // default pair
```

```
srcName = "1.txt";
dstName = "2.txt";
System.out.println("No command-line arguments ? using defaults:");
System.out.println("  Source      : " + srcName);
System.out.println("  Destination : " + dstName);
} else {                        // wrong count
System.err.println("?  Please pass **exactly** two file names, or none for defaults.");
System.exit(1);
return;
}

/* 2. Validate the source file */
Path src = Paths.get(srcName);
if (!Files.exists(src) || !Files.isRegularFile(src)) {
System.err.println("?  '" + srcName + "' does not exist or is not a regular file.");
System.exit(1);
}

/* 3. Perform the copy in 4 KiB chunks (buffered) */
Path dst = Paths.get(dstName);
try (InputStream  in  = new BufferedInputStream(Files.newInputStream(src));
OutputStream out = new BufferedOutputStream(Files.newOutputStream(dst))) {

byte[] buf = new byte[4096];
int n;
while ((n = in.read(buf)) != -1) {
out.write(buf, 0, n);
}
System.out.println("?  Copied '" + srcName + "' ? '" + dstName + "' successfully.");
}
catch (IOException ex) {
```

```
System.err.println("?  I/O error while copying: " + ex.getMessage());
}
}
}
```

```
No command-line arguments ? using defaults:
  Source      : 1.txt
  Destination : 2.txt
?  Copied '1.txt' ? '2.txt' successfully.
Press any key to continue . . .
```

AAT 9

1. Write a Java program to check whether the file is readable, writable and

hidden. FileChecker.java

```
import java.io.File;

class FileChecker {
private File f;

FileChecker(String p) {
f = new File(p);
}

boolean exists() {
return f.exists();
}

String check() {
if (!exists()) return "File not found.";

StringBuilder res = new StringBuilder();
res.append("Readable: ").append(f.canRead() ? "Yes" : "No").append("\n");
res.append("Writable: ").append(f.canWrite() ? "Yes" : "No").append("\n");
res.append("Hidden: ").append(f.isHidden() ? "Yes" : "No");
return res.toString();
}
}
```

Main.java

```
import java.util.Scanner;

public class Main {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.print("Enter file path: ");
String p = sc.nextLine();

FileChecker fc = new FileChecker(p);
System.out.println(fc.check());

System.out.print("\nPress any key to continue . . . ");
```

```
try {
System.in.read();
} catch (Exception e) {}
}
}
```

Output:

```
Enter file path: test.txt
Readable: Yes
Writable: Yes
Hidden: No
Press any key to continue . . .
```

2. Write a program using Lambda expression to filter the Employee objects. Each employee has a name, department, and salary.
• Employees with salary > 50,000
• Employees in the "IT" department

Employee.java

```
class Employee {
String name;
String dept;
double sal;

Employee(String name, String dept, double sal) {
this.name = name;
this.dept = dept;
this.sal = sal;
}

@Override
public String toString() {
return name + " (" + dept + ", $" + sal + ")";
}
}
```

Main.java

```
import java.util.*;
import java.util.stream.Collectors;

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

```
List<Employee> empList = Arrays.asList(
new Employee("Alice", "IT", 60000),
new Employee("Bob", "HR", 45000),
new Employee("Charlie", "IT", 55000),
new Employee("David", "Sales", 70000),
new Employee("Eve", "IT", 49000)
);

// Filter: Salary > 50000
 Employee> highEarners = empList.stream()
.filter(e -> e.sal > 50000)
            .collect(Collectors.toList());

// Filter: Department is IT
List<Employee> itDept = empList.stream()
        .filter(e -> e.dept.equals("IT"))
        .collect(Collectors.toList());

// Output high earners
System.out.println("Employees with salary > $50,000:");
highEarners.forEach(System.out::println);

// Output IT department
System.out.println("\nEmployees in IT department:");
itDept.forEach(System.out::println);

// Pause before exit
System.out.print("\nPress any key to continue . . . ");
try {
System.in.read();
} catch (Exception e) {}
}
}
```

Output:

```
Employees with salary > $50,000:
Alice (IT, $60000.0)
Charlie (IT, $55000.0)
David (Sales, $70000.0)

Employees in IT department:
Alice (IT, $60000.0)
Charlie (IT, $55000.0)
Eve (IT, $49000.0)

Press any key to continue . . .
```

3. You're building a system that handles different types of items in an online store. You want to create an Inventory<T> class that can store any type of item (books, electronics, etc.).

Inventory.java

```
import java.util.ArrayList;
import java.util.List;

class Inventory<T> {
private List<T> items = new ArrayList<>();

void add(T item) {
items.add(item);
}

void show() {
for (T item : items) {
System.out.println(item);
}
}
}
```

Book.java

```
class Book {
String title;

Book(String title) {
this.title = title;
}

public String toString() {
return "Book: " + title;
}
}
```

Electronics.java

```
class Electronics {
String name;

Electronics(String name) {
this.name = name;
}

public String toString() {
return "Electronics: " + name;
}
}
```

Main.java

```
public class Main {
public static void main(String[] args) {
// Create and populate book inventory
```

```
Inventory<Book> bookInv = new Inventory<>();
bookInv.add(new Book("Java Programming"));
bookInv.add(new Book("Python Basics"));

// Create and populate electronics inventory
Inventory<Electronics> elecInv = new Inventory<>();
elecInv.add(new Electronics("Smartphone"));
elecInv.add(new Electronics("Laptop"));

// Display inventories
System.out.println("Books in inventory:");
bookInv.show();

System.out.println("\nElectronics in inventory:");
elecInv.show();

// Pause before exit
System.out.print("\nPress any key to continue . . . ");
try {
System.in.read();
} catch (Exception e) {}
}
}
```

Output:

4. Write a program using autoboxing to calculate the discounted prices for a shopping cart. The system stores prices as Double, but the calculations are done using double.

Cart.java

```
import java.util.ArrayList;
import java.util.List;

class Cart {
private List<Double> prices = new ArrayList<>();

void add(Double price) {
prices.add(price); // Autoboxing: double → Double
}
```

```
double total(double discountPercent) {
double total = 0.0;
for (Double p : prices) {
total += p; // Unboxing: Double → double
}
return total * (1 - discountPercent / 100);
}
}
```

Main.java

```
public class Main {
public static void main(String[] args) {
Cart c = new Cart();
// Add prices (Autoboxing: double -> Double)
c.add(19.99);
c.add(45.50);
c.add(99.00);

double discount = 10; // 10% off
double finalPrice = c.total(discount);

      System.out.println("Total before discount: $" + String.format("%.2f", finalPrice / (1 -
discount / 100)));
System.out.println("Discount applied: " + discount + "%");
System.out.println("Final price after discount: $" + String.format("%.2f", finalPrice));

// Pause before exit
System.out.print("\nPress any key to continue . . . ");
try {
System.in.read();
} catch (Exception e) {}
}
}
```

Output:

```
Total before discount: $164.49
Discount applied: 10%
Final price after discount: $148.04

Press any key to continue . . .
```

AAT-10

1. Write a program to create a calculator using event handling.

CODE-

```
import javax.swing.*;

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

public class Calculator extends JFrame implements ActionListener {
JTextField textField;
String operator = "";
double num1 = 0;

Calculator() {
setTitle("Calculator");
setSize(250, 350);
setDefaultCloseOperation(EXIT_ON_CLOSE);

textField = new JTextField();
textField.setFont(new Font("Arial", Font.PLAIN, 20));
textField.setHorizontalAlignment(SwingConstants.RIGHT);
textField.setEditable(false);
add(textField, BorderLayout.NORTH);

JPanel panel = new JPanel(new GridLayout(4, 4, 5, 5));
String[] buttons = {
"7", "8", "9", "/",
"4", "5", "6", "*",
"1", "2", "3", "-",
"0", "C", "=", "+"
};

for (String text : buttons) {

JButton btn = new JButton(text);
btn.setFont(new Font("Arial", Font.BOLD, 18));
btn.addActionListener(this);
panel.add(btn);
}

add(panel);
setVisible(true);
```

```
}

public void actionPerformed(ActionEvent e) {
String cmd = e.getActionCommand();

if (cmd.matches("[0-9]")) {
textField.setText(textField.getText() + cmd);
} else if (cmd.equals("C")) {
textField.setText("");
operator = "";
num1 = 0;
} else if (cmd.equals("=")) {
double num2 = Double.parseDouble(textField.getText());
switch (operator) {
case "+": textField.setText("" + (num1 + num2)); break;
case "-": textField.setText("" + (num1 - num2)); break;
case "*": textField.setText("" + (num1 * num2)); break;
case "/": textField.setText(num2 == 0 ? "Error" : "" + (num1 / num2)); break;
}
} else {
num1 = Double.parseDouble(textField.getText());
operator = cmd;
textField.setText("");

}
}

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

OUTPUT-

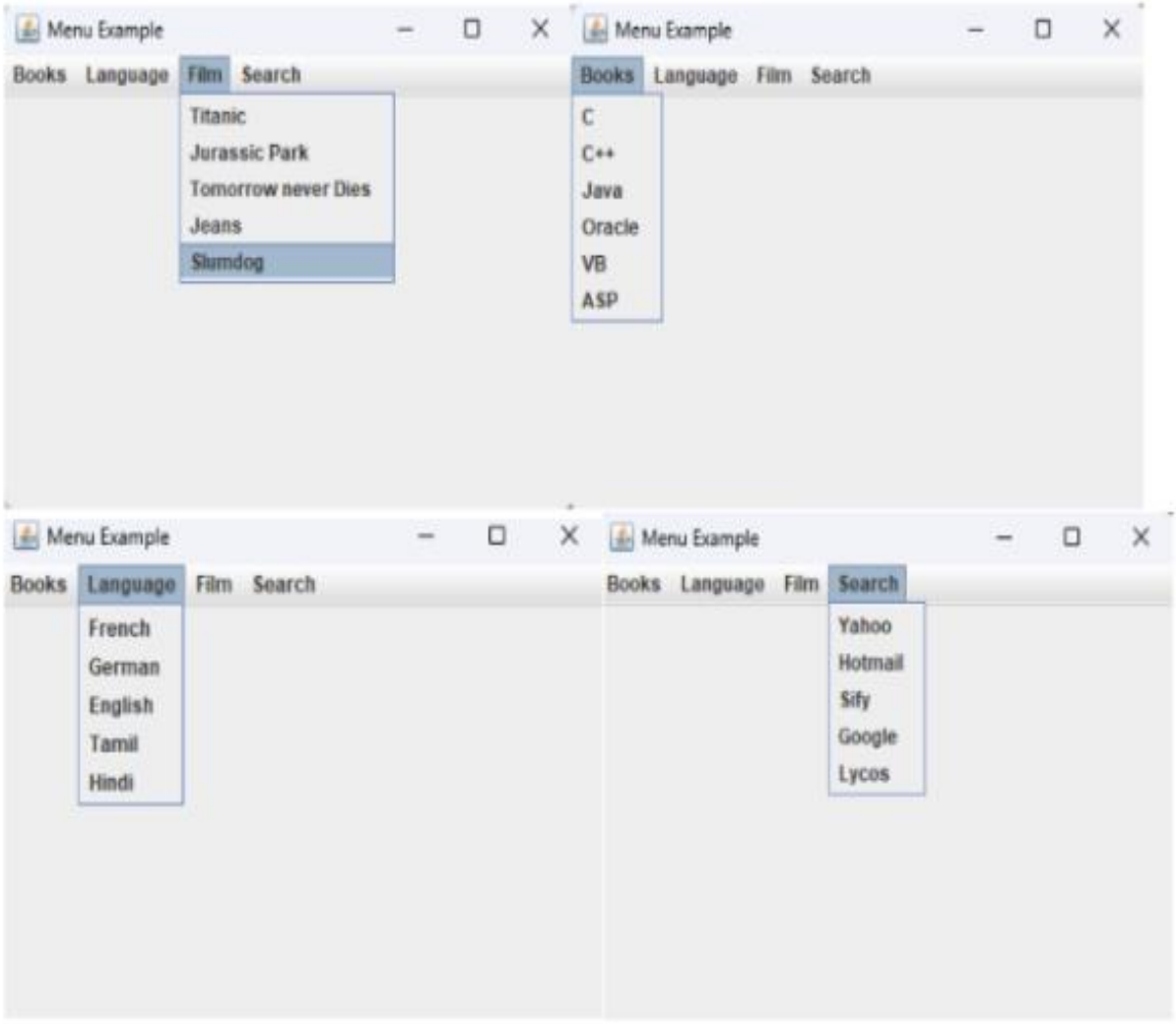
2-Create a menu is a frame as shown below:
Books Language Film Search
• The Books menu has the following items: C, C++, Java, Oracle, VB, and ASP.
• The language menu consists of French, German, English, Tamil, and Hindi.
• The File menu has the following: Titanic, Jurassic Park, Tomorrow never Dies, Jeans, and Slumdog.
• The search engine consists of Yahoo, Hotmail, Sify, Google, and Lycos.

CODE-

```
import javax.swing.*;

public class MenuExample extends JFrame {
public MenuExample() {
setTitle("Menu Example");
setSize(400, 300);
setDefaultCloseOperation(EXIT_ON_CLOSE);

JMenuBar menuBar = new JMenuBar();

JMenu booksMenu = new JMenu("Books");
String[] books = {"C", "C++", "Java", "Oracle", "VB", "ASP"};
for (String item : books) {
booksMenu.add(new JMenuItem(item));
}

JMenu languageMenu = new JMenu("Language");
String[] languages = {"French", "German", "English", "Tamil", "Hindi"};
for (String item : languages) {
languageMenu.add(new JMenuItem(item));
}

JMenu fileMenu = new JMenu("Film");
String[] films = {"Titanic", "Jurassic Park", "Tomorrow never Dies", "Jeans", "Slumdog"};
for (String item : films) {
fileMenu.add(new JMenuItem(item));
}

JMenu searchMenu = new JMenu("Search");
String[] searchEngines = {"Yahoo", "Hotmail", "Sify", "Google", "Lycos"};
for (String item : searchEngines) {
searchMenu.add(new JMenuItem(item));
}

menuBar.add(booksMenu);
menuBar.add(languageMenu);
```

```
menuBar.add(fileMenu);
menuBar.add(searchMenu);

setJMenuBar(menuBar);

setVisible(true);
}

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

OUTPUT-



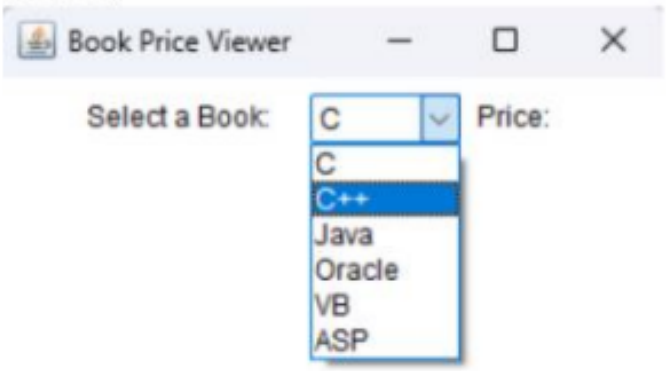
3. Write a Java program to display the different prices of the books using the choice object.

CODE-

```
import java.awt.*;

import java.awt.event.*;

public class BookPriceViewer extends Frame implements ItemListener {
Choice bookChoice;
Label priceLabel;
public BookPriceViewer() {
setTitle("Book Price Viewer");
setSize(300, 200);
setLayout(new FlowLayout());
setVisible(true);
Label label = new Label("Select a Book:");
add(label);
bookChoice = new Choice();
bookChoice.add("C");
bookChoice.add("C++");
bookChoice.add("Java");
bookChoice.add("Oracle");
bookChoice.add("VB");
bookChoice.add("ASP");
add(bookChoice);
priceLabel = new Label("Price: ");
add(priceLabel);
bookChoice.addItemListener(this);
addWindowListener(new WindowAdapter() {
public void windowClosing(WindowEvent e) {
dispose();
}
});
}
public void itemStateChanged(ItemEvent e) {
String selectedBook = bookChoice.getSelectedItem();
String price = "";
switch (selectedBook) {
case "C": price = "Rs. 200"; break;
case "C++": price = "Rs. 250"; break;
case "Java": price = "Rs. 300"; break;
case "Oracle": price = "Rs. 350"; break;
case "VB": price = "Rs. 220"; break;
case "ASP": price = "Rs. 270"; break;
}
```

```
priceLabel.setText("Price: " + price);
}
public static void main(String[] args) {
new BookPriceViewer();
}
}
```

OUTPUT-



4. Write Java program to display the different car names using list object.

CODE-

```
import java.awt.*;

import java.awt.event.*;
public class CarListViewer extends Frame implements ActionListener {
List carList;
Label selectedCarLabel;
public CarListViewer() {
setTitle("Car List Viewer");
setSize(300, 200);
setLayout(new FlowLayout());
setVisible(true);
Label label = new Label("Select a Car:");
add(label);
carList = new List(5, false);
carList.add("Toyota");
carList.add("Honda");
carList.add("Ford");
carList.add("BMW");
carList.add("Mercedes");
carList.add("Tesla");
add(carList);

selectedCarLabel = new Label("Selected Car: ");
add(selectedCarLabel);
carList.addActionListener(this);
addWindowListener(new WindowAdapter() {
public void windowClosing(WindowEvent e) {
dispose();
}
});
}
public void actionPerformed(ActionEvent e) {
String selectedCar = carList.getSelectedItem();
selectedCarLabel.setText("Selected Car: " + selectedCar);
}
public static void main(String[] args) {
new CarListViewer();
}
}
```

OUTPUT-


Car List Viewer
Select a Car:
Toyota
Honda
Ford
BMW
Mercedes
Selected Car:

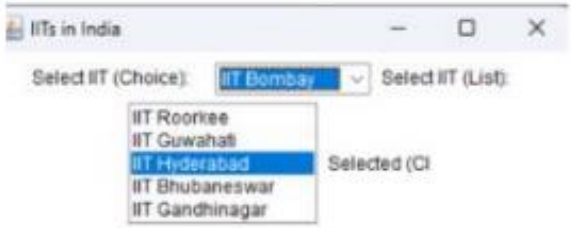
5. Write a Java program to display the different IIT's names in India using the choice object and list object.

CODE-

```
import java.awt.*;

import java.awt.event.*;
public class IITViewer extends Frame implements ItemListener, ActionListener {
Choice iitChoice;
List iitList;
Label selectedLabel;
public IITViewer() {
setTitle("IITs in India");
setSize(400, 300);

setLayout(new FlowLayout());
Label choiceLabel = new Label("Select IIT (Choice):");
add(choiceLabel);
iitChoice = new Choice();
iitChoice.add("IIT Bombay");
iitChoice.add("IIT Delhi");
iitChoice.add("IIT Madras");
iitChoice.add("IIT Kanpur");
iitChoice.add("IIT Kharagpur");
add(iitChoice);
Label listLabel = new Label("Select IIT (List):");
add(listLabel);
iitList = new List(5, false); // 5 visible rows, single selection
iitList.add("IIT Roorkee");
iitList.add("IIT Guwahati");
iitList.add("IIT Hyderabad");
iitList.add("IIT Bhubaneswar");
iitList.add("IIT Gandhinagar");
add(iitList);
selectedLabel = new Label("Selected: ");
add(selectedLabel);
iitChoice.addItemListener(this);
iitList.addActionListener(this);
addWindowListener(new WindowAdapter() {
public void windowClosing(WindowEvent we) {
dispose();
}
});
setVisible(true);
}
```

```
public void itemStateChanged(ItemEvent e) {
String selected = iitChoice.getSelectedItem();
selectedLabel.setText("Selected (Choice): " + selected);
}
public void actionPerformed(ActionEvent e) {
String selected = iitList.getSelectedItem();
selectedLabel.setText("Selected (List): " + selected);
}

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

OUTPUT-



6. Write a Java program to display the following 3 × 3 magic square (total = 15) using JTable:
2 9 4
7 5 3
6 1 8

CODE-

```
import javax.swing.*;

import java.awt.*;
public class MagicSquareTable extends JFrame {
public MagicSquareTable() {
setTitle("3x3 Magic Square");
setSize(300, 200);
setDefaultCloseOperation(EXIT_ON_CLOSE);
Object[][] data = {
{2, 9, 4},
{7, 5, 3},
{6, 1, 8}
};
String[] columnNames = {"", "", ""};
JTable table = new JTable(data, columnNames);
table.setRowHeight(40);
table.setEnabled(false); // Make it non-editable
table.setFont(new Font("Arial", Font.BOLD, 18));
table.setGridColor(Color.BLACK);
table.setShowGrid(true);

add(new JScrollPane(table));
setVisible(true);
}
public static void main(String[] args) {
new MagicSquareTable();
}
}
```

OUTPUT-