

**Department of Computer Science and Engineering**

---

**LAB MANUAL**

**Program: B. Tech. – AI & ML**

**Year & Semester: 2<sup>nd</sup> / IV Sem.**

**PCS-408 (Java Programming Lab)**

**Submitted by: Deepanshu Gupta**  
**University Roll No.: 2019472**

## **Vision and Mission of the Department of Computer Sc. and Engineering**

To impart quality education for producing world class technocrats and entrepreneurs with sound ethics, latest knowledge and innovative ideas in Computer Science and Engineering to meet industrial needs and societal expectations.

### **Mission**

M1. To impart world class value based technical education in all aspects of Computer Science and Engineering through state of the art infrastructure and innovative approach.

M2. To produce ethical, motivated and skilled engineers through theoretical knowledge and practical applications.

M3. To inculcate ability for tackling simple to complex individually as well as in a team.

M4. To develop globally competent engineers with strong foundations, capable of “out of the box” thinking to adapt to the rapidly changing scenarios requiring socially conscious green computing solutions.

### **Program Educational Objectives (PEOs)**

PEO 1. To produce students employable towards building a successful career based on sound understanding of theoretical and applied aspects as well as methodology to solve multidisciplinary real life problems.

PEO 2. To produce professional graduates ready to work with a sense of responsibility, ethics and enabling them to work efficiently individually and also as a team.

PEO 3. To impart the competency in students so that they are able to pursue higher studies and research in areas of engineering and other professionally related fields.

PEO 4. To inculcate ability to adapt to the changing technology through continuous learning.

## **Program Outcomes**

PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write

effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

### **PROGRAM SPECIFIC OUTCOMES (PSOs)**

PSO1. Ability to analyze, design, implement, and test software systems based on requirement specifications and development methodologies of software systems.

PSO2. Apply computer science theory blended with engineering mathematics to solve computational tasks and model real world problems using appropriate programming language, data structure, and algorithms.

PSO3. Ability to explore technological advancements in various domains, evaluate its merits and identify research gaps to provide solution to new ideas and innovations.

### **Course Outcomes**

After completion of the course students will be able to

1. Understand the object-oriented approach in programming along with the purpose and usage principles of inheritance, polymorphism, encapsulation and method overloading etc.
2. Demonstrate ability to test and debug Java programs using IDE
3. Analyze, design and develop small to medium sized application programs that demonstrate professionally acceptable programming standards
4. Demonstrate skills of developing event-driven programs using graphical user interfaces
5. Design applications for accessing databases using Java features.

# Problem Statements

## List of Programs

Topic	Concept and Problem statement	
(i)	Taking input from Command line and convert objects into primitive data type	Page No.
	1. Write a java program to take input as a command line argument. Your name, course, universityrollno and semester. Display the information. Name: UniversityRollNo: Course: Semester:	20
(ii)	Concepts of Java Control statements, Conditional statements, loops and iterations, Wrapper classes, Scanner Class	
	2. Using the switch statement, write a menu-driven program to calculate the maturity amount of a bank deposit. The user is given the following options: (i) Term Deposit (ii) Recurring Deposit  For option (i) accept Principal (p), rate of interest (r) and time period in years (n). Calculate and output the maturity amount (a) receivable using the formula $a = p[1 + r / 100]n$ .  For option (ii) accept monthly installment (p), rate of interest (r) and time period in months (n). Calculate and output the maturity amount (a) receivable using the formula $a = p * n + p * n(n + 1) / 2 * r / 100 * 1 / 12$ . For an incorrect option, an appropriate error message should be displayed. [ Use Scanner Class to take input ]	22
	3. Program to find if the given numbers are Friendly pair or not (Amicable or not). <b>Friendly Pair</b> are two or more numbers with a common abundance.  <b>Input &amp; Output format:</b> <ul style="list-style-type: none"> <li>• Input consists of <b>2 integers</b>.</li> <li>• The first integer corresponds to number 1 and the second integer corresponds to number 2.</li> <li>• If it is a Friendly Pair display <b>Friendly Pair</b> or displays <b>Not Friendly Pair</b>.</li> </ul> For example, <b>6</b> and <b>28</b> are <b>Friendly Pair</b> . (Sum of divisors of 6)/6 = (Sum of divisors of 28)/28. Steps to check whether the given numbers are friendly pair or not	25

		<ul style="list-style-type: none"> <li>• Input the numbers num1 and num2.</li> <li>• Initialize sum1 = sum2 = 0.</li> <li>• sum1 = sum of all divisors of num1.</li> <li>• sum2 = sum of all divisors of num2.</li> <li>• If (sum1 == num1) and (sum2 == num2), then print "<b>Abundant No.</b>".</li> <li>• Else, print "<b>Not Abundant No.</b>".</li> </ul> <p>Program to check whether the given numbers are friendly pair or not</p>	
	4.	<p>Program to replace all 0's with 1 in a given integer. Given an integer as an input, all the 0's in the number has to be replaced with 1.</p> <p>For example, consider the following number Input: 102405</p> <p>Output: 112415</p> <p>Input: 56004</p> <p>Output: 56114</p> <p>Steps to replace all 0's with 1 in a given integer</p> <ul style="list-style-type: none"> <li>• Input the integer from the user.</li> <li>• Traverse the integer digit by digit.</li> <li>• If a '0' is encountered, replace it by '1'.</li> <li>• Print the integer.</li> </ul>	27
(iii)	Array in Java		
	5.	<p>Printing an array into Zigzag fashion. Suppose you were given an array of integers, and you are told to sort the integers in a zigzag pattern. In general, in a zigzag pattern, the first integer is less than the second integer, which is greater than the third integer, which is less than the fourth integer, and so on. Hence, the converted array should be in the form of <math>e_1 &lt; e_2 &gt; e_3 &lt; e_4 &gt; e_5 &lt; e_6</math>.</p> <p>Test cases: Input 1: 7 4 3 7 8 6 2 1</p> <p>Output 1: 3 7 4 8 2 6 1</p> <p>Input 2: 4 1 4 3 2</p> <p>Output 2: 1 4 2 3</p>	29

	6.	<p>The problem to rearrange positive and negative numbers in an array .</p> <p>Method: This approach moves all negative numbers to the beginning and positive numbers to the end but changes the order of appearance of the elements of the array.</p> <p>Steps:</p> <ol style="list-style-type: none"> <li>1. Declare an array and input the array elements.</li> <li>2. Start traversing the array and if the current element is negative, swap the current element with the first positive element and continue traversing until all the elements have been encountered.</li> <li>3. Print the rearranged array.</li> </ol> <p>Test case:</p> <ul style="list-style-type: none"> <li>• Input: 1 -1 2 -2 3 -3</li> <li>• Output: -1 -2 -3 1 3 2</li> </ul>	32
	7.	<p>Program to find the saddle point coordinates in a given matrix. A saddle point is an element of the matrix, which is the minimum element in its row and the maximum in its column.</p> <p>For example, consider the matrix given below Mat[3][3]</p> <pre>1 2 3 4 5 6 7 8 9</pre> <p>Here, 7 is the saddle point because it is the minimum element in its row and maximum element in its column.</p> <p>Steps to find the saddle point coordinates in a given matrix</p> <ol style="list-style-type: none"> <li>1. Input the matrix from the user.</li> <li>2. Use two loops, one for traversing the row and the other for traversing the column.</li> <li>3. If the current element is the minimum element in its row and maximum element in its column, then return its coordinates.</li> <li>4. Else, continue traversing.</li> </ol>	34
(iv)		String Handling in Java (using String and StringBuffer class)	
	8.	<p>Program to find all the patterns of 0(1+)0 in the given string. Given a string containing 0's and 1's, find the total number of 0(1+)0 patterns in the string and output it.</p> <p>0(1+)0 - There should be at least one '1' between the two 0's.</p> <p>For example, consider the following string.</p> <p>Input: 01101111010 Output: 3 Explanation: 01101111010 - count = 1</p>	37



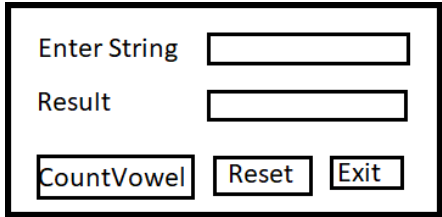
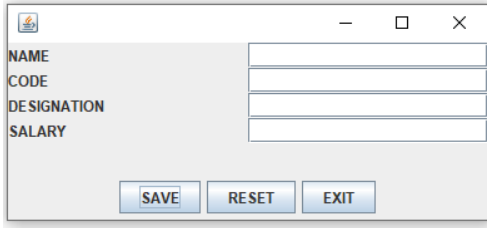
		0110 <b>1111</b> 010 - count = 2 011011110 <b>10</b> - count = 3  Step to find all the patterns of 0(1+)0 in the given string <ul style="list-style-type: none"> <li>• Input the given string.</li> <li>• Scan the string, character by character.</li> <li>• If the given pattern is encountered, incrementcount.</li> <li>• Print count.</li> </ul> Program to find all the patterns of 0(1+)0	
	9.	Write a java program to delete vowels from given string using StringBuffer class.	39
(v)		Class definition, creating objects and constructors	
	10.	Write a java program to create a class named ' <b>Bank</b> ' with the following data members: Name of depositor Address of depositor Account Number Balance in account  Class ' <b>Bank</b> ' has a method for each of the following: 1 - Generate a unique account number for each depositor For first depositor, account number will be 1001, forsecond depositor it will be 1002 and so on 2 - Display information and balance of depositor 3 - Deposit more amount in balance of any depositor 4 - Withdraw some amount from balance deposited 5 - Change address of depositor  After creating the class, do the following operations 1- Enter the information (name, address, account number, balance) of the depositors. Number of depositors is to be entered by user. 2 - Print the information of any depositor. 3 - Add some amount to the account of any depositorand then display the final information of that depositor 4 - Remove some amount from the account of any depositor and then display final information of that depositor. 5 - Change the address of any depositor and then display the final information of that depositor 6 - Randomly repeat these processes for some other bank accounts.	41

	11.	<p>Define a class <b>WordExample</b> having the following description: Data members/instance variables: private String strdata : to store a sentence.</p> <p><b>Parameterized Constructor WordExample(String) :</b> Accept a sentence which may be terminated by either '.', '? 'or '!' only. The words may be separated by more than one blank space and are in UPPER CASE.</p> <p><b>Member Methods:</b> <b>void countWord():</b> Find the number of words beginning and ending with a vowel. <b>void placeWord():</b> Place the words which begin and end with a vowel at the beginning, followed by the remaining words as they occur in the sentence.</p>	49
(vi)	Method overloading (Compile time Polymorphism)		
	12.	<p>Write a Java program to create a class called <b>ArrayDemo</b> and overload arrayFunc() function. <b>void arrayFunc(int [], int)</b> □ To find all pairs of elements in an Array whose sum is equal to a given number : Array numbers= [4, 6, 5, -10, 8, 5, 20], target=10</p> <p><b>Output:</b> Pairs of elements whose sum is 10 are : 4 + 6 = 10 5 + 5 = 10 -10 + 20 = 10</p> <p><b>void arrayFunc(int A[], int p, int B[], int q)</b> □ Given two sorted arrays A and B of size p and q, Overload method arrayFunc() to merge elements of A with B by maintaining the sorted order i.e. fill A with first p smallest elements and fill B with remaining elements.</p> <p>Example: Input : int[] A = { 1, 5, 6, 7, 8, 10 } int[] B = { 2, 4, 9 } Output: Sorted Arrays: A: [1, 2, 4, 5, 6, 7] B: [8, 9, 10]</p> <p>(Use Compile time Polymorphism MethodOverloading)</p>	54

(vii)	Method overriding (Runtime Polymorphism), Abstract class & Abstract method, Inheritance, interfaces		
	13.	Write a java program to calculate the area of a rectangle, a square and a circle. Create an abstract class ' <b>Shape</b> ' with three abstract methods namely <b>rectangleArea()</b> taking two parameters, <b>squareArea()</b> and <b>circleArea()</b> taking one parameter each. Now create another class ' <b>Area</b> ' containing all the three methods rectangleArea(), squareArea() and circleArea() for printing the area of rectangle, square and circle respectively. Create an object of class Area and call all the three methods. <b>(Use Runtime Polymorphism)</b>	58
	14.	Write a java program to implement abstract class and abstract method with following details:  <b>Create a abstract Base Class Temperature</b> <b>Data members:</b> double temp; <b>Method members:</b> void setTempData(double) abstract void changeTemp()  <b>Sub Class Fahrenheit (subclass of Temperature)</b> <b>Data members:</b> double ctemp; <b>Method member:</b> Override abstract method changeTemp() to convert Fahrenheit temperature into degree Celsius by using formula $C = 5/9 * (F - 32)$ and display converted temperature  <b>Sub Class Celsius (subclass of Temperature)</b> <b>Data member:</b> double ftemp; <b>Method member:</b> Override abstract method changeTemp() to convert degree Celsius into Fahrenheit temperature by using formula $F = 9/5 * c + 32$ and display converted temperature	60
	15.	Write a java program to create an interface that consists of a method to display volume () as an abstract method and redefine this method in the derived classes to suit their requirements.  Create classes called Cone, Hemisphere and Cylinder that implements the interface. Using these three classes, design a program that will accept dimensions of a cone, cylinder and hemisphere interactively and display the volumes.  Volume of cone = $(1/3)\pi r^2 h$ Volume of hemisphere = $(2/3)\pi r^3$ Volume of cylinder = $\pi r^2 h$	63

(viii)	Exception handling	
	<p>16. Write a java program to accept and print the employee details during runtime. The details will include employee id, name, dept_ Id. The program should raise an exception if user inputs incomplete or incorrect data. The entered value should meet the following conditions:</p> <ul style="list-style-type: none"> <li>(i) First Letter of employee name should be in capital letter.</li> <li>(ii) Employee id should be between 2001 and 5001</li> <li>(iii) Department id should be an integer between 1 and 5.</li> </ul> <p>If the above conditions are not met then the application should raise specific exception else should complete normal execution.</p>	66
	<p>17. Create a class MyCalculator which consists a method power(int, int). This method takes two integers, <b>n</b> and <b>p</b>, as parameters and finds <b>n<sup>p</sup></b>. If either n or p is negative, then the method must throw an exception which says "n and p should be non-negative".</p> <p><b>Input Format</b> Each line of the input contains two integers, n and p.</p> <p><b>Output Format</b> Each line of the output contains the result, if neither of n and p is negative. Otherwise the output contains "n and p should be non-negative".</p> <p><b>Sample Input</b>  3 5  2 4  0 0  -1 -2  -1 3</p> <p><b>Sample Output</b>  243  16  java.lang.Exception: n and p should not be zero. java.lang.Exception: n or p should not be negative. java.lang.Exception: n or p should not be negative.</p> <p><b>Explanation</b>  In the first two cases, both <b>n</b> and <b>p</b> are positive. So, the power function returns the answer correctly.  In the third case, both <b>n</b> and <b>p</b> are zero. So, the exception, "<b>n</b> and <b>p</b> should not be zero." is printed.  In the last two cases, at least one out of <b>n</b> and <b>p</b> is negative. So, the exception, "<b>n</b> or <b>p</b> should not be negative." is printed for these two cases.</p>	69

(ix)	File Handling in Java		
	18.	Write a java file handling program to count and display the number of palindrome present in a text file "myfile.txt". Example: If the file "myfile.txt" contains the following lines, My name is NITIN Hello aaa and bbb wordHow are You ARORA is my friendOutput will be => 4	72
(x)	Multithreaded programming		
	19.	Write a program MultiThreads that creates two threads-one thread with the name CSthread and the other thread named ITthread. Each thread should display its respective name and execute after a gap of 500 milliseconds. Each thread should also display a number indicating the number of times it got a chance to execute.	74
	20.	Write a java program for to solve producer consumer problem in which a producer produce a value and consumer consume the value before producer generate the next value.	77
(xi)	Collection and Generic Framework:		
	21.	Write a method removeEvenLength that takes an ArrayList of Strings as a parameter and that removes all of the strings of even length from the list.	
	22.	Write a method swapPairs that switches the order of values in an ArrayList of Strings in a pairwise fashion. Your method should switch the order of the first two values, then switch the order of the next two, switch the order of the next two, and so on.  For example, if the list initially stores these values: {"four", "score", "and", "seven", "years", "ago"} your method should switch the first pair, "four", "score", the second pair, "and", "seven", and the third pair, "years", "ago", to yield this list: {"score", "four", "seven", "and", "ago", "years"}  If there are an odd number of values in the list, the final element is not moved.  For example, if the original list had been: {"to", "be", "or", "not", "to", "be", "hamlet"} It would again switch pairs of values, but the final value, "hamlet" would not be moved, yielding this list: {"be", "to", "not", "or", "be", "to", "hamlet"}	83
	23.	Write a method called alternate that accepts two Lists of integers as its parameters and returns a new List containing alternating elements from the two lists, in the following order: <ul style="list-style-type: none"><li>• First element from first list</li><li>• First element from second list</li></ul>	86

		<ul style="list-style-type: none"> <li>• Second element from first list</li> <li>• Second element from second list</li> <li>• Third element from first list</li> <li>• Third element from second list</li> </ul> <p>If the lists do not contain the same number of elements, the remaining elements from the longer list should be placed consecutively at the end. For example, for a first list of (1, 2, 3, 4, 5) and a second list of (6, 7, 8, 9, 10, 11, 12), a call of alternate(list1, list2) should return a list containing (1, 6, 2, 7, 3, 8, 4, 9, 5, 10, 11, 12). Do not modify the parameter lists passed in.</p>	
(xii)	AWT & Swing, Event Handling		
	24.	<p>Write a GUI program to develop an application that receives a string in one text field, and count number of vowels in a string and returns it in another text field, when the button named “CountVowel” is clicked.</p> <p>When the button named “Reset” is clicked it will reset the value of textfield one and Textfield two .</p> <p>When the button named “Exit” is clicked it will close the application.</p> 	88
(xi)	Java Database Connectivity (JDBC)		
	25.	<p>Create a database of employee with the following fields:</p> <p>Name Code Designation Salary</p>  <p>(a) Write a java program to create GUI java application to take employee data from the TextFields and store in database using JDBC connectivity. (b) Write a JDBC Program to retrieve all the records from employee database.</p>	92

## **Typical solutions and output**

## Problem Statement 1 and solution

### •Problem Statement

Write a Java program to take input as a command line argument. Your name, course, university roll number, and semester. Display the information.

### •Theory, important classes, and methods

#### **System.out.println()**

Java **System.out.println()** is used to print an argument that is passed to it.

#### **class Integer**

The Integer class wraps a value of the primitive type int in an object.

#### **parseInt(String s)**

Parses the string argument as a signed decimal integer.

### **PROGRAM:**

```
public class StudentInfo {  
    public static void main(String[] args) {  
        String name;  
        String urollno;  
        String course;  
        int sem;  
        name=args[0];  
        urollno=args[1];  
        course=args[2];  
        sem=Integer.parseInt(args[3]);  
        System.out.println("Name: "+name);  
        System.out.println("University Roll Number: "+urollno);  
        System.out.println("Course: "+course);  
        System.out.println("Semester: "+sem);  
    }  
}
```

### **Output:**

```
Name: Raj  
University Roll Number: 210211  
Course: BTech  
Semester: 4
```



## Problem Statement 2 and solution

### •Problem Statement

Using the switch statement, write a menu-driven program to calculate the maturity amount of a bank deposit. The user is given the following options:

- (i) Term Deposit
- (ii) Recurring Deposit

### •Theory, important classes and methods

#### **class Scanner**

Scanner is a class in java.util package used for obtaining the input of the primitive types like int, double, etc. and strings.

#### **nextInt()**

Scans the next token of the input as an int.

#### **nextDouble()**

Scans the next token of the input as a double.

### **PROGRAM:**

```
import java.util.Scanner;
public class BankDeposit {
    public static void main(String[] args) {
        int choice=0;
        double maturityAmount=0;
        Scanner sc=new Scanner(System.in);
        System.out.println("Type 1: Term Deposit");
        System.out.println("Type 2: Recurring Deposit");
        System.out.print("Enter your choice: ");
        choice=sc.nextInt();
        switch (choice) {
            case 1:
                System.out.print("Enter Principal: ");
                double P1=sc.nextDouble();
                System.out.print("Enter Rate of Interest: ");
                double r1=sc.nextDouble();
                System.out.print("Enter Time (in years): ");
                double n1=sc.nextDouble();
                maturityAmount=P1*Math.pow(1+r1/100.0, n1);
                System.out.println("Maturity Amount = "+maturityAmount);
            break;
```

```

        case 2:
            System.out.print("Enter Monthly Installment: ");
            double P2=sc.nextDouble();
            System.out.print("Enter Rate of Interest: ");
            double r2=sc.nextDouble();
            System.out.print("Enter Period (in months): ");
            double n2=sc.nextDouble();
            maturityAmount=P2*n2+P2*(n2*(n2+1)/2.0)*(r2/100.0)*(1.0/12);
            System.out.println("Maturity Amount = "+maturityAmount);
            break;
        default:
            System.out.println("Invalid Choice");
    }
}
}

```

### **Output 1:**

```

Type 1: Term Deposit
Type 2: Recurring Deposit
Enter your choice: 1
Enter Principal: 50000
Enter Rate of Interest: 5
Enter Time (in years): 5
Maturity Amount = 63814.078125000015

```

### **Output 2:**

```

Type 1: Term Deposit
Type 2: Recurring Deposit
Enter your choice: 2
Enter Monthly Installment: 40000
Enter Rate of Interest: 4
Enter Period (in months): 4
Maturity Amount = 161333.33333333334

```

## Problem Statement 3 and solution

### •Problem Statement

Program to find if the given numbers are Friendly pair or not (Amicable or not). **Friendly Pair** are two or more numbers with a common abundance.

For example, 6 and 28 are Friendly Pair.  $(\text{Sum of divisors of } 6)/6 = (\text{Sum of divisors of } 28)/28$ .

### •Theory, important classes and methods

#### class Scanner

Scanner is a class in java.util package used for obtaining the input of the primitive types like int, double, etc. and strings.

#### nextInt()

Scans the next token of the input as an int.

**Java for loop** provides a concise way of writing the loop structure. The for statement consumes the initialization, condition and increment/decrement in one line thereby providing a shorter, easy to debug structure of looping.

#### Syntax:

```
for (initialization expr ; test expr ; update exp) {  
    // body of the loop  
}
```

#### PROGRAM:

```
import java.util.*;  
public class FriendlyPair {  
    public static void main(String[] args) {  
        int i, num1, num2;  
        Scanner sr = new Scanner(System.in);  
        System.out.print("Enter First Number: ");  
        num1 = sr.nextInt();  
        System.out.print("Enter Second Number: ");  
        num2 = sr.nextInt();  
        int sum1 = 0, sum2 = 0;  
        for (i = 1; i < num1; i++) {  
            if (num1 % i == 0)  
                sum1 = sum1 + i;  
        }  
        for (i = 1; i < num2; i++) {  
            if (num2 % i == 0)  
                sum2 = sum2 + i;  
        }  
    }  
}
```

```
        if (sum1==num1 && sum2==num2)
            System.out.print("Friendly Pair");
        else
            System.out.print("Not Friendly Pair");
    }
}
```

**Output:**

Enter First Number: 6  
Enter Second Number: 28  
Friendly Pair

## Problem Statement 4 and solution

### •Problem Statement

Program to replace all 0's with 1 in a given integer. Given an integer as an input, all the 0's in the number has to be replaced with 1.

For example, consider the following number Input: 102405

Output: 112415

Input: 56004

Output: 56114

### •Theory, important classes and methods

#### **class Scanner**

Scanner is a class in java.util package used for obtaining the input of the primitive types like int, double, etc. and strings.

#### **nextInt()**

Scans the next token of the input as an int.

**Java While loop** starts with the checking of condition. If it evaluated to true, then the loop body statements are executed otherwise first statement following the loop is executed. For this reason it is also called **Entry control loop**

#### **Syntax :**

```
while (boolean condition)
{
// loop statements...
}
```

#### **PROGRAM:**

```
import java.util.*;
public class ZeroOne {
    public static void main(String[] args) {
        int number, tnum;
        Scanner sr=new Scanner(System.in);
        System.out.print("Enter the number: ");
        number=sr.nextInt();
        tnum=number;
        int result=0;
        int decimalPlace=1;
```

```
    if (number==0)
        result += (1*decimalPlace);
    while (number>0) {
        if (number%10==0)
            result += (1*decimalPlace);
        number /= 10;
        decimalPlace *= 10;
    }
    tnum += result;
    System.out.print("Result: "+tnum);
}
}
```

**Output:**

Enter the number: 530012  
Result: 531112

## Problem Statement 5 and solution

### •Problem Statement

Printing an array into Zigzag fashion. Suppose you were given an array of integers, and you are told to sort the integers in a zigzag pattern. In general, in a zigzag pattern, the first integer is less than the second integer, which is greater than the third integer, which is less than the fourth integer, and so on. Hence, the converted array should be in the form of  $e_1 < e_2 > e_3 < e_4 > e_5 < e_6$ .

### •Theory, important classes and methods

#### Arrays in Java

An array is a group of like-typed variables that are referred to by a common name.

#### Field length

length is a final variable applicable for arrays. With the help of length variable, we can obtain the size of the array.

#### PROGRAM:

```
import java.util.*;
class ZigZag {
    public static void main(String[] args) {
        Scanner sr=new Scanner(System.in);
        int n, i;
        System.out.print("Enter the size: ");
        n=sr.nextInt();
        int arr[]=new int[n];
        System.out.println("Enter elements:");
        for (i=0; i<arr.length; i++)
            arr[i]=sr.nextInt();
        System.out.println("Original array:");
        for (i=0; i<arr.length; i++)
            System.out.print(arr[i]+" ");
        boolean flag=true;
        int temp=0;
        for (i=0; i<=arr.length-2; i++) {
            if (flag && arr[i]>arr[i+1]) {
                temp=arr[i];
                arr[i]=arr[i+1];
                arr[i+1]=temp;
            }
            else if (arr[i]<arr[i+1]) {
                temp=arr[i];
```

```

        arr[i]=arr[i+1];
        arr[i+1]=temp;
    }
    flag=!flag;
}
System.out.println("Rearranged array :");
for (i=0; i<arr.length; i++)
    System.out.print(arr[i]+" ");
}
}

```

**Output:**

```

Enter the size: 5
Enter elements:
3 5 8 6 9
Original array:
3 5 8 6 9
Rearranged array :
5 8 6 9 3

```



## Problem Statement 6 and solution

### •Problem Statement

The problem to rearrange positive and negative numbers in an array .

**Method:** This approach moves all negative numbers to the beginning and positive numbers to the end but changes the order of appearance of the elements of the array.

### Steps:

1. Declare an array and input the array elements.
2. Start traversing the array and if the current element is negative, swap the current element with the first positive element and continue traversing until all the elements have been encountered.
3. Print the rearranged array.

### Test case:

- Input: 1 -1 2 -2 3 -3
- Output: -1 -2 -3 1 3 2

### •Theory, important classes and methods

#### Arrays in Java

An array is a group of like-typed variables that are referred to by a common name.

#### class Scanner

Scanner is a class in java.util package used for obtaining the input of the primitive types like int, double, etc. and strings.

#### nextInt()

Scans the next token of the input as an int.

### PROGRAM:

```
import java.util.*;
public class Rearranged {
    static void rearrange(int arr[], int n) {
        int i, j=0;
        for (i=0; i<n; i++) {
            if(arr[i]<0 && i != j) {
                int temp=arr[i];
                arr[i]=arr[j];
                arr[j]=temp;
            }
        }
    }
}
```

```

        j++;
    }
}
public static void main(String []args) {
    Scanner sr=new Scanner(System.in);
    int n, i;
    System.out.print("Enter the size: ");
    n=sr.nextInt();
    int arr[]=new int[n];
    System.out.println("Enter elements: ");
    for (i=0; i<n; i++)
        arr[i]=sr.nextInt();
    System.out.println("Original array: ");
    for (i=0; i<n; i++)
        System.out.print(arr[i]+" ");
    System.out.println();
    rearrange(arr, n);
    System.out.print("Rearranged array: ");
    for (i=0; i<n; i++)
        System.out.print(arr[i]+" ");
    }
}

```

**Output:**

```

Enter the size: 5
Enter elements:
4 -4 7 -10 11
Rearranged array: -4 -10 7 4 11

```

## Problem Statement 7 and solution

### •Problem Statement

Program to find the saddle point coordinates in a given matrix. A saddle point is an element of the matrix, which is the minimum element in its row and the maximum in its column.

For example, consider the matrix given below

1 2 3

4 5 6

7 8 9

Here, 7 is the saddle point because it is the minimum element in its row and maximum element in its column.

### •Theory, important classes and methods

#### Arrays in Java

An array is a group of like-typed variables that are referred to by a common name.

A two — dimensional array can be seen as a table with ‘x’ rows and ‘y’ columns where the row number ranges from 0 to (x-1) and column number ranges from 0 to (y-1). A two — dimensional array ‘x’ with 3 rows and 3 columns.

#### PROGRAM:

```
import java.util.Scanner;
public class SaddlePoint {
    public static void main(String[] args) {
        int[][] matrix={{10, 11, 12}, {13, 14, 15}, {16, 17, 18}};
        int m=3, min=0, max=0, k;
        int[][] pos=new int[2][2];
        int i, j;
        for (i=0; i<m; i++) {
            for (j=0; j<m; j++)
                System.out.print(matrix[i][j]+" ");
            System.out.println();
        }
        for (i=0; i<m; i++) {
            min=matrix[i][0];
            for (j=0; j<m; j++) {
                if (min >= matrix[i][j]) {
                    min=matrix[i][j];
                    pos[0][0]=i;
                    pos[0][1]=j;
                }
            }
        }
    }
}
```

```

    }
}
j=pos[0][1];
max=matrix[0][j];
for (k=0; k<m; k++) {
    if (max <= matrix[k][j]) {
        max=matrix[k][j];
        pos[1][0]=k;
        pos[1][1]=j;
    }
}
if (min==max) {
    if (pos[0][0]==pos[1][0] && pos[0][1]==pos[1][1])
        System.out.print("Saddle point('"+pos[0][0]+",""+pos[0][1]+"):'"+max);
}
}
}
}

```

**Output:**

```

10 11 12
13 14 15
16 17 18
Saddle point(2,0):

```

## Problem Statement 8 and solution

### •Problem Statement

Program to find all the patterns of 0(1+)0 in the given string. Given a string containing 0's and 1's, find the total number of 0(1+)0 patterns in the string and output it.

0(1+)0 - There should be at least one '1' between the two 0's. For example, consider the following string.

**Input:** 01101111010

**Output:** 3

### •Theory, important classes and methods

The **String class** represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.

Strings are constant; their values cannot be changed after they are created.

### toCharArray() Method Of String class

Converts this string to a new character array. A newly allocated character array whose length is the length of this string and whose contents are initialized to contain the character sequence represented by this string.

### PROGRAM:

```
import java.util.*;
public class Pattern {
    static int find_pattern(char str[]) {
        char last=str[0];
        int i=1, counter=0;
        while (i<str.length) {
            if (str[i]=='1' && last=='0') {
                while (str[i]=='1')
                    i++;
                if (str[i]=='0')
                    counter++;
            }
            last=str[i];
            i++;
        }
        return counter;
    }
    public static void main (String args[]) {
```

```
String string;  
Scanner sc=new Scanner(System.in);  
System.out.print("Enter the string: ");  
string=sc.next();  
char[] str=string.toCharArray();  
System.out.print("Number of patterns = "+find_pattern(str));  
}  
}
```

**Output:**

Enter the string: 01101111010

Number of patterns = 3

## Problem Statement 9 and solution

### •Problem Statement

Write a java program to delete vowels from given string using StringBuffer class.

### •Theory, important classes and methods

**Class StringBuffer** A thread-safe, mutable sequence of characters. A string buffer is like a String, but can be modified. At any point in time it contains some particular sequence of characters, but the length and content of the sequence can be changed through certain method calls.

**toUpperCase(char ch) Method Of Character Class** Converts the character argument to uppercase

**deleteCharAt(int index)** Removes the char at the specified position in this sequence. This sequence is shortened by one char.

### PROGRAM:

```
public class StringDemo {
    public static void main(String[] args) {
        StringBuffer sb=new StringBuffer("Dehradun");
        int ln=sb.length();
        char ch, c;
        for (int i=0; i<ln; i++) {
            c=0;
            ch=Character.toUpperCase(sb.charAt(i));
            switch(ch) {
                case 'A': case 'E': case 'I': case 'O': case 'U':
                    c++;
                    break;
            }
            if (c>0) {
                sb.deleteCharAt(i);
                ln--;
            }
        }
        System.out.println(sb);
    }
}
```

### Output:

Dhrdn

## Problem Statement 10 and solution

### •Problem Statement

Write a Java program to create a class named '**Bank**' with the following data members: Name of depositor & Address of depositor Account Number Balance in account.

Class '**Bank**' has a method for each of the following:

- 1- Generate a unique account number for each depositor  
For first depositor, account number will be 1001, for second depositor it will be 1002 and so on
- 2- Display information and balance of depositor
- 3- Deposit more amount in balance of any depositor
- 4 - Withdraw some amount from balance deposited
- 5 - Change address of depositor

After creating the class, do the following operations

- 1- Enter the information (name, address, account number, balance) of the depositors. Number of depositors is to be entered by user.
- 2- Print the information of any depositor.
- 3- Add some amount to the account of any depositor and then display final information of that depositor
- 4- Remove some amount from the account of any depositor and then display final information of that depositor
- 5- Change the address of any depositor and then display the final information of that depositor
- 6- Randomly repeat these processes for some other bank accounts.

### •Theory, important classes and methods

#### Classes and Objects in Java

Classes and Objects are basic concepts of Object Oriented Programming which revolve around the real life entities.

#### Class

A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. In general, class declarations can include these components, in order:

#### Object

It is a basic unit of Object Oriented Programming and represents the real life entities. A typical Java program creates many objects, which interact by invoking methods. An object consists of:

1. **State** : It is represented by attributes of an object. It also reflects the properties of an object.
2. **Behavior** : It is represented by methods of an object. It also reflects the response of an object with other objects.
3. **Identity** : It gives a unique name to an object and enables one object to interact with other objects.



**PROGRAM:**

```
import java.util.*;
public class Bank {
    private int accountno;
    private String personname, address;
    private double balance;
    public void InputInfo(String pname, String add, double bal, int acno) {
        personname=pname;
        address=add;
        balance=bal;
        accountno=acno;
    }
    public void showInfo() {
        System.out.println("Person Name = "+personname);
        System.out.println("Address = "+address);
        System.out.println("Account Number = "+accountno);
        System.out.println("Balance = "+balance);
    }
    public void deposit(double amt) {
        balance=balance+amt;
    }
    public void withdraw(double amt) {
        if(balance>amt)
            balance=balance-amt;
        else
            System.out.println("Insufficant Balance");
    }
    public int getAccountNo() {
        return accountno;
    }
    public double getBalance() {
        return balance;
    }
    public String getAddress() {
        return address;
    }
    public void changeAddress(String add) {
        address=add;
    }

    public static void main(String[] args) {
        int size, i, f, acno=1000;
        char ans;
        String pname, add;
        double amt;
```

```

Scanner sr=new Scanner(System.in);
System.out.print("Enter no. of new depositor: ");
size=sr.nextInt();
System.out.print("Press Any Key For Entering Depositor Information: ");
ans=sr.next().charAt(0);
Bank ob[]=new Bank[size];
    for (i=0; i<size; i++) {
        ob[i]=new Bank();
        System.out.print("Enter Name: ");
        pname=sr.next();
        System.out.print("Enter Address: ");
        add=sr.next();
        System.out.print("Enter Amount: ");
        amt=sr.nextDouble();
        acno=acno+1;
        ob[i].InputInfo(pname, add, amt, acno);
    }
for(;;) {
    System.out.println("1 - Display information and balance of Depositor");
    System.out.println("2 - Deposit more amount in balance of any depositor");
    System.out.println("3 - Withdraw some amount from balance deposited");
    System.out.println("4 - Change address of depositor");
    System.out.print("Enter your choice: ");
    int ch=sr.nextInt();
    switch(ch) {
        case 1:
            f=0;
            System.out.print("Enter Account No.: ");
            acno=sr.nextInt();
            for (i=0; i<size; i++) {
                if(acno==ob[i].getAccountNo()) {
                    ob[i].showInfo();
                    f=1;
                    break;
                }
            }
            if(f==0)
                System.out.println("Invalid AccountNo ");
            break;
        case 2:
            f=0;
            System.out.print("Enter Account No.: ");
            acno=sr.nextInt();
            for (i=0; i<size; i++) {
                if(acno==ob[i].getAccountNo()) {

```

```

        System.out.println("Enter amount: ");
        amt=sr.nextDouble();
        ob[i].deposit(amt);
        System.out.println("Updated Amount =
"+ob[i].getBalance());
        f=1;
        break;
    }
}
if(f==0)
    System.out.println("Invalid AccountNo ");
    break;
case 3:
    f=0;
    System.out.println("Enter Account No.: ");
    acno=sr.nextInt();
    for (i=0; i<size; i++) {
        if(acno==ob[i].getAccountNo()) {
            System.out.println("Enter amount: ");
            amt=sr.nextDouble();
            ob[i].withdraw(amt);
            System.out.println("Updated Amount =
"+ob[i].getBalance());
            f=1;
            break;
        }
    }
    if(f==0)
        System.out.println("Invalid Account Number");
        break;
case 4:
    f=0;
    System.out.println("Enter Account No.: ");
    acno=sr.nextInt();
    for (i=0; i<size; i++) {
        if(acno==ob[i].getAccountNo()) {
            System.out.println("Enter New Address: ");
            add=sr.nextLine();
            ob[i].changeAddress(add);
            System.out.println("Updated Address:
"+ob[i].getAddress());
            f=1;
            break;
        }
    }
}

```

```

        if(f==0)
            System.out.println("Invalid Account Number");
        break;
    }
    System.out.print("Do you want to more transactions (Y/N): ");
    ans=sr.next().charAt(0);
    if(ans=='N' || ans=='n')
        break;
    }
    }
}

```

### **Output:**

```

Enter no. of new depositor: 2
Press Any Key For Entering Depositor Information: y
Enter Name: Salman
Enter Address: Mumbai
Enter Amount: 8000
Enter Name: Salim
Enter Address: Mumbai
Enter Amount: 5000
1 - Display information and balance of Depositor
2 - Deposit more amount in balance of any depositor
3 - Withdraw some amount from balance deposited
4 - Change address of depositor
Enter your choice: 1
Enter Account No.: 2001
Invalid Account Number
Do you want to more transactions (Y/N): y
1 - Display information and balance of Depositor
2 - Deposit more amount in balance of any depositor
3 - Withdraw some amount from balance deposited
4 - Change address of depositor
Enter your choice: 1
Enter Account No.: 1001
Person Name = Salman
Address = Mumbai
Account Number = 1001
Balance = 8000.0

```

## Problem Statement 11 and solution

### •Problem Statement

Define a class **WordExample** having the following description:Data members/instance variables:  
private String strdata : to store a sentence.

### Parameterized Constructor

**WordExample(String)** : Accept a sentence which may be terminated by either '.', '? 'or'!' only. The words may be separated by more than one blank space and are in UPPERCASE.

### Member Methods

**void countWord():** Find the number of words beginning and ending with a vowel.

**void placeWord():** Place the words which begin and end with a vowel at the beginning, followed by the remaining words as they occur in the sentence.

### •Theory, important classes and methods

#### Classes and Objects in Java

Classes and Objects are basic concepts of Object Oriented Programming which revolve around the real life entities.

#### Class

A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. In general, class declarations can include these components, in order:

#### Object

It is a basic unit of OOPS and represents the real life entities. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of:

1. **State** : It is represented by attributes of an object. It also reflects the properties of an object.
2. **Behavior** : It is represented by methods of an object. It also reflects the response of an object with other objects.
3. **Identity** : It gives a unique name to an object and enables one object to interact with other objects.

**charAt(int index)** Returns the char value at the specified index. An index ranges from 0 to length()  
The first char value of the sequence is at index 0, the next at index 1, and so on, as for array indexing.

#### length()

Returns the length of this string. The length is equal to the number of Unicode code units in the string.

**PROGRAM:**

```
import java.util.*;
class WordExample {
    private String strdata;
    public WordExample (String str) {
        strdata=str;
    }
    public void wordCount() {
        int i, ln, wln, c1, c2, c=0;
        String tmp;
        char ch1, ch2;
        ln=strdata.length();
        for (i=0; i<ln; i++) {
            tmp="";
            c1=0;
            c2=0;
            for (; strdata.charAt(i)!=' '; i++) {
                if(i==(ln-1))
                    break;
                tmp=tmp+strdata.charAt(i);
            }
            if(!tmp.isEmpty()) {
                wln=tmp.length();
                ch1=tmp.charAt(0);
                ch2=tmp.charAt(wln-1);
                switch(ch1) {
                    case 'A':
                    case 'E':
                    case 'I':
                    case 'O':
                    case 'U':
                        c1++;
                }
                switch(ch2) {
                    case 'A':
                    case 'E':
                    case 'I':
                    case 'O':
                    case 'U':
                        c2++;
                }
                if(c1==1 && c2==1)
                    c++;
            }
        }
    }
}
```

```

    }
    System.out.println("Number Of Words Are= "+c);
}
public void placeWord() {
    int i, ln, wln, c1, c2, c=0;
    String tmp, fstr="", lstr="";
    char ch1, ch2;
    ln=strdata.length();
    for (i=0; i<ln; i++) {
        tmp="";
        c1=0;
        c2=0;
        for (; strdata.charAt(i)!=' '; i++) {
            if(i==(ln-1))
                break;
            tmp=tmp+strdata.charAt(i);
        }
        if(!tmp.isEmpty()) {
            wln=tmp.length();
            ch1=tmp.charAt(0);
            ch2=tmp.charAt(wln-1);
            switch(ch1) {
                case 'A':
                case 'E':
                case 'I':
                case 'O':
                case 'U':
                    c1++;
            }
            switch(ch2) {
                case 'A':
                case 'E':
                case 'I':
                case 'O':
                case 'U':
                    c2++;
            }
            if(c1==1&& c2==1)
                fstr=fstr+tmp+" ";
            else
                lstr=lstr+tmp+" ";
        }
    }
    tmp=fstr+lstr;
}

```

```

        tmp=tmp.trim();
        tmp=tmp+".";
        System.out.println(tmp);
    }
    public static void main(String args[]) {
        Scanner sr=new Scanner(System.in);
        System.out.print("Sentence: ");
        String s=sr.nextLine();
        s=s.trim();
        s=s.toUpperCase();
        int len=s.length();
        char last=s.charAt(len - 1);
        if(last != '.' && last != '?' && last != '!') {
            System.out.println("INVALID INPUT");
            System.exit(0);
        }
        WordExample ob=new WordExample(s);
        ob.wordCount();
        ob.placeWord();
    }
}

```

**Output:**

Sentence: RAJ AND SHYAM ARE GOING TO PICNIC TOGETHER.  
 Number Of Words Are= 1  
 ARE RAJ AND SHYAM GOING TO PICNIC TOGETHER.



## Problem Statement 12 and solution

### •Problem Statement

Write a Java program to create a class called **ArrayDemo** and overload arrayFunc() function.

**void arrayFunc(int [], int)** □ To find all pairs of elements in an Array whose sum is equal to a given number :

Array numbers= [4, 6, 5, -10, 8, 5, 20], target=10

#### **Output :**

Pairs of elements whose sum is 10 are:

$4 + 6 = 10$

$5 + 5 = 10$

$-10 + 20 = 10$

**void arrayFunc(int A[], int p, int B[], int q)** □ Given two sorted arrays A and B of size p and q, Overload method arrayFunc() to merge elements of A with B by maintaining the sorted order i.e. fill A with first p smallest elements and fill B with remaining elements.

#### **Example:**

##### **Input :**

int[] A = { 1, 5, 6, 7, 8, 10 }

int[] B = { 2, 4, 9 }

##### **Output:**

Sorted Arrays:

A: [1, 2, 4, 5, 6, 7]

B: [8, 9, 10]

(Use Compile time Polymorphism Method Overloading)

## •Theory, important classes and methods

### Polymorphism

Polymorphism gives us the ultimate flexibility in extensibility. Polymorphism is a term that describes a situation where one name may refer to different methods.

### Method Overloading In Java

Overloading allows different methods to have same name, but different signatures where signature can differ by number of input parameters or type of input parameters or both. Overloading is related to compile time (or static) polymorphism.

#### PROGRAM:

```
import java.util.*;
class ArrayDemo {
    public void arrayFunc(int arr[],int sum) {
        System.out.println("Pairs of elements whose sum is "+sum+" = ");
        for (int i=0; i<arr.length; i++) {
            for (int j=i+1; j<arr.length; j++) {
                if(arr[i]+arr[j]==sum)
                    System.out.println(arr[i]+" + "+arr[j]+" = "+sum);
            }
        }
    }
    public void arrayFunc(int[] A, int p, int[] B, int q) {
        for (int i=0; i<p; i++) {
            if (A[i]>B[0]) {
                int temp=A[i];
                A[i]=B[0];
                B[0]=temp;
                int first_arr=B[0];
                int k;
                for (k=1; k<q && B[k]<first_arr; k++)
                    B[k-1]=B[k];
                B[k-1]=first_arr;
            }
        }
    }
}
public class MethodOver {
    public static void main(String args[]) {
        ArrayDemo ob=new ArrayDemo();
        int[] arr={1, 5, 7, -1, 5};
        int sum=6;
        ob.arrayFunc(arr, sum);
    }
}
```

```

int[] A={1, 5, 6, 7, 8, 10};
int[] B={2, 4, 9};
int p=A.length;
int q=B.length;
System.out.println("\nOriginal Arrays:");
System.out.println("A: "+Arrays.toString(A));
System.out.println("B: "+Arrays.toString(B));
ob.arrayFunc(A, p, B, q);
System.out.println("\nSorted Arrays:");
System.out.println("A: "+Arrays.toString(A));
System.out.println("B: "+Arrays.toString(B));
    }
}

```

### **Output:**

Pairs of elements whose sum is 6 =

1 + 5 = 6

1 + 5 = 6

7 + -1 = 6

Original Arrays:

A: [1, 5, 6, 7, 8, 10]

B: [2, 4, 9]

Sorted Arrays:

A: [1, 2, 4, 5, 6, 7]

B: [8, 9, 10]

## Problem Statement 13 and solution

### •Problem Statement

Write a java program to calculate the area of a rectangle, a square and a circle. Create an abstract class 'Shape' with three abstract methods namely **rectangleArea()** taking two parameters, **squareArea()** and **circleArea()** taking one parameter each.

Now create another class 'Area' containing all the three methods **rectangleArea()**, **squareArea()** and **circleArea()** for printing the area of rectangle, square and circle respectively. Create an object of class Area and call all the three methods. (Use **Runtime Polymorphism**)

### •Theory, important classes and methods

#### Polymorphism

Polymorphism gives us the ultimate flexibility in extensibility. Polymorphism is a term that describes a situation where one name may refer to different methods.

#### Method Overriding In Java

In **runtime polymorphism**, the method to be invoked is determined at the run time. The example of run time polymorphism is method overriding. When a subclass contains a method with the same name and signature as in the super class then it is called as method overriding.

#### Abstract class and Abstract Method

**Abstract class**  
A class that is declared using "abstract" keyword is known as abstract class. It can have abstract methods (methods without body) as well as concrete methods (regular methods with body).

Abstract classes cannot have objects but they can be used to create object references because java's runtime polymorphism is implemented through the use of super class references. Thus, an object can be used to create a reference to an abstract class that can point to a subclass object.

#### Abstract Method

Certain methods in the super class do not contain any logic and need to be overridden by the subclass. In such situations, the method in the super class should be declared by using the keyword **abstract**.

The subclass provides the implementation details of such abstract methods. The super class only provides the name and signatures of the method. Thus, it is the responsibility of the subclass to override it.

**PROGRAM:**

```
abstract class Shape {
    abstract double rectangleArea(double len,double br);
    abstract double squareArea(double side);
    abstract double circleArea(double radius);
}
class Area extends Shape {
    double rectangleArea(double length,double breadth) {
        return length * breadth;
    }
    double squareArea(double side) {
        return side * side;
    }
    double circleArea(double radius) {
        return (22.0/7.0) * radius * radius;
    }
}
public class CalArea {
    public static void main(String arg[]) {
        Shape sp=new Area();
        System.out.println("Rectangle Area: "+sp.rectangleArea(10, 5));
        System.out.println("Square Area: "+sp.squareArea(5));
        System.out.println("Circle Area: "+sp.circleArea(7));
    }
}
```

**Output:**

```
Rectangle Area: 50.0
Square Area: 25.0
Circle Area: 154.0
```

## Problem Statement 14 and solution

### •Problem Statement

Write a java program to implement abstract class and abstract method with following details:

#### Create a abstract Base Class: Temperature

##### Data members:

double temp;

##### Method members:

void setTempData(double) abstract void changeTemp()

#### Sub Class Fahrenheit (subclass of Temperature)

##### Data members:

double ctemp;

##### Method member:

Override abstract method changeTemp() to convert Fahrenheit temperature into degree Celsius by using formula  $C = 5/9 * (F - 32)$  and display converted temperature

#### Sub Class Celsius (subclass of Temperature)

##### Data member:

double ftemp;

##### Method member:

Override abstract method changeTemp() to convert degree Celsius into Fahrenheit temperature by using formula  $F = 9/5 * c + 32$  and display converted temperature

### •Theory, important classes and methods

#### Abstract class and Abstract Method

##### Abstract class

A class that is declared using “abstract” keyword is known as abstract class. It can have abstract methods (methods without body) as well as concrete methods (regular methods with body).

Abstract classes cannot have objects but they can be used to create object references because java's runtime polymorphism is implemented through the use of super class references. Thus, an object can be used to create a reference to an abstract class that can point to a subclass object.

##### Abstract Method

Certain methods in the super class do not contain any logic and need to be overridden by the subclass. In such situations, the method in the super class should be declared by using the keyword abstract.

The subclass provides the implementation details of such abstract methods. The super class only provides the name and signatures of the method. Thus, it is the responsibility of the subclass to override it.

**PROGRAM:**

```
abstract class Temperature {
    protected double temp;
    void setTempData(double tmp) {
        temp=tmp;
    }
    abstract void changeTemp();
}
class Fahrenheit extends Temperature {
    double ctemp;
    void changeTemp() {
        ctemp=5.0/9.0*(temp-32.0);
        System.out.println("Fahrenheit into Degree Celsius = "+ctemp);
    }
}
class Celsius extends Temperature {
    double ftemp;
    void changeTemp() {
        ftemp=(9.0/5.0)*temp+32.0 ;
        System.out.println("Degree Celsius into Fahrenheit = "+ftemp);
    }
}
public class ConvertTemp {
    public static void main(String [] args) {
        Temperature ref;
        Fahrenheit fr=new Fahrenheit();
        fr.setTempData(108);
        Celsius cr=new Celsius();
        cr.setTempData(50);
        ref=fr;
        ref.changeTemp();
        ref=cr;
        ref.changeTemp();
    }
}
```

**Output:**

```
Fahrenheit into Degree Celsius = 42.22222222222222
Degree Celsius into Fahrenheit = 122.0
```

## Problem Statement 15 and solution

### •Problem Statement

Write a java program to create an interface that consists of a method to display **volume ()** as an abstract method and redefine this method in the derived classes to suit their requirements.

Create classes called **Cone**, **Hemisphere** and **Cylinder** that implements the interface. Using these three classes, design a program that will accept dimensions of a cone, cylinder and hemisphere interactively and display the volumes.

Volume of cone =  $(1/3)\pi r^2 h$  Volume of hemisphere =  $(2/3)\pi r^3$  Volume of cylinder =  $\pi r^2 h$

### •Theory, important classes and methods

#### Java – Interfaces

Like a class, an interface can have methods and variables, but the methods declared in interface are by default abstract (only method signature, no body).

#### PROGRAM:

```
interface CalVolume {
    static double pi=3.14159;
    void volume();
}
class Cone implements CalVolume {
    double r, h;
    Cone(double r, double h) {
        this.r=r;
        this.h=h;
    }
    public void volume() {
        double vol;
        vol=(double)1/3*pi*h*r*r;
        System.out.println("Volume of Cone = "+vol);
    }
}
class Hemisphere implements CalVolume {
    double r;
    Hemisphere(double r) {
        this.r=r;
    }
    public void volume() {
        double vol;
```



```

        vol=(double)(2*pi*(double)Math.pow(r,3))/(double)(3);
        System.out.println("Volume of Hemisphere = "+vol);
    }
}
class Cylinder implements CalVolume {
    double r, h;
    Cylinder(double r, double h) {
        this.r=r;
        this.h=h;
    }
    public void volume() {
        double vol;
        vol=pi*r*r*h;
        System.out.println( "Volume of Cylinder = "+vol );
    }
}
public class VolumeCal {
    public static void main(String args[]) {
        Cone c1=new Cone(10, 15.0);
        Hemisphere h1=new Hemisphere(20.0);
        Cylinder cy=new Cylinder(7.0,14.0);
        CalVolume ref;
        ref=c1;
        ref.volume();
        ref=h1;
        ref.volume();
        ref=cy;
        ref.volume();
    }
}

```

**Output:**

Volume of Cone = 1570.795  
 Volume of Hemisphere = 16755.146666666664  
 Volume of Cylinder = 2155.1307399999996

## Problem Statement 16 and solution

### •Problem Statement

Write a java program to accept and print the employee details during runtime. The details will include employee id, name, dept\_ Id. The program should raise an exception if the user inputs incomplete or incorrect data. The entered value should meet the following conditions:

- (i) First Letter of employee's name should be in capital letter.
- (ii) Employee id should be between 2001 and 5001
- (iii) Department id should be an integer between 1 and 5.

If the above conditions are not met then the application should raise specific exception else should complete normal execution.

### •Theory, important classes and methods

#### Exceptions

An exception (or exceptional event) is a problem that arises during the execution of a program. When an Exception occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.

#### Handling Exceptions in JavaCatching Exceptions

A method catches an exception using a combination of the try and catch keywords. A try/catch block is placed around the code that might generate an exception. Code within a try/catch block is referred to as protected code, and the syntax for using try/catch looks like the following –Syntax

```
try {  
    // Protected code  
} catch (ExceptionName e1) {  
    // Catch block  
}
```

#### **PROGRAM:**

```
import java.util.Scanner;  
public class DemoException extends Exception {  
    public DemoException(String string) {  
        super(string);  
    }  
    public static void main(String[] args) {  
        String name=null;  
        int empid, depid, i=0;  
        Scanner sc=new Scanner(System.in);  
        try {
```

```

        System.out.print("Enter Name: ");
        name=sc.nextLine();
        if(!(Character.isUpperCase(name.charAt(0))))
            throw new DemoException("First Letter should have only Capital");
        System.out.print("Enter EmpId: ");
        empid=sc.nextInt();
        if((empid>=2001 && empid<=5001))
            throw new DemoException("Employee ID should be between 2001 and 5001");
        System.out.print("Enter DepId:");
        depid=sc.nextInt();
        if(!(depid>=1 && depid<=5))
            throw new DemoException("Department ID should be between 0 and 5");
        System.out.println("All data is valid");
        System.out.println("Name: "+name);
        System.out.println("Employee ID: "+empid);
        System.out.println("Department ID:"+depid);
    }
    catch(Exception e) {
        System.out.println(e.getMessage());
    }
    finally {
        System.out.println("Thank you!!");
    }
}
}

```

**Output:**

```

Enter Name: Ram
Enter EmpId: 2023
Enter DepId:4
All data is valid
Name: Ram
Employee ID: 2023
Department ID:4
Thank you!!

```

**Output:**

```

Enter Name: Ram
Enter EmpId: 5078
Employee ID should be between 2001 and 5001
Thank you!!

```

**Output:**

```

Enter Name: Ram
Enter EmpId: 4001
Enter DepId:7
Department ID should be between 0 and 5
Thank you!!

```

## Problem Statement 17 and solution

### •Problem Statement

Create a class **MyCalculator** which consists of a single method **power(int, int)**. This method takes two integers, **n** and **p**, as parameters and finds **n<sup>p</sup>**.  
If either **n** or **p** is negative, then the method must throw an exception which says "n and p should be non-negative".

### Input Format

Each line of the input contains two integers, **n** and **p**.

Output Format  
Each line of the output contains the result, if neither of **n** and **p** is negative. Otherwise the output contains "n and p should be non-negative".

### Sample Input

```
3 5
2 4
0 0
-1 -2
-1 3
```

### Sample Output

```
243
16
```

```
java.lang.Exception: n and p should not be zero.
java.lang.Exception: n or p should not be negative.
java.lang.Exception: n or p should not be negative.
```

### Explanation

In the first two cases, both **n** and **p** are positive. So, the power function returns the answer correctly.  
In the third case, both **n** and **p** are zero. So, the exception, "**n** and **p** should not be zero." is printed.  
In the last two cases, at least one out of **n** and **p** is negative. So, the exception, "**n** or **p** should not be negative." is printed for these two cases.

### •Theory, important classes and methods

#### Exceptions

An exception (or exceptional event) is a problem that arises during the execution of a program. When an Exception occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.

#### Handling Exceptions in Java

**Catching Exceptions**  
A method catches an exception using a combination of the try and catch keywords. A try/catch block is placed around the code that might generate an exception. Code within a try/catch block is referred to as protected code, and the syntax for using try/catch looks like the following –Syntax

```

try {
    // Protected code
} catch (ExceptionName e1) {
    // Catch block
}

```

**PROGRAM:**

```

import java.util.Scanner;
class MyCalculator {
    int power(int n, int p) throws Exception {
        if(n==0 && p==0)
            throw new Exception("n and p should not be 0.");
        else if (n<0 || p<0)
            throw new Exception("n or p should not be negative.");
        else return (int)Math.pow(n, p);
    }
}
class Except{
    public static void main(String []args) {
        Scanner in=new Scanner(System.in);
        System.out.print("Enter Value of n and p: ");
        int n=in.nextInt();
        int p=in.nextInt();
        MyCalculator M=new MyCalculator();
        try {
            System.out.println(M.power(n,p));
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}

```

**Output:**

Enter Value of n and p: 5 2  
25

**Output:**

Enter Value of n and p: 2 -7  
java.lang.Exception: n or p should not be negative.

**Output:**

Enter Value of n and p: 0 0  
java.lang.Exception: n and p should not be 0.

## Problem Statement 18 and solution

### •Problem Statement

Write a java file handling program to count and display the number of palindromes present in a text file "myfile.txt".

Example: If the file "myfile.txt" contains the following lines, My name is NITIN  
Hello aaa and bbb word  
How are You ARORA is my friend

Output will be => 4

### •Theory, important classes and methods

#### **public class** BufferedReader

Reads text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines.

#### **public class** FileReader

Convenience class for reading character files. The constructors of this class assume that the default character encoding and the default byte-buffer size are appropriate.

#### **public String** readLine()

Reads a line of text. A line is considered to be terminated by any one of a line feed ('\n'), a carriage return ('\r'), a carriage return followed immediately by a line feed, or by reaching the end-of-file (EOF).

### **PROGRAM:**

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
public class MyFileProg {
    public static void main(String a[]){
        BufferedReader br=null;
        String revword, strLine="";
        int c=0;
        try {
            br=new BufferedReader(new FileReader("MyFile.txt"));
            while((strLine=br.readLine()) != null) {
                System.out.println(strLine);
                String st[]=strLine.split(" ");
```

```

        for(String tmp:st) {
            StringBuffer data=new StringBuffer(tmp);
            data.reverse();
            revword=data.toString();
            if(tmp.equalsIgnoreCase(revword))
                c++;
        }
    }
    System.out.println("Total number of palindrome = "+c);
    br.close();
}
catch (FileNotFoundException e) {
    System.err.println("File not found");
}
catch (IOException e) {
    System.err.println("Unable to read the file.");
}
}
}

```

**Output:**

```

Hello User!
My name is NAMAN
My roll number is 10001
Total number of palindrome = 2

```

## Problem Statement 19 and solution

### •Problem Statement

Write a program MultiThreads that creates two threads-one read with the name CSthread and the other thread named ITthread.

Each thread should display its respective name and execute after a gap of 500 milliseconds.

Each thread should also display a number indicating the number of times it got a chance to execute.

### •Theory, important classes and methods

**public class Thread**

A thread is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently.

Java provides a thread class that has various method calls in order to manage the behavior of threads.

### **PROGRAM:**

```
class CsThread extends Thread {
    private Thread t;
    private String threadName;
    CsThread( String name) {
        threadName=name;
        System.out.println("Creating "+threadName);
    }
    public void run() {
        System.out.println("Running "+threadName);
        try {
            for (int i=1; i<=4; i++) {
                System.out.println("Thread: "+threadName+", "+i);
                Thread.sleep(500);
            }
        }
        catch (InterruptedException e) {
            System.out.println("Thread "+threadName+" interrupted.");
        }
        System.out.println("Thread "+threadName+" exiting.");
    }
    public void start() {
        System.out.println("Starting "+threadName);
        if (t==null) {
```



```

        t=new Thread(this, threadName);
        t.start();
    }
}
}
class ItThread extends Thread {
    private Thread t;
    private String threadName;
    ItThread( String name) {
        threadName=name;
        System.out.println("Creating "+threadName);
    }
    public void run() {
        System.out.println("Running "+threadName );
        try {
            for (int i=1; i<=4; i++) {
                System.out.println("Thread: "+threadName+", "+i);
                Thread.sleep(500);
            }
        }
        catch (InterruptedException e) {
            System.out.println("Thread "+threadName+" interrupted.");
        }
        System.out.println("Thread "+threadName+" exiting.");
    }
    public void start() {
        System.out.println("Starting "+threadName);
        if (t==null) {
            t=new Thread(this, threadName);
            t.start();
        }
    }
}
public class MyThread {
    public static void main(String args[]) {
        CsThread T1=new CsThread("CSthread");
        T1.start();
        ItThread T2=new ItThread( "ITthread");
        T2.start();
    }
}

```

**Output:**

Creating CSthread  
Starting CSthread  
Running CSthread  
Creating ITthread  
Starting ITthread  
Running ITthread  
Thread: CSthread, 1  
Thread: ITthread, 1  
Thread: ITthread, 2  
Thread: CSthread, 2  
Thread: CSthread, 3  
Thread: ITthread, 3  
Thread: CSthread, 4  
Thread: ITthread, 4  
Thread ITthread exiting.  
Thread CSthread exiting.

## Problem Statement 20 and solution

- **Problem Statement**

Write a Java program for to solve producer consumer problem in which a producer produce a value and consumer consume the value before producer generate the next value.

- **Theory, important classes and methods**

### **public interface Runnable**

The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread. The class must define a method of no arguments called run

### **public final void wait()**

Causes the current thread to wait until it is awakened, typically by being notified or interrupted.

### **public final void notify()**

Wakes up a single thread that is waiting on this object's monitor. If any threads are waiting on this object, one of them is chosen to be awakened. The choice is arbitrary and occurs at the discretion of the implementation. A thread waits on an object's monitor by calling one of the wait methods.

### **PROGRAM:**

```
class Queue {
    int item=0;
    boolean busy=false;
    synchronized int get() {
        if(!busy)
            try {
                wait();
            }
        catch(InterruptedException e) {
            System.out.println("Get:InterruptedException");
        }
        System.out.println("Get:"+item);
        busy=false;
        notify();
        return item;
    }
    synchronized void put(int item) {
        if(busy)
            try {
                wait();
            }
    }
```

```

        catch(InterruptedException e) {
            System.out.println("Put:InterruptedException");
        }
        this.item=item;
        busy=true;
        System.out.println("Put:"+item);
        notify();
    }
}
class Producer implements Runnable {
    int i;
    Queue q;
    Producer(Queue q) {
        this.q=q;
        new Thread(this, "Producer").start();
    }
    public void run() {
        try {
            while(i<=10)
                q.put(i++);
            Thread.sleep(500);
        }
        catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
class Consumer implements Runnable {
    Queue q;
    Consumer(Queue q) {
        this.q=q;
        new Thread(this, "Consumer").start();
    }
    public void run() {
        int i=0;
        while(true) {
            try {
                while(i<=10)
                    q.get();
                Thread.sleep(500);
            }
            catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```
    }  
}  
public class ThreadDemo {  
    public static void main(String[] args) {  
        Queue q=new Queue();  
        new Producer(q);  
        new Consumer(q);  
    }  
}
```

**Output:**

Put:0  
Get:0  
Put:1  
Get:1  
Put:2  
Get:2  
Put:3  
Get:3  
Put:4  
Get:4  
Put:5  
Get:5  
Put:6  
Get:6  
Put:7  
Get:7  
Put:8  
Get:8  
Put:9  
Get:9  
Put:10  
Get:10

## Problem Statement 21 and solution

### •Problem Statement

Write a method `removeEvenLength` that takes an `ArrayList` of `Strings` as a parameter and that removes all of the strings of even length from the list. (**Use `ArrayList`**)

### •Theory, important classes and methods

#### **ArrayList in Java**

`ArrayList` is a part of the **collection framework** and is present in `java.util` package. It provides us with dynamic arrays in Java.

**Adding Elements:** In order to add an element to an `ArrayList`, we can use the `add()` method

**add(Object):** This method is used to add an element at the end of the `ArrayList`.

#### **PROGRAM:**

```
import java.util.ArrayList;
class ArrayListData {
    public static void removeEvenLength(ArrayList<String> array) {
        for (int i=0; i<array.size(); i++) {
            String word=array.get(i);
            if (word.length()%2==0)
                array.remove(i--);
        }
    }
}

public class RemoveList {
    public static void main(String[] args) {
        ArrayList<String> arrlist=new ArrayList<String>(5);
        arrlist.add("C++");
        arrlist.add("DataScience");
        arrlist.add("Python");
        arrlist.add("Cyber");
        System.out.println("-----Array List Item-----");
        for (String str: arrlist)
            System.out.println(str);
        ArrayListData.removeEvenLength(arrlist);
        System.out.println("-----Resultant Array List Item-----");
        for (String str: arrlist)
            System.out.println(str);
    }
}
```

**Output:**

-----Array List Item-----

C++

DataScience

Python

Cyber

-----Resultant Array List Item-----

C++

DataScience

Cyber

## Problem Statement 22 and solution

- **Problem Statement**

Write a method `swapPairs` that switches the order of values in an `ArrayList` of `Strings` in a pairwise fashion. Your method should switch the order of the first two values, then switch the order of the next two, switch the order of the next two, and so on.

For example, if the list initially stores these values: {"four", "score", "and", "seven", "years", "ago"} your method should switch the first pair, "four", "score", the second pair, "and", "seven", and the third pair, "years", "ago", to yield this list: {"score", "four", "seven", "and", "ago", "years"}

If there are an odd number of values in the list, the final element is not moved.

For example, if the original list had been: {"to", "be", "or", "not", "to", "be", "hamlet"} It would again switch pairs of values, but the final value, "hamlet" would not be moved, yielding this list: {"be", "to", "not", "or", "be", "to", "hamlet"}

- **Theory, important classes and methods**

### **ArrayList in Java**

`ArrayList` is a part of **collection framework** and is present in `java.util` package. It provides us with dynamic arrays in Java.

**Adding Elements:** In order to add an element to an `ArrayList`, we can use the `add()` method

**add(Object):** This method is used to add an element at the end of the `ArrayList`.

### **PROGRAM:**

```
import java.util.*;
class PairDemo {
    public static void swapPairs(ArrayList<String> list) {
        for (int i=0; i<=list.size()-2; i+=2) {
            String str=list.get(i+1);
            list.set(i+1, list.get(i));
            list.set(i, str);
        }
    }
}
public class SPair {
    public static void main(String[] args) {
        int n;
        String str;
        Scanner sr=new Scanner(System.in);
        ArrayList<String> arrlist=new ArrayList<String>();
        System.out.print("Enter number of items: ");
        n=sr.nextInt();
        System.out.print("Enter item: ");
```



```

    for (int i=1; i<=n; i++) {
        str=sr.next();
        arrlist.add(str);
    }
    System.out.println("-----Array List Item-----");
    System.out.println(arrlist.toString());
    PairDemo.swapPairs(arrlist);
    System.out.println("----- Resultant Array List Item-----");
    System.out.println(arrlist.toString());
}
}

```

**Output:**

```

Enter number of items: 4
Enter item: What is your name
-----Array List Item-----
[What, is, your, name]
----- Resultant Array List Item-----
[is, What, name, your]

```

## Problem Statement 23 and solution

### • Problem Statement

Write a method called `alternate` that accepts two Lists of integers as its parameters and returns a new List containing alternating elements from the two lists, in the following order:

- ☐ First element from first list
- ☐ First element from second list
- ☐ Second element from first list
- ☐ Second element from second list
- ☐ Third element from first list
- ☐ Third element from second list
- ☐ . . .

If the lists do not contain the same number of elements, the remaining elements from the longer list should be placed consecutively at the end. For example, for a first list of (1, 2, 3, 4, 5) and a second list of (6, 7, 8, 9, 10, 11, 12), a call of `alternate(list1, list2)` should return a list containing (1, 6, 2, 7, 3, 8, 4, 9, 5, 10, 11, 12). Do not modify the parameter lists passed in.

### • Theory, important classes and methods

#### List Interface

The List interface provides a way to store the ordered collection. It is a child interface of Collection. It is an ordered collection of objects in which duplicate values can be stored. Since List preserves the insertion order, it allows positional access and insertion of elements.

**Adding Elements:** In order to add an element to the list, we can use the `add()` method. This method is overloaded to perform multiple operations based on different parameters. They are:

**`add(Object)`:** This method is used to add an element at the end of the List.

**`add(int index, Object)`:** This method is used to add an element at a specific index in the List.

#### PROGRAM:

```
import java.util.*;
class Alternate {
    public static List<Integer> alternate(List<Integer> list1, List<Integer> list2) {
        Iterator<Integer> i1=list1.iterator();
        Iterator<Integer> i2=list2.iterator();
        List<Integer> result=new ArrayList<Integer>();
        while(i1.hasNext() || i2.hasNext()) {
            if (i1.hasNext())
                result.add(i1.next());
            if (i2.hasNext())
```

```

        result.add(i2.next());
    }
    return result;
}
}
public class AlternateList {
    public static void main(String[] args) {
        List<Integer> L1=new ArrayList<>();
        Collections.addAll(L1, 1, 2, 3, 4, 5);
        List<Integer> L2=new ArrayList<>();
        Collections.addAll(L2, 6, 7, 8, 9, 10,11,12);
        List<Integer> L3=new ArrayList<>();
        L3=Alternate.alternate(L1,L2);
        System.out.println(L1);
        System.out.println(L2);
        System.out.println(L3);
    }
}

```

**Output:**

```

[1, 2, 3, 4, 5]
[6, 7, 8, 9, 10, 11, 12]
[1, 6, 2, 7, 3, 8, 4, 9, 5, 10, 11, 12]

```

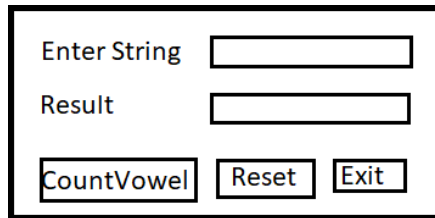
## Problem Statement 24 and solution

### • Problem Statement

Write a GUI program to develop an application that receives a string in one text field, and count number of vowels in a string and returns it in another text field, when the button named “CountVowel” is clicked.

When the button named “Reset” is clicked it will reset the value of textfield one and Textfield two.

When the button named “Exit” is clicked it will closed the application.



### • Theory, important classes and methods

**Swing API** is a set of extensible GUI Components to ease the developer's life to create JAVA based Front End/GUI Applications. It is build on top of AWT API and acts as a replacement of AWT API, since it has almost every control corresponding to AWT controls.

**JButton Class** The class JButton is an implementation of a push button. This component has a label and generates an event when pressed. It can have Image also.

**JTextField Class** The class JTextField is a component which allows editing of a single line of text.

**JLABEL Class** A JLabel object provides text instructions or information on a GUI — display a singleline of read-only text, an image or both text and image.

### Event Handling in Java

#### Event

An event is an Object that describes a state change in a source. Some of the activities that causes event to be generated are:

- ☐ Pressing a Button.
- ☐ Entering a character through Key Board.
- ☐ Selecting an item form a list etc.

### The ActionListener Interface

This interface defines the **actionPerformed( )** method that is invoked when an action event occurs. Its general form is shown here:

**void actionPerformed(ActionEvent ae)**

**PROGRAM:**

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class VowelDemo extends JFrame implements ActionListener {
    JLabel one, two;
    JTextField tstr, tres;
    JButton b1, b2, b3;
    VowelDemo() {
        JPanel p1=new JPanel();
        p1.setLayout(new GridLayout(2, 2));
        FlowLayout layout=new FlowLayout();
        JPanel p2=new JPanel();
        p2.setLayout(layout);
        one=new JLabel(" Enter String: ");
        tstr=new JTextField(20);
        two=new JLabel(" Result");
        tres=new JTextField(20);
        b1=new JButton("CountVowel");
        b2=new JButton("Reset");
        b3=new JButton("Exit");
        p1.add(one);
        p1.add(tstr);
        p1.add(two);
        p1.add(tres);
        p2.add(b1);
        p2.add(b2);
        p2.add(b3);
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);
        add(p1,"North");
        add(p2,"South");
        setVisible(true);
        this.setSize(300,180);
    }
    public void actionPerformed(ActionEvent ae) {
        String str, vstr;
        char ch;
        int ln, c=0;
        str=ae.getActionCommand();
        try {
            vstr=tstr.getText();
            if(str.equals("CountVowel")) {
```

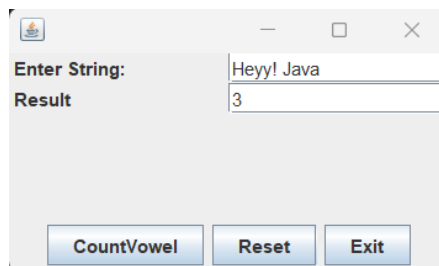
```

        ln=vstr.length();
        for (int i=0; i<ln; i++) {
            ch=vstr.charAt(i);
            ch=Character.toUpperCase(ch);
            switch(ch) {
                case 'A':
                case 'E':
                case 'I':
                case 'O':
                case 'U':
                    c++;
                    break;
            }
        }
        tres.setText(Integer.toString(c));
    }
    else if(str.equals("Reset")) {
        tstr.setText("");
        tres.setText("");
        tstr.requestFocus();
    }
    else if(str.equals("Exit"))
        System.exit(0);
    }
    catch(Exception ob){}
}
}

public class GUIDemo {
    public static void main(String args[]) {
        new VowelDemo();
    }
}

```

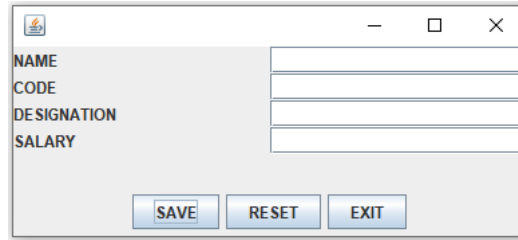
### **Output:**



## Problem Statement 25 and solution

- **Problem Statement**

Create a database of employees with the following fields Name, Code, Designation, & Salary.



- (a) Write a java program to create GUI java application to take employee data from theTextFields and store in database using JDBC connectivity.
- (b) Write a JDBC Program to retrieves all the records from employee database.

- **Theory, important classes and methods**

### What is JDBC?

JDBC stands for Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

### Common JDBC Components

The JDBC API provides the following interfaces and classes –

**DriverManager:** This class manages a list of database drivers. Matches connection requests from the java application with the proper database driver using communication sub protocol. The first driver that recognizes a certain sub protocol under JDBC will be used to establish a database Connection.

**Connection:** Interface used to establish a connection to a database. SQL statementsrun within the context of a connection.

**Statement:** Interface used to send static SQL statements to the database server andobtain results.

**ResultSet:** Interface used to process the results returned from executing an SQLstatement.

**PreparedStatement:** Interface used to send precompiled SQL statements to thedatabase server and obtain results.

### PROGRAM (a):

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

class GridLayoutDemo extends JFrame implements ActionListener {
    JLabel one, two, three, four;
    JTextField tname, tsalary, tcode, tdesig;
    JButton buttonSave, buttonExit, buttonReset;
    GridLayoutDemo() {
        JPanel p1=new JPanel();
```

```

    p1.setLayout(new GridLayout(4,2));
    FlowLayout layout=new FlowLayout();
    JPanel p2=new JPanel();
    p2.setLayout(layout);
    one=new JLabel("NAME");
    tname=new JTextField(20);
    two=new JLabel("CODE");
    tcode=new JTextField(20);
    three=new JLabel("DESIGNATION");
    tdesig=new JTextField(20);
    four=new JLabel("SALARY");
    tsalary=new JTextField(20);
    buttonSave=new JButton("SAVE");
    buttonReset=new JButton("RESET");
    buttonExit=new JButton("EXIT");
    p1.add(one);
    p1.add(tname);
    p1.add(two);
    p1.add(tcode);
    p1.add(three);
    p1.add(tdesig);
    p1.add(four);
    p1.add(tsalary);
    buttonSave.addActionListener(this);
    buttonReset.addActionListener(this);
    buttonExit.addActionListener(this);
    p2.add(buttonSave);
    p2.add(buttonReset);
    p2.add(buttonExit);
    add(p1,"North");
    add(p2,"South");
    setVisible(true);
    this.setSize(400,180);
}

public void actionPerformed(ActionEvent ae) {
    String str, mname, mdesig;
    int mcode, msal;
    str=ae.getActionCommand();
    if(str.equals("RESET")) {
        tname.setText("");
        tcode.setText("");
        tdesig.setText("");
        tsalary.setText("");
        tname.requestFocus();
    }
}

```



```

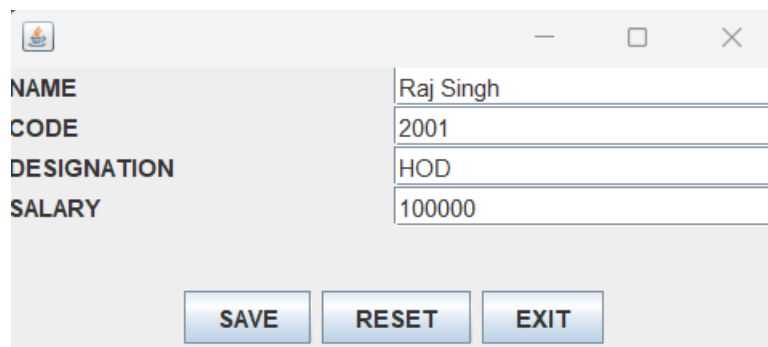
if(str.equals("EXIT"))
    System.exit(0);
try {
    mname=tname.getText();
    mcode=Integer.parseInt(tcode.getText());
    mdesig=tdesig.getText();
    msal=Integer.parseInt(tsalary.getText());
    if(str.equals("SAVE")) {
        try {
            PreparedStatement inst;
            Connection conn;
            Class.forName("com.mysql.jdbc.Driver");
            conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/", "root", "root");
            inst=conn.prepareStatement("INSERT INTO emp VALUES(?,?,?,?)");
            inst.setString(1,mname);
            inst.setInt(2,mcode);
            inst.setString(3,mdesig);
            inst.setInt(4,msal);
            int n=inst.executeUpdate();
            if(n>0)
                System.out.println("Record Inserted");
            else
                System.out.println("Record Not Inserted");
            inst.close();
            conn.close();
        }
        catch(ClassNotFoundException e) {
            e.printStackTrace();
        }
        catch(SQLException w) {
            w.printStackTrace();
        }
    }
}
catch(Exception ob){}
}

class EmpRecord {
    public static void main(String args[]) {
        new GridLayoutDemo();
    }
}

```

**PROGRAM (b):**

```
import java.sql.*;
class MyApp {
    public static void main(String s[]) {
        int mcode, msal;
        String mname, mdesig;
        try {
            Connection conn;
            Statement stmt;
            ResultSet rs;
            Class.forName("com.mysql.jdbc.Driver");
            conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/ ", "root", "root");
            stmt=conn.createStatement();
            rs=stmt.executeQuery("SELECT * FROM emp");
            while(rs.next()) {
                mname=rs.getString(1);
                mcode=rs.getInt(2);
                mdesig=rs.getString(3);
                msal=rs.getInt(4);
                System.out.println(mcode+"      "+mname+" "+mdesig+" "+msal);
            }
            rs.close();
            stmt.close();
            conn.close();
        }
        catch(ClassNotFoundException e) {
            e.printStackTrace();
        }
        catch(SQLException w) {
            w.printStackTrace();
        }
    }
}
```

**Output:**

NAME	Raj Singh
CODE	2001
DESIGNATION	HOD
SALARY	100000

SAVE RESET EXIT