EXERCISES ON INPUT, DECISION AND LOOP

4.1 Introduction

In this Java lab, students will explore Input, Decisions, and Loops. These are important concepts for making software. Through practical exercises, students will learn how to get information from users, make choices in programs, and repeat tasks efficiently. This lab is like a stepping stone, helping students build a strong foundation for writing interactive and useful computer programs. It's a great start for students' journey into software development.

4.2 Prelab Preparation

4.2.1 Objective

By the end of this lab, students should be able to:

1. Understand the basics of taking input from users and displaying output in Java programs.
2. Develop problem-solving skills through hands-on programming exercises that involve user interaction, decision making, and looping.

4.2.2 Background

In this lab on input data in Java, students will get on a foundational journey into the core concepts of acquiring and utilizing input within their programs. This lab lays the groundwork for understanding how Java programs can communicate with users. Key concepts to be explored include:

User Interaction: Students will learn how to facilitate communication between their Java programs and users. They'll discover methods for collecting input from users during program execution, enabling interactive applications.

Input Streams: The lab will introduce the concept of input streams, emphasizing the role they play in reading data sequentially from various sources, such as the keyboard. Students will grasp the significance of input streams in Java's input mechanisms.

Scanner Class: Students will become acquainted with the Scanner class, an essential tool for reading input data. They will learn how to utilize this class to efficiently gather input of various data types, such as integers, strings, and floating-point numbers.

System.in: The lab will explore System.in as the standard input stream, enabling students to read data directly from the keyboard. This foundational skill is crucial for building interactive programs.

Data Validation: Students will understand the importance of validating and handling input data. They will learn basic techniques to ensure that user-provided input meets the expected format and constraints.

Syntax for Inputting Data using Scanner: To input data in Java using the Scanner class, you need to follow these basic steps:

1. **Import the Scanner class:**

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

2. **Create a Scanner object to read input:**

   Scanner scanner = new Scanner(System.in);

3. **Use the Scanner methods to read data. Common methods include:**

   next(): Reads a single word (a string without spaces).
   nextInt(): Reads an integer.
   nextDouble(): Reads a double-precision floating-point number.
   nextLine(): Reads a whole line of text.

   Don't forget to close the Scanner object when you're done to release system resources.

4.2.3 Hardware/Software Requirements

To successfully complete this lab, students will need access to the following:

1. A computer with a suitable text editor or Integrated Development Environment (IDE) installed, such as Eclipse, IntelliJ IDEA, or Visual Studio Code.
2. Java Development Kit (JDK) installed on the computer. Ensure that the JDK is correctly configured, and the system's PATH variable is set to include the JDK's bin directory.

4.2.4 Pre-Lab Questions

1. What is the primary purpose of input data handling in Java programming?
2. How do you create a Scanner object for reading user input?
3. Explain the significance of the System.in stream in Java.
4. What are the common data types that the Scanner class can read from user input?

4.3 Experiment Procedure

1. Ensure you have access to a Java development environment on your computer. Open your Java IDE or text editor for coding.
2. Create a new Java class with a suitable name in your IDE or text editor.
3. Write Java code to solve the problem, incorporating the following elements:
   - Input: Use the Scanner class to read input from the user or define input data within the program.

- Decision: Implement conditional statements (e.g., if, else if, switch) to make decisions based on the input or other conditions.
- Loops: Utilize looping structures (e.g., for, while, do-while) to perform repetitive tasks or iterate through data.

4. Test your code with different input values to ensure it behaves as expected. Debug and fix any issues that arise during testing.
5. Document the problem statement, the approach used to solve it, and the code you've written in your lab notebook or report.
6. Experiment with variations of the problem, explore different scenarios, and modify your code to handle various cases.

4.4 Experiemnts

4.4.1 Add2Integer

Aim:
Write a program called Add2Integers that prompts user to enter two integers. The program shall read the two integers as int; compute their sum; and print the result.

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/

import java.util.Scanner;

public class Add2Integers {
    public static void main (String[] args) {
        // Declare variables
        int number1, number2, sum;

        // Put up prompting messages and read inputs as "int"
        Scanner in = new Scanner(System.in);  // Scan the keyboard for
            input
        System.out.print("Enter first integer: ");  // No newline for
            prompting message
        number1 = in.nextInt();                     // Read next input as "
            int"

        System.out.print("Enter second integer: ");
        number2 = in.nextInt();
        in.close();  // Close Scanner

        // Compute sum
        sum = number1 + number2;

        // Display result
        System.out.println("The sum is: " + sum);   // Print with newline
    }
}
```

Result

Enter first integer: 34
Enter second integer: 25
The sum is: 59

4.4.2 SumProductMinMax3

Aim: Write a program called SumProductMinMax3 that prompts user for three integers. The program shall read the inputs as int; compute the sum, product, minimum and maximum of the three integers; and print the results.

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/
import java.util.Scanner;

public class SumProductMinMax3 {
    public static void main(String args[]){
        // Declare variables
        int number1, number2, number3;
        int sum, product, min, max;

        Scanner in = new Scanner(System.in);

        System.out.print("Enter first integer: ");
        number1 = in.nextInt();

        System.out.print("Enter second integer: ");
        number2 = in.nextInt();

        System.out.print("Enter third integer: ");
        number3 = in.nextInt();
        in.close();

        // Compute sum and product
        sum = number1+number2+number3;
        product = number1*number2*number3;

        min = number1;
        if (number2 < min) {
            min = number2;
        }
        if (number3 < min) {
            min = number3;
        }

        // Compute max - similar to min
        max = number1;
        if(number2 > max){
            max = number2;
        }
```

```
        if(number3 > max){
            max = number3;
        }

        // Print results
        System.out.println("The sum is: " + sum);
        System.out.println("The product is: " + product);
        System.out.println("The min is: " + min);
        System.out.println("The max is: " + max);
    }
}
```

Result

Enter first integer: 56
Enter second integer: 45
Enter third integer: 89
The sum is: 190
The product is: 224280
The min is: 45
The max is: 89

4.4.3 CircleComputation

Aim: Write a program called CircleComputation that prompts user for the radius of a circle in floating point number. The program shall read the input as double; compute the diameter, circumference, and area of the circle in double; and print the values rounded to 2 decimal places. Use System-provided constant Math.PI for pi.

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/
import java.util.Scanner;
public class CircleComputation {
    public static void main(String args[]){
        // Declare variables
        double radius, diameter, circumference, area;

        Scanner in = new Scanner(System.in);
        System.out.print("Enter the radius: ");
        radius = in.nextDouble();  // read input as double

        // Compute in "double"
        diameter = 2.0 * radius;
        area = Math.PI * radius * radius;
        circumference = 2.0 * Math.PI * radius;

        System.out.printf("Diameter is: %.2f%n", diameter);
        System.out.printf("Area is: %.2f%n", area);
        System.out.printf("Circumference is: %.2f%n", circumference);
    }
}
```

Result

Enter the radius: 5
Diameter is: 10.00
Area is: 78.54
Circumference is: 31.42

4.4.4 Swap2Integers

Aim: Write a program called Swap2Integers that prompts user for two integers. The program shall read the inputs as int, save in two variables called number1 and number2; swap the contents of the two variables; and print the results.

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/
import java.util.Scanner;

public class Swap2Integers {
    public static void main(String args[]){
        int number1, number2;
        Scanner in = new Scanner(System.in);  // Scan the keyboard

        System.out.print("Enter first integer: ");
        number1 = in.nextInt();

        System.out.print("Enter second integer: ");
        number2 = in.nextInt();

        int temp = number1;
        number1 = number2;
        number2 = temp;
        System.out.println("After the swap, first integer is: "+number1+ ",
            second integer is: "+number2);
    }
}
```

Result

Enter first integer: 56
Enter second integer: 34
After the swap, first integer is: 34, second integer is: 56

4.4.5 IncomeTaxCalculator

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

For example, suppose that the taxable income is $85000, the income tax payable is $20000*0% + $20000*10% + $20000*20% + $25000*30%.

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

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/
import java.util.Scanner;

public class IncomeTaxCalculator {
    public static void main(String args[]){
        // Declare constants first (variables may use these constants)
        // The keyword "final" marked these as constant (i.e., cannot be
           changed).
        // Use uppercase words joined with underscore to name constants
        final double TAX_RATE_ABOVE_20K = 0.1;
        final double TAX_RATE_ABOVE_40K = 0.2;
        final double TAX_RATE_ABOVE_60K = 0.3;

        // Declare variables
        int taxableIncome;
        double taxPayable;

        Scanner in = new Scanner(System.in);  // Scan the keyboard

        System.out.print("Enter the taxable income: $");
        taxableIncome = in.nextInt();


        // Compute tax payable in "double" using a nested-if to handle 4
           cases
        if (taxableIncome <= 20000) {         // [0, 20000]
            taxPayable = 0;
            System.out.println(taxPayable);
        } else if (taxableIncome <= 40000) {  // [20001, 40000]
            taxPayable = (taxableIncome-20000) * TAX_RATE_ABOVE_20K;
        } else if (taxableIncome <= 60000) {  // [40001, 60000]
            taxPayable =  20000*TAX_RATE_ABOVE_20K + (taxableIncome-40000)
               * TAX_RATE_ABOVE_40K;
        } else {                              // [60001, ]
            taxPayable = (20000*TAX_RATE_ABOVE_20K) + (20000 *
               TAX_RATE_ABOVE_40K) + (taxableIncome-60000) *
               TAX_RATE_ABOVE_60K;
        }
        // Print results rounded to 2 decimal places
        System.out.printf("The income tax payable is: $%.2f%n", taxPayable)
           ;
    }
}
```

Result

Enter the taxable income: $41234
The income tax payable is: $2246.80

Enter the taxable income: $67891
The income tax payable is: $8367.

Enter the taxable income: $85432
The income tax payable is:$13629.60

Enter the taxable income: $12345
The income tax payable is: $0.00

4.4.6 IncomeTaxCalculatorWithSentinel

Aim: Based on the previous exercise, write a program called IncomeTaxCalculatorWithSentinel which shall repeat the calculation until user enter -1.

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/
import java.util.Scanner;

public class IncomeTaxCalculatorWithSentinel {
    public static void main(String args[]){
        // Declare constants first (variables may use these constants)
        // The keyword "final" marked these as constant (i.e., cannot be
            changed).
        // Use uppercase words joined with underscore to name constants
        final double TAX_RATE_ABOVE_20K = 0.1;
        final double TAX_RATE_ABOVE_40K = 0.2;
        final double TAX_RATE_ABOVE_60K = 0.3;
        final int SENTINEL = -1;    // Terminating value for input

        // Declare variables
        int taxableIncome;
        double taxPayable;

        Scanner in = new Scanner(System.in);  // Scan the keyboard

        System.out.print("Enter the taxable income (or -1 to end): $");
        taxableIncome = in.nextInt();

        while (taxableIncome != SENTINEL){
            // Compute tax payable in "double" using a nested-if to handle
                4 cases
            if (taxableIncome <= 20000) {         // [0, 20000]
                taxPayable = 0;
                System.out.println(taxPayable);
            } else if (taxableIncome <= 40000) {  // [20001, 40000]
                taxPayable = (taxableIncome-20000) * TAX_RATE_ABOVE_20K;
            } else if (taxableIncome <= 60000) {  // [40001, 60000]
                taxPayable =  20000*TAX_RATE_ABOVE_20K + (taxableIncome
                    -40000) * TAX_RATE_ABOVE_40K;
```

```
            } else {                                // [60001, ]
                taxPayable = (20000*TAX_RATE_ABOVE_20K) + (20000 *
                    TAX_RATE_ABOVE_40K) + (taxableIncome-60000) *
                    TAX_RATE_ABOVE_60K;
            }
            // Print results rounded to 2 decimal places
            System.out.printf("The income tax payable is: $%.2f%n",
                taxPayable);

            System.out.print("Enter the taxable income (or -1 to end): $");
            taxableIncome = in.nextInt();
        }
        System.out.print("Bye. Have a good day!");
    }
}
```

Result

Enter the taxable income (or -1 to end): $41000
The income tax payable is: $2200.00

Enter the taxable income (or -1 to end): $62000
The income tax payable is: $6600.00

Enter the taxable income (or -1 to end): $73123
The income tax payable is: $9936.90

Enter the taxable income (or -1 to end): $84328
The income tax payable is: $13298.40

Enter the taxable income: $-1
Bye. Have a good day!

4.4.7 PensionContributionCalculatorWithSentinel

Aim: Both the employer and the employee are mandated to contribute a certain percentage of the employeeś salary towards the employeeś pension fund. The rate is tabulated as follows:

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

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/
import java.util.Scanner;
```

```
public class IncomeTaxCalculatorWithSentinel {
    public static void main(String args[]){
        // Declare constants first (variables may use these constants)
        // The keyword "final" marked these as constant (i.e., cannot be
           changed).
        // Use uppercase words joined with underscore to name constants
        final double TAX_RATE_ABOVE_20K = 0.1;
        final double TAX_RATE_ABOVE_40K = 0.2;
        final double TAX_RATE_ABOVE_60K = 0.3;
        final int SENTINEL = -1;    // Terminating value for input

        // Declare variables
        int taxableIncome;
        double taxPayable;

        Scanner in = new Scanner(System.in);  // Scan the keyboard

        System.out.print("Enter the taxable income (or -1 to end): $");
        taxableIncome = in.nextInt();

        while (taxableIncome != SENTINEL){
            // Compute tax payable in "double" using a nested-if to handle
               4 cases
            if (taxableIncome <= 20000) {         // [0, 20000]
                taxPayable = 0;
                System.out.println(taxPayable);
            } else if (taxableIncome <= 40000) {  // [20001, 40000]
                taxPayable = (taxableIncome-20000) * TAX_RATE_ABOVE_20K;
            } else if (taxableIncome <= 60000) {  // [40001, 60000]
                taxPayable =  20000*TAX_RATE_ABOVE_20K + (taxableIncome
                   -40000) * TAX_RATE_ABOVE_40K;
            } else {                              // [60001, ]
                taxPayable = (20000*TAX_RATE_ABOVE_20K) + (20000 *
                   TAX_RATE_ABOVE_40K) + (taxableIncome-60000) *
                   TAX_RATE_ABOVE_60K;
            }
            // Print results rounded to 2 decimal places
            System.out.printf("The income tax payable is: $%.2f%n",
               taxPayable);

            System.out.print("Enter the taxable income (or -1 to end): $");
            taxableIncome = in.nextInt();
        }
        System.out.print("Bye. Have a good day!");
    }
}
```

Result

Enter the monthly salary (or -1 to end): $5123
Enter the age: 21
The employee's contribution is: $1024.60
The employer's contribution is: $870.91
The total contribution is: $1895.51

Enter the monthly salary (or -1 to end): $5123

Enter the age: 64
The employee's contribution is: 384.22
$The employer's contribution is$:461.07
The total contribution is: $845.30

Enter the monthly salary (or -1 to end): $-1
Bye. Have a good day!

4.4.8 SalesTaxCalculator

Aim: A sales tax of 7tags should include the sales tax. For example, if an item has a price tag of $107, the actual price is $100 and $7 goes to the sales tax.

Write a program using a loop to continuously input the tax-inclusive price (in double); compute the actual price and the sales tax (in double); and print the results rounded to 2 decimal places. The program shall terminate in response to input of -1; and print the total price, total actual price, and total sales tax

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/
import java.util.Scanner;

public class SalesTaxCalculator {
    public static void main(String args[]){
        // Declare constants
        final double SALES_TAX_RATE = 0.07;
        final int SENTINEL = -1;

        // Declare variables
        double price, actualPrice, salesTax;  // inputs and results
        double totalPrice = 0.0, totalActualPrice = 0.0, totalSalesTax =
            0.0;

        Scanner in = new Scanner(System.in);
        // Read the first input to "seed" the while loop
        System.out.print("Enter the tax-inclusive price in dollars (or -1
            to end): ");
        price =  in.nextDouble();

        while(price != SENTINEL) {

            actualPrice = price/(1+(SALES_TAX_RATE));
            salesTax = price - actualPrice;

            totalPrice += price;
            totalActualPrice += actualPrice;
            totalSalesTax += (int)salesTax;

            System.out.printf("Actual Price is: $%.2f ", actualPrice);
            System.out.printf("Sales Tax is: $%.2f%n", salesTax);
```

```
            System.out.print("Enter the tax-inclusive price in dollars (or
                -1 to end): ");
            price =  in.nextDouble();
        }
        System.out.printf("Total Price is: $%.2f%n", totalPrice);
        System.out.printf("Total actual Price is: $%.2f%n",
           totalActualPrice);
        System.out.printf("Total Sales Tax is: $%.2f%n", totalSalesTax);
    }
}
```

Result

Enter the tax-inclusive price in dollars (or -1 to end): 107
Actual Price is: $100.00 Sales Tax is: $7.00
Enter the tax-inclusive price in dollars (or -1 to end): 214
Actual Price is: $200.00 Sales Tax is: $14.00
Enter the tax-inclusive price in dollars (or -1 to end): 321
Actual Price is: $300.00 Sales Tax is: $21.00
Enter the tax-inclusive price in dollars (or -1 to end): -1
Total Price is: $642.00
Total actual Price is: $600.00
Total Sales Tax is: $42.00

4.4.9 ReverseInt

Aim: Write a program that prompts user for a positive integer. The program shall read the input as int; and print the ”reverse” of the input integer.

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/
import java.util.Scanner;

public class ReverseInt {
    public static void main(String args[]){
        // Declare variables
        int inNumber;   // to be input
        int inDigit;    // each digit
        int rev = 0;

        Scanner in = new Scanner(System.in);
        System.out.print("Enter a positive integer: ");
        inNumber = in.nextInt();

        // Extract and drop the "last" digit repeatably using a while-loop
           with modulus/divide operations
        while (inNumber > 0) {
            inDigit = inNumber % 10; // extract the "last" digit
            rev = rev*10+inDigit;
            inNumber /= 10;          // drop "last" digit and repeat
```

```
        }
        System.out.println("The reverse is: "+rev);
    }
}
```

Result

Enter a positive integer: 12345
The reverse is: 54321

Enter a positive integer: 32564
The reverse is: 46523

4.4.10 SumOfDigitsInt

Aim: Write a program that prompts user for a positive integer. The program shall read the input as int; compute and print the sum of all its digits.

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/
import java.util.Scanner;

public class SumOfDigitsInt {
    public static void main(String args[]){
        // Declare variables
        int inNumber;   // to be input
        int inDigit;    // each digit
        int sum = 0;

        Scanner in = new Scanner(System.in);
        System.out.print("Enter a positive integer: ");
        inNumber = in.nextInt();

        // Extract and drop the "last" digit repeatably using a while-loop
           with modulus/divide operations
        while (inNumber > 0) {
            inDigit = inNumber % 10; // extract the "last" digit
            sum +=inDigit;
            inNumber /= 10;          // drop "last" digit and repeat
        }
        System.out.println("The sum of all digits is: "+sum);
    }
}
```

Result

Enter a positive integer: 12345
The sum of all digits is: 15

Enter a positive integer: 34261
The sum of all digits is: 16

4.4.11 InputValidation

Aim: Your program often needs to validate the user's inputs, e.g., marks shall be between 0 and 100.

Write a program that prompts user for an integer between 0-10 or 90-100. The program shall read the input as int; and repeat until the user enters a valid input.

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/
import java.util.Scanner;

public class InputValidation {
    public static void main(String args[]){
        // Declare variables
        int numberIn;      // to be input
        boolean isValid;   // boolean flag to control the loop

        Scanner in = new Scanner(System.in);

        // Use a do-while loop controlled by a boolean flag
        // to repeatably read the input until a valid input is entered
        isValid = false;   // default assuming input is not valid

        do{
            // Prompt and read input
            System.out.print("Enter a number between 0-10 or 90-100: ");
            numberIn = in.nextInt();
            // Validate input by setting the boolean flag accordingly
            if((numberIn>=0 && numberIn<=10) || (numberIn>=90 && numberIn
               <=100)) {
                isValid = true;   // exit the loop
                break;
            } else {
                System.out.println("Invalid input, try again...");  //
                   Print error message and repeat
            }
        }while (!isValid);
        System.out.println("You have entered: "+numberIn);
    }
}
```

Result

```
Enter a number between 0-10 or 90-100: -5
Invalid input, try again...
Enter a number between 0-10 or 90-100: 45
Invalid input, try again...
Enter a number between 0-10 or 90-100: 108
Invalid input, try again...
Enter a number between 0-10 or 90-100: 99
You have entered: 99
```

4.4.12 AverageWithInputValidation

Aim: Write a program that prompts user for the mark (between 0-100 in int) of 3 students; computes the average (in double); and prints the result rounded to 2 decimal places. Your program needs to perform input validation.

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/
import java.util.Scanner;

public class AverageWithInputValidation {
    public static void main(String args[]){
        // Declare variables
        int numberIn;      // to be input
        double average=0.0;
        int counter=0, sum =0;

        Scanner in = new Scanner(System.in);

        do {
            // Prompt and read input
            System.out.print("Enter a number between 0-100: ");
            numberIn = in.nextInt();
            // Validate input by setting the boolean flag accordingly
            if((numberIn>=0 && numberIn<=100)) {
                sum += numberIn;
                counter++;
            } else {
                System.out.println("Invalid input, try again...");  //
                   Print error message and repeat
            }
        } while (counter!=3);
        average = sum/3;
        System.out.printf("The average is: %.2f",average);
    }
}
```

Result

Enter a number between 0-100: -1
Invalid input, try again...
Enter a number between 0-100: 45
Enter a number between 0-100: 106
Invalid input, try again...
Enter a number between 0-100: 92
Enter a number between 0-100: 23
The average is: 53.00

4.5 Post-Lab Viva Voce Questions

1. Describe the role of the Scanner class in reading input data. How does it simplify the process of data input?
2. How would you read a whole line of text from a user using the Scanner class?
3. What does the break statement do in a loop, and when might you use it?
4. How can you prevent an infinite loop when using a while or do-while loop in Java?

4.6 Further Probing Experiments

1. Develop a Java program called SphereComputation that prompts user for the radius of a sphere in floating point number. The program shall read the input as double; compute the volume and surface area of the sphere in double; and print the values rounded to 2 decimal places.
2. Write a program called CylinderComputation that prompts user for the base radius and height of a cylinder in floating point number. The program shall read the inputs as double; compute the base area, surface area, and volume of the cylinder; and print the values rounded to 2 decimal places.

EXERCISES ON NESTED-LOOPS

5.1 Introduction

In this lab, we will explore the concept of nested loops, a flexible programming technique that allows us to solve more complicated problems. Just like a set of building blocks, nested loops enable us to construct complex patterns and structures within our programs. Our goal is to understand how to use nested loops effectively, mastering a skill that will empower us to tackle a wide range of challenges in Java programming. By the end of this lab, you will have a deeper understanding of how to apply nested loops to create exclusive solutions, enhancing your programming toolkit significantly. So, let's dive in and unlock the full potential of nested loops in Java!

5.2 Prelab Preparation

5.2.1 Objective

By the end of this lab, students should be able to:

1. Understand the syntax of nested loops and using them effectively to solve problems that involve multi-layered iterations.
2. learn strategies for optimizing the efficiency of the nested loop structures, ensuring that the programs run smoothly and perform well, even in scenarios involving extensive iterations.

5.2.2 Background

It is also possible to place a loop inside another loop. This is called a nested loop. The "inner loop" will be executed one time for each iteration of the "outer loop".

Syntax for Nested for loop:

```
for ( initialization; condition; increment ) {
    for ( initialization; condition; increment ) {
        // statement of inside loop
    }
    // statement of outer loop
}
```

Syntax for Nested while loop:

```
while(condition) {
    while(condition) {
        // statement of inside loop
    }
    // statement of outer loop}
```

Syntax for Nested do-while loop:

```
do{
    do{
        // statement of inside loop
    }while(condition);
    // statement of outer loop
}while(condition);
```

Note: There is no rule that a loop must be nested inside its own type. In fact, there can be any type of loop nested inside any type and to any level.

Syntax:

```
do{
    while(condition) {
        for ( initialization; condition; increment ) {
            // statement of inside for loop
        }
        // statement of inside while loop
    }
    // statement of outer do-while loop
}while(condition);
```

5.2.3 Hardware/Software Requirements

To successfully complete this lab, students will need access to the following:

1. A computer with a suitable text editor or Integrated Development Environment (IDE) installed, such as Eclipse, IntelliJ IDEA, or Visual Studio Code.
2. Java Development Kit (JDK) installed on the computer. Ensure that the JDK is correctly configured, and the system's PATH variable is set to include the JDK's bin directory.

5.2.4 Pre-Lab Questions

1. What is a nested loop in Java, and how does it differ from a regular loop?
2. Explain the syntax of a nested "for" loop in Java with an example.
3. How are nested "while" loops implemented in Java, and why might you use them?
4. How many loops can be nested?

5.3 Experiment Procedure

1. Ensure you have access to a Java development environment on your computer. Open your Java IDE or text editor for coding.
2. Create a new Java class with a suitable name in your IDE or text editor.
3. Write Java code to solve the problem, incorporating the following elements:
    - Input: Use the Scanner class to read input from the user or define input data within the program.

- Decision: Implement conditional statements (e.g., if, else if, switch) to make decisions based on the input or other conditions.
- Loops: Utilize looping structures (e.g., for, while, do-while) to perform repetitive tasks or iterate through data.

4. Test your code with different input values to ensure it behaves as expected. Debug and fix any issues that arise during testing.
5. Document the problem statement, the approach used to solve it, and the code you've written in your lab notebook or report.
6. Experiment with variations of the problem, explore different scenarios, and modify your code to handle various cases.

5.4 Experiemnts

5.4.1 SquarePattern (nested-loop)

Aim: Write a program called SquarePattern that prompts user for the size (a non-negative integer in int); and prints the following square pattern using two nested for-loops.

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/
import java.util.Scanner;

public class SquarePattern {
    public static void main(String args[]){
        int size;
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the size: ");
        size = in.nextInt();
        for (int row = 1; row <= size; row++) {
            for (int col = 1; col <= size; col++) {
                System.out.print("# ");
            }
            System.out.println();
        }
    }
}
```

Result

```
Enter the size: 5
# # # # #
# # # # #
# # # # #
# # # # #
# # # # #
```

5.4.2 SquarePattern (nested-loop)

Aim: Write a program called CheckerPattern that prompts user for the size (a non-negative integer in int); and prints the following checkerboard pattern.

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/

import java.util.Scanner;

public class CheckerPattern {
    public static void main(String args[]){
        int size;
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the size: ");
        size = in.nextInt();
        for (int row = 1; row <= size; row++) {
            for (int col = 1; col <= size; col++) {
                if ((row % 2) == 0) {   // row 2, 4, 6, ...
                    System.out.print(" #");
                }
                else{
                    System.out.print("# ");
                }
            }
            System.out.println();
        }
    }
}
```

Result

```
Enter the size: 8
# # # # # # # #
 # # # # # # # #
# # # # # # # #
 # # # # # # # #
# # # # # # # #
 # # # # # # # #
# # # # # # # #
 # # # # # # # #
```

5.4.3 MultiplicationTable (nested-loop)

Aim: Write a program called MultiplicationTable that prompts user for the size (a positive integer in int); and prints the multiplication table

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
```

```
* @author cvrpa
*/

import java.util.Scanner;

public class MultiplicationTable {
    public static void main(String args[]){
        int size;
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the size: ");
        size = in.nextInt();

        System.out.printf("   * |");
        for (int row = 1; row <= size; row++) {
            System.out.printf("%4d",row);
        }

        System.out.println();
        for (int row = 0; row <= size; row++) {
            System.out.print("----");
        }
        System.out.print("--");
        System.out.println();

        for (int row = 1; row <= size; row++) {
            System.out.printf("%4d |",row);
            for (int col = 1; col <= size; col++) {
                System.out.printf("%4d",row*col);
            }
            System.out.println();
        }
    }
}
```

Result

```
Enter the size: 10
 * |   1   2   3   4   5   6   7   8   9  10
-----------------------------------------------
 1 |   1   2   3   4   5   6   7   8   9  10
 2 |   2   4   6   8  10  12  14  16  18  20
 3 |   3   6   9  12  15  18  21  24  27  30
 4 |   4   8  12  16  20  24  28  32  36  40
 5 |   5  10  15  20  25  30  35  40  45  50
 6 |   6  12  18  24  30  36  42  48  54  60
 7 |   7  14  21  28  35  42  49  56  63  70
 8 |   8  16  24  32  40  48  56  64  72  80
 9 |   9  18  27  36  45  54  63  72  81  90
10 |  10  20  30  40  50  60  70  80  90 100
```

5.4.4 TriangularPattern (nested-loop)

Aim: Write a program called TriangularPatternX that prompts user for the size (a non-negative integer in int); and prints triangular patterns (Refer Syllabus)

Program

```
/**
*author Dr.C V Rama Padmaja
```

```
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/

import java.util.Scanner;

public class TriangularPattern {
    public static void main(String args[]){
        int size;
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the size: ");
        size = in.nextInt();

        // For displaying (a)
        for (int row = 1; row <= size; row++) {  // row = 1, 2, 3, ...,
           size
            // Inner loop to print each of the columns of a particular row
            for (int col = 1; col <= size; col++) {  // col = 1, 2, 3, ...,
                size
                if (row >= col) {
                    System.out.print("# ");
                } else {
                    System.out.print("  ");  // Need to print the "leading"
                        blanks
                }
            }
            // Print a newline after printing all the columns
            System.out.println();
        }

        System.out.println("\n\n");
        // For displaying (b)
        for (int row = 1; row <= size; row++) {  // row = 1, 2, 3, ...,
           size
            // Inner loop to print each of the columns of a particular row
            for (int col = 1; col <= size; col++) {  // col = 1, 2, 3, ...,
                size
                if (row + col <= size + 1) {
                    System.out.print("# ");
                } else {
                    System.out.print("  ");  // Need to print the "leading"
                        blanks
                }
            }
            // Print a newline after printing all the columns
            System.out.println();
        }


        System.out.println("\n\n");
        // For displaying (c)
        for (int row = 1; row <= size; row++) {  // row = 1, 2, 3, ...,
           size
            // Inner loop to print each of the columns of a particular row
            for (int col = 1; col <= size; col++) {  // col = 1, 2, 3, ...,
                size
                if (row <= col) {
```

```
                    System.out.print("# ");
                } else {
                    System.out.print("  ");  // Need to print the "leading"
                        blanks
                }
            }
            // Print a newline after printing all the columns
            System.out.println();
        }

        System.out.println("\n\n");
        // For displaying (d)
        for (int row = 1; row <= size; row++) {  // row = 1, 2, 3, ...,
            size
            // Inner loop to print each of the columns of a particular row
            for (int col = 1; col <= size; col++) {  // col = 1, 2, 3, ...,
                size
                if (row + col >= size + 1) {
                    System.out.print("# ");
                } else {
                    System.out.print("  ");  // Need to print the "leading"
                        blanks
                }
            }
            // Print a newline after printing all the columns
            System.out.println();
        }
    }
}
```

Result

```
Enter the size: 5
#
# #
# # #
# # # #
# # # # #


# # # # #
# # # #
# # #
# #
#


# # # # #
  # # # #
    # # #
      # #
        #


        #
      # #
    # # #
  # # # #
# # # # #
```

5.4.5 BoxPattern (nested-loop)

Aim: Write a program called BoxPatternX that prompts user for the size (a nonnegative integer in int) and print the box patterns

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/

import java.util.Scanner;

public class BoxPatternX {
    public static void main(String args[]){
        int size;
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the size: ");
        size = in.nextInt();

        // For displaying (a)
        for (int row = 1; row <= size; row++) {  // row = 1, 2, 3, ...,
           size
            // Inner loop to print each of the columns of a particular row
            for (int col = 1; col <= size; col++) {  // col = 1, 2, 3, ...,
                size
                if (row == 1 || row == size || col == 1 || col == size) {
                    System.out.print("# ");
                } else {
                    System.out.print("  ");  // Need to print the "leading"
                        blanks
                }
            }
            // Print a newline after printing all the columns
            System.out.println();
        }

        System.out.println("\n");
        // For displaying (b)
        for (int row = 1; row <= size; row++) {  // row = 1, 2, 3, ...,
           size
            // Inner loop to print each of the columns of a particular row
            for (int col = 1; col <= size; col++) {  // col = 1, 2, 3, ...,
                size
                if (row == 1 || row == size || row == col) {
                    System.out.print("# ");
                } else {
                    System.out.print("  ");  // Need to print the "leading"
                        blanks
                }
            }
            // Print a newline after printing all the columns
            System.out.println();
        }

        System.out.println("\n");
```

```
        // For displaying (c)
        for (int row = 1; row <= size; row++) {  // row = 1, 2, 3, ...,
            size
            // Inner loop to print each of the columns of a particular row
            for (int col = 1; col <= size; col++) {  // col = 1, 2, 3, ...,
                size
                if (row == 1 || row == (size-col+1) || row == size) {
                    System.out.print("# ");
                } else {
                    System.out.print("  ");  // Need to print the "leading"
                        blanks
                }
            }
            // Print a newline after printing all the columns
            System.out.println();
        }

        System.out.println("\n");
        // For displaying (d)
        for (int row = 1; row <= size; row++) {  // row = 1, 2, 3, ...,
            size
            // Inner loop to print each of the columns of a particular row
            for (int col = 1; col <= size; col++) {  // col = 1, 2, 3, ...,
                size
                if (row == 1 || row == size || row == (size-col+1) || row
                    == col) {
                    System.out.print("# ");
                } else {
                    System.out.print("  ");  // Need to print the "leading"
                        blanks
                }
            }
            // Print a newline after printing all the columns
            System.out.println();
        }

        System.out.println("\n");
        // For displaying (e)
        for (int row = 1; row <= size; row++) {  // row = 1, 2, 3, ...,
            size
            // Inner loop to print each of the columns of a particular row
            for (int col = 1; col <= size; col++) {  // col = 1, 2, 3, ...,
                size
                if (row == 1 || row == size || row == (size-col+1) || row
                    == col || col == 1 || col == size) {
                    System.out.print("# ");
                } else {
                    System.out.print("  ");  // Need to print the "leading"
                        blanks
                }
            }
            // Print a newline after printing all the columns
            System.out.println();
        }
    }
}
```

Result

```
Enter the size: 8
# # # # # # # #
#               #
#               #
#               #
#               #
#               #
#               #
# # # # # # # #


# # # # # # # #
  #
    #
      #
        #
          #
            #
# # # # # # # #


# # # # # # # #
            #
          #
        #
      #
    #
  #
# # # # # # # #


# # # # # # # #
  #           #
    #       #
      # #
      # #
    #       #
  #           #
# # # # # # # #


# # # # # # # #
# #           # #
#   #       #   #
#     # #       #
#     # #       #
#   #       #   #
# #           # #
# # # # # # # # #
```

5.4.6 HillPattern (nested-loop)

Aim: Write 3 programs called HillPatternX (X = A, B, C, D) that prompts user for the size (a nonnegative integer in int); and prints the pattern

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/

import java.util.Scanner;

public class HillPatternX {
    public static void main(String args[]){
        int size;
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the size: ");
        size = in.nextInt();

        // For displaying (a)
        System.out.println("HillPattern (a)");
        for (int row = 1; row <= size; row++) {  // row = 1, 2, 3, ...,
           size
            // Inner loop to print each of the columns of a particular row
            for(int col = 1; col <= size-row; col++) {
                System.out.print("  ");
            }
            for(int col = 1; col <= 2*row-1; col++){
                System.out.print("# ");
            }
            // Print a newline after printing all the columns
            System.out.println();
        }
        System.out.println();

        // For displaying (b)
        System.out.println("HillPattern (b)");
        for (int row = 1; row <= size; row++) {  // row = 1, 2, 3, ...,
           size
            // Inner loop to print each of the columns of a particular row
            for(int col = 1; col <= row-1; col++) {
                System.out.print("  ");
            }
            for(int col = 1; col <= 2*(size-row)+1; col++){
                System.out.print("# ");
            }
            // Print a newline after printing all the columns
            System.out.println();
        }
        System.out.println("\n");


        // For displaying (c)
        System.out.println("HillPattern (c)");
        for (int row = 1; row <= size; row++) {  // row = 1, 2, 3, ...,
           size
            // Inner loop to print each of the columns of a particular row
            for(int col = 1; col <= size-row; col++) {
                System.out.print("  ");
            }
            for(int col = 1; col <= 2*row-1; col++){
```

```
                System.out.print("# ");
            }
            // Print a newline after printing all the columns
            System.out.println();
        }
        for (int row = 2; row <= size; row++) {  // row = 1, 2, 3, ...,
           size
            // Inner loop to print each of the columns of a particular row
            for(int col = 1; col <= row-1; col++) {
                System.out.print("  ");
            }
            for(int col = 1; col <= 2*(size-row)+1; col++){
                System.out.print("# ");
            }
            // Print a newline after printing all the columns
            System.out.println();
        }
        System.out.println("\n");
    }
}
```

Result

```
Enter the size: 6
HillPattern (a)
          #
        # # #
      # # # # #
    # # # # # # #
  # # # # # # # # #
# # # # # # # # # # #

HillPattern (b)
# # # # # # # # # # #
  # # # # # # # # #
    # # # # # # #
      # # # # #
        # # #
          #


HillPattern (c)
          #
        # # #
      # # # # #
    # # # # # # #
  # # # # # # # # #
# # # # # # # # # # #
  # # # # # # # # #
    # # # # # # #
      # # # # #
        # # #
          #
```

5.4.7 NumberPattern (nested-loop)

Aim: Write a program called NumberPatternX that prompts user for the size (a nonnegative integer in int); and prints the NumberPattern

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/

import java.util.Scanner;

public class NumberPattern {
    public static void main(String args[]){
        int size;
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the size: ");
        size = in.nextInt();

        // For displaying (a)
        for (int row = 1; row <= size; row++) {  // row = 1, 2, 3, ...,
           size
            // Inner loop to print each of the columns of a particular row
            for (int col = 1; col <= size; col++) {  // col = 1, 2, 3, ...,
                size
                if (row >= col) {
                    System.out.print(col + " ");
                } else {
                    System.out.print("  ");  // Need to print the "leading"
                        blanks
                }
            }
            // Print a newline after printing all the columns
            System.out.println();
        }

        System.out.println("\n\n");
        // For displaying (b)
        for (int row = 1; row <= size; row++) {  // row = 1, 2, 3, ...,
           size
            // Inner loop to print each of the columns of a particular row
            for (int col = 1; col <= size; col++) {  // col = 1, 2, 3, ...,
                size
                if (row + col <= size + 1) {
                    System.out.print(col + " ");
                } else {
                    System.out.print("  ");  // Need to print the "leading"
                        blanks
                }
            }
            // Print a newline after printing all the columns
            System.out.println();
        }


        System.out.println("\n\n");
        // For displaying (c)
        for (int row = 1; row <= size; row++) {  // row = 1, 2, 3, ...,
           size
            // Inner loop to print each of the columns of a particular row
```

```
            for (int col = 1; col <= size; col++) {  // col = 1, 2, 3, ...,
                size
                if (row <= col) {
                    System.out.print(col + " ");
                } else {
                    System.out.print("  ");  // Need to print the "leading"
                        blanks
                }
            }
            // Print a newline after printing all the columns
            System.out.println();
        }

        System.out.println("\n\n");
        // For displaying (d)
        for (int row = 1; row <= size; row++) {  // row = 1, 2, 3, ...,
           size
            // Inner loop to print each of the columns of a particular row
            for (int col = 1; col <= size; col++) {  // col = 1, 2, 3, ...,
                size
                if (row + col >= size + 1) {
                    System.out.print(col + " ");
                } else {
                    System.out.print("  ");  // Need to print the "leading"
                        blanks
                }
            }
            // Print a newline after printing all the columns
            System.out.println();
        }
    }
}
```

Result

```
Enter the size: 8
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8



1 2 3 4 5 6 7 8
1 2 3 4 5 6 7
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

```
1 2 3 4 5 6 7 8
  2 3 4 5 6 7 8
    3 4 5 6 7 8
      4 5 6 7 8
        5 6 7 8
          6 7 8
            7 8
              8


              8
            7 8
          6 7 8
        5 6 7 8
      4 5 6 7 8
    3 4 5 6 7 8
  2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
```

5.5 Sample Viva Voce Questions

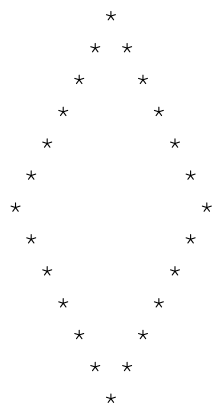
1. Explain the concept of a "loop within a loop" and why it is useful.
2. What precautions should you take to prevent infinite loops when using nested loops?
3. Provide an example of a practical application where nested loops are commonly used in Java programming.
4. What is the output of a simple nested loop program that prints a pattern of stars or numbers?

5.6 Further Probing Experiments

1. Write a program called HillPattern that prompts user for the size (a non-negative integer in int) and prints the pattern as shown:

```
# # # # # # # # # # #
# # # # #   # # # # #
# # # #       # # # #
# # #           # # #
# #               # #
#                   #
# #               # #
# # #           # # #
# # # #       # # # #
# # # # #   # # # # #
# # # # # # # # # # #
```

2. Write a program called DPattern that prompts user for the size (a non-negative integer in int) and prints the pattern as shown:



MAGIC(SPECIAL) NUMBERS

6.1 Introduction

In this session, we will explore a unique category of numbers, such as Armstrong numbers, neon numbers, spy numbers, palindrome numbers, and more. These special numbers possess remarkable properties that make them stand out in the world of mathematics and programming. As we unravel the secrets behind these magical digits, you will gain valuable insights into the power of logic and algorithms in programming, enabling you to apply your newfound knowledge to solve complex problems and create efficient code.

6.2 Prelab Preparation

6.2.1 Objective

By the end of this lab, students should be able to:

1. Understand the unique properties and characteristics of special numbers, gaining a deep understanding of their mathematical significance and relevance in programming.

2. Learn how to identify and fix common programming errors that occur when working with special numbers.

6.2.2 Background

Number Manipulation is a fundamental concept in computer programming that involves the alteration and manipulation of numerical values to achieve specific outcomes or solve computational problems. It plays a crucial role in various programming tasks, from basic arithmetic operations to more complex mathematical algorithms. Here are some key aspects of number manipulation in programming

Arithmetic Operations:
Addition (+), Subtraction (-), Multiplication (*), Division (/), and Modulo (%): These fundamental arithmetic operations are used to manipulate numbers for various purposes, such as calculations, comparisons, and transformations.

Rounding and Truncation
Mathematical Functions: Programming languages provide functions for rounding numbers (e.g., Math.round()) or truncating decimals (e.g., Math.floor() and Math.ceil()). These functions are essential for ensuring precision and formatting in calculations.

Type Conversion Casting: Converting a number from one data type to another is known as casting. For instance, casting a floating-point number to an integer type involves removing the decimal portion. Explicit casting helps handle different data types during operations.

Random Number Generation

Pseudorandom Number Generators: Programming languages offer methods to generate random numbers within a specified range. This is useful for simulations, games, and cryptographic applications.

Number Decomposition

Extracting Digits: For tasks like checking special numbers (e.g., Armstrong, spy, and palindromic numbers), numbers can be decomposed into their individual digits, often using modulo and division operations.

Number Reversal

Reversing Digits: Reversing the order of digits in a number is important for tasks like checking palindromic numbers. This can be achieved through iterative operations.

Bitwise Operations (for binary numbers)

Bitwise AND (&), OR (—), XOR ($\hat{}$), Shifts (¡¡ and ¿¿), and Complement ( ): Bit manipulation is common when working with binary numbers or performing low-level operations. Bitwise operations are used in various scenarios, including optimizing code or solving specific problems related to data encoding.

Numerical Comparisons

Relational Operators (e.g., ¡, ¿, ¡=, ¿=, ==, and !=): Comparing numbers to determine relationships or validate conditions is a fundamental aspect of programming. These comparisons help make decisions based on numerical values.

Precision and Rounding Errors

Floating-Point Arithmetic Considerations: Due to limited precision in floating-point numbers, handling rounding errors and ensuring accuracy in calculations are vital in numerical programming.

Mathematical Functions and Libraries

Math Libraries: Many programming languages provide extensive math libraries that offer a wide range of mathematical functions for more advanced number manipulation, including trigonometric, exponential, and logarithmic functions.

Number manipulation is a versatile skill that programmers rely on when working with numerical data in a wide range of applications, from scientific simulations to financial calculations and game development. It forms the foundation of many algorithms and data processing tasks, making it an essential aspect of programming expertise.

To successfully complete this lab, students will need access to the following:

1. A computer with a suitable text editor or Integrated Development Environment (IDE) installed, such as Eclipse, IntelliJ IDEA, or Visual Studio Code.

2. Java Development Kit (JDK) installed on the computer. Ensure that the JDK is correctly configured, and the system's PATH variable is set to include the JDK's bin directory.

6.2.3 Hardware/Software Requirements

To successfully complete this lab, students will need access to the following:

1. A computer with a suitable text editor or Integrated Development Environment (IDE) installed, such as Eclipse, IntelliJ IDEA, or Visual Studio Code.

2. Java Development Kit (JDK) installed on the computer. Ensure that the JDK is correctly configured, and the system's PATH variable is set to include the JDK's bin directory.

6.2.4 Pre-Lab Questions

1. When would you use casting to manipulate numbers in programming?
2. How can you round a floating-point number to the nearest integer in code?
3. What does it mean to reverse the digits of a number, and why is it useful?
4. Explain the purpose of a random number generator in programming.

6.3 Experiment Procedure

1. Ensure you have access to a Java development environment on your computer. Open your Java IDE or text editor for coding.
2. Create a new Java class with a suitable name in your IDE or text editor.
3. Write Java code to solve the problem
4. Test your code with different input values to ensure it behaves as expected. Debug and fix any issues that arise during testing.
5. Document the problem statement, the approach used to solve it, and the code you've written in your lab notebook or report.
6. Experiment with variations of the problem, explore different scenarios, and modify your code to handle various cases.

6.4 Experiemnts

6.4.1 AmicableNumbers

Aim: Two different numbers are said to be so amicable numbers if each sum of divisors is equal to the other number. Amicable Numbers are: (220, 284), (1184, 1210), (2620, 2924), (5020, 5564), (6232, 6368).

Program

```
/**
* author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/

import java.util.Scanner;

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

        Scanner in = new Scanner(System.in);

        System.out.print("Enter first number: ");
        int number1 = in.nextInt();
```

```
        System.out.print("Enter second number: ");
        int number2 = in.nextInt();

        int sum1 = 1;

        for (int i = 2; i <= number1/2; i++) {
            if (number1 % i == 0) {
                sum1 = sum1 + i;
            }
        }

        int sum2 = 1;
        for (int i = 2; i <= number2/2; i++) {
            if (number2 % i == 0) {
                sum2 = sum2 + i;
            }
        }

        if((sum1 == number2) && (sum2 == number1)){
            System.out.println(number1 + " and " + number2 + " are Amicable
                Numbers");
        }
        else{
            System.out.println(number1 + " and " + number2 + " are not
               Amicable Numbers");
        }
        in.close();
    }
}
```

Result Enter first number: 220
Enter second number: 284
220 and 284 are Amicable Numbers

6.4.2 ArmstrongNumber

Aim: Armstrong number is a positive number if it is equal to the sum of cubes of its digits is called Armstrong number and if its sum is not equal to the number then it's not a Armstrong number.

Program

```
/**
* author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/

import java.util.Scanner;

public class ArmstrongNumber {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Enter a number: ");

        int number = in.nextInt();
```

```
        int originalNumber = number;
        int result = 0;

        int numberOfDigits = (int) Math.log10(number) + 1;

        while (number > 0) {
            int digit = number % 10;
            result += Math.pow(digit, numberOfDigits);
            number /= 10;
        }

        if (result == originalNumber) {
            System.out.println(originalNumber + " is an Armstrong number.")
                ;
        } else {
            System.out.println(originalNumber + " is not an Armstrong
                number.");
        }
        in.close();
    }
}
```

Result
Enter a number: 153
153 is an Armstrong number.

Enter a number: 150
150 is not an Armstrong number.

6.4.3 CapricornNumber

Aim: A number is called Capricorn or Kaprekar number whose square is divided into two parts in any conditions and parts are added, the additions of parts is equal to the number, is called Capricorn or Kaprekar number.

Program

```
/**
* author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/

import java.util.Scanner;

public class CapricornNumber {
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int n = scanner.nextInt();
        boolean isCapricorn = false;

        int square = n * n;
        int temp = square;
        int contDigits = 0;
```

```
        while (temp > 0)
        {
            contDigits++;
            temp /= 10;
        }

        for (int i = 1; i < contDigits; i++)
        {
            int divisor = (int) Math.pow(10, i);
            int quotient = square / divisor;
            int remainder = square % divisor;
            if (quotient + remainder == n)
            {
                isCapricorn = true;
            }
        }
        if (isCapricorn)
        {
            System.out.println(n + " is a Capricorn/Kaprekar number");
        } else
        {
            System.out.println(n + "is not a Capricorn/Kaprekar number");
        }
    }
}
```

Result
Enter a number: 45
45 is a Capricorn/Kaprekar number

Enter a number: 297
297 is a Capricorn/Kaprekar number

6.4.4 CircularPrime

Aim: A circular prime is a prime number with the property that the number generated at each intermediate step when cyclically permuting its digits will be prime. For example, 1193 is a circular prime, since 1931, 9311 and 3119 all are also prime.

Program

```
/**
* author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/

import java.util.Scanner;

public class CircularPrime {
    public static void main(String[] args)
    {
        boolean flag = true;
        int n, num, r, c = 0;

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

```
        System.out.print("Enter a number: ");
        n = sc.nextInt();

        num = n;
        while (num > 0)
        {
            r = num % 10;
            c++;
            num = num / 10;
        }
        num = n;
        for (int i = 1; i <= c; i++)
        {
            r = num % 10;
            num = num / 10;
            num = (r * (int) Math.pow(10, c - 1)) + num;
            if (!prime(num))
            {
                flag = false;
                break;
            }
        }

        if(flag)
        {
            System.out.println(n + " is a Circular Prime");
        }
        else
        {
            System.out.println(n + " is not a Circular Prime");
        }

    }
    static boolean prime(int n)
    {
        // TODO code application logic here
        int i = 2;
        boolean flag = true;
        while (n > i)
        {
            if (n % 2 == 0)
            {
                flag = false;
                break;
            }
            i++;
        }
        return flag;
    }
}
```

Result

Enter a number: 137
137 is a Circular Prime

Enter a number: 44
44 is not a Circular Prime

6.4.5 HappyNumber

Aim: A happy number is a natural number in a given number base that eventually reaches 1 when iterated over the perfect digital invariant function for. Those numbers that do not end in 1 are -unhappy numbers.

Program

```
/**
* author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/

import java.util.Scanner;

public class HappyNumber {
    public static void main(String[] args)
    {
        int n, r = 1, num, sum = 0;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number: ");
        n = sc.nextInt();
        num = n;
        while (num > 9)
        {
            while (num > 0)
            {
                r = num % 10;
                sum = sum + (r * r);
                num = num / 10;
            }
            num = sum;
            sum = 0;
        }
        if (num == 1)
        {
            System.out.println(n + " is a Happy Number");
        }
        else
        {
            System.out.println(n + " is not a Happy Number");
        }
    }
}
```

Result

Enter number: 31
31 is a Happy Number

Enter number: 32
32 is a Happy Number

Enter number: 33
33 is not a Happy Number

6.4.6 AutomorphicNumber

Aim: An Automorphic number is a number whose square "ends" in the same digits as the number itself.

Program

```
/**
* author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/

import java.util.Scanner;

public class AutomorphicNumber {
    public static void main(String[] args)
    {
        int n, sqrNum, temp,sqrNumRemainder,c = 0;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        n = sc.nextInt();

        // get the number of digits
        temp=n;
        while (temp > 0)
        {
            temp=temp/10;
            c++;
        }

        sqrNum = n * n;
        sqrNumRemainder= sqrNum%(int)Math.pow(10, c);

        if(sqrNumRemainder==n)
        {
            System.out.println(n + " is an Automorphic Number");
        }
        else
        {
            System.out.println(n + " is not an Automorphic Number");
        }
    }
}
```

Result

Enter a number: 5
5 is an Automorphic Number

Enter a number: 25
25 is an Automorphic Number

Enter a number: 6
6 is an Automorphic Number

Enter a number: 7
7 is not an Automorphic Number

6.4.7 DisariumNumber

Aim: A number is called Disarium number if the sum of its power of the positions from left to right is equal to the number.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
*/

import java.util.Scanner;

public class DisariumNumber {
    public static void main(String[] args)
    {
        int r, n, num,digits=0,
        sum = 0;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        n = sc.nextInt();
        num = n;
        while (num > 0)
        {
            digits++;
            num = num / 10;
        }
        num = n;
        while (num > 0)
        {
            r = num % 10;
            sum = sum + (int)Math.pow(r, digits);
            num = num / 10;
            digits--;
        }

        if(n==sum)
        {
            System.out.println(n + " is a Disarium Number");
        }
        else
        {
            System.out.println(n + " is not a Disarium Number");
        }
    }
}
```

Result

Enter a number: 135
135 is a Disarium Number

Enter a number: 130
130 is not a Disarium Number

6.4.8 Magic Number

Aim: Magic number is the if the sum of its digits recursively are calculated till a single digit If the single digit is 1 then the number is a magic number. Magic number is very similar with Happy Number.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/

import java.util.Scanner;

public class MagicNumbers {

    public static void main(String[] args) {
        int n, r = 1, num, sum = 0;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        n = sc.nextInt();
        num = n;
        while (num > 9)
        {
            while (num > 0)
            {
                r = num % 10;
                sum = sum + r;
                num = num / 10;
            }
            num = sum;
            sum = 0;
        }
        if (num == 1)
        {
            System.out.println(n + " is a Magic Number");
        }
        else
        {
            System.out.println(n + " is not a Magic Number");
        }
    }
}
```

Result

Enter a number: 541
541 is a Magic Number

Enter a number: 226
226 is a Magic Number

Enter a number: 150
150 is not a Magic Number

6.4.9 NeonNumber

Aim: A neon number is a number where the sum of digits of square of the number is equal to the number. For example if the input number is 9, its square is 9*9 = 81 and sum of the digits is 9. i.e. 9 is a neon number.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/

import java.util.Scanner;

public class NeonNumber {
    public static void main(String[] args)
    {
        // TODO code application logic here
        int n,
        sqr = 1,
        sum = 0, r;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        n = sc.nextInt();
        sqr = n * n;
        while (sqr > 0)
        {
            r = sqr % 10;
            sum = sum + r;
            sqr = sqr / 10;
        }
        if (n == sum)
        {
            System.out.println(n + " is a Neon Number");
        }
        else
        {
            System.out.println(n + " is not a Neon Number");
        }
    }
}
```

Result
Enter a number: 9
9 is a Neon Number

Enter a number: 8
8 is not a Neon Number

6.4.10 PalindromicNumber

Aim: A palindromic number is a number that remains the same when its digits are reversed.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/

import java.util.Scanner;

public class PalindromicNumber {
    public static void main(String[] args)
    {
        int n, num, r,
        rev = 0;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        n = sc.nextInt();
        num = n;
        while (num > 0)
        {
            r = num % 10;
            rev = (rev * 10) + r;
            num = num / 10;
        }
        if (n == rev)
        {
            System.out.println(n + " is a Palindrome Number");
        }
        else
        {
            System.out.println(n + " is not a Palindrome Number");
        }
    }
}
```

Result
Enter a number: 245
245 is not a Palindrome Number

Enter a number: 242
242 is a Palindrome Number

6.4.11 PerfectNumber

Aim:A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. For instance, 6 has divisors 1, 2 and 3, and $1 + 2 + 3 = 6$, so 6 is a perfect number.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/

import java.util.Scanner;
```

```
public class PerfectNumber {
    public static void main(String[] args)
    {
        int n,sum = 0;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        n = sc.nextInt();
        for (int i = 1; i < n; i++)
        {
            if (n % i == 0)
            {
                sum = sum + i;
            }
        }
        if (n== sum)
        {
            System.out.println(n + " is a Perfect Number");
        }
        else
        {
            System.out.println(n + "is not a Perfect Number");
        }
    }
}
```

Result
Enter a number: 6
6 is a Perfect Number

Enter a number: 5
5 is not a Perfect Number

6.4.12 SpecialNumber

Aim: A number is said to be special number when the sum of factorial of its digits is equal to the number itself. Example- 145 is a Special Number as 1!+4!+5!=145.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/

import java.util.Scanner;

public class SpecialNumber {
    public static void main(String[] args)
    {
        int n, num, r,
        sumOfFactorial = 0;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        n = sc.nextInt();
        num = n;
        while (num > 0)
```

```
        {
            r = num % 10;
            int fact=1;
            for(int i=1;i<=r;i++)
            {
                fact=fact*i;
            }
            sumOfFactorial = sumOfFactorial+fact;
            num = num / 10;
        }
        if(n==sumOfFactorial)
        {
            System.out.println(n + " is a Special Number" );
        }
        else
        {
            System.out.println(n + "is not a Special Number" );
        }
    }
}
```

Result
Enter a number: 145
145 is a Special Number

Enter a number: 23
23 is not a Special Number

6.4.13 SpyNumber

Aim: A spy number is a number where the sum of its digits equals the product of its digits. For example, 1124 is a spy number, the sum of its digits is 1+1+2+4=8 and the product of its digits is 1*1*2*4=8.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/

import java.util.Scanner;

public class SpyNumber {
    public static void main(String[] args)
    {
        int r, n, num, mul = 1, sum = 0;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        n = sc.nextInt();
        num = n;
        while (num > 0)
        {
            r = num % 10;
            sum = sum + r;
            mul = mul * r;
```

```
            num = num / 10;
        }
        if (mul == sum)
        {
            System.out.println(n + " is a Spy Number");
        }
        else
        {
            System.out.println(n + " is not a Spy Number");
        }
    }
}
```

Result
Enter a number: 1124
1124 is a Spy Number

Enter a number: 12
12 is not a Spy Number

6.4.14 UglyNumber

Aim: A number is said to be an Ugly number if positive numbers whose prime factors only include 2, 3, 5. For example, 6(2×3), 8(2x2x2), 15(3×5) are ugly numbers while 14(2×7) is not ugly since it includes another prime factor 7. Note that 1 is typically treated as an ugly number.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/

import java.util.Scanner;

public class UglyNumber {
    public static void main(String[] args)
    {
        int n;
        boolean flag=true;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        n = sc.nextInt();
        int temp = n;
        while (n != 1)
        {
            if (n % 5 == 0)
            {
                n /= 5;
            }
            else if (n % 3 == 0)
            {
                n /= 3;
            }
            else if (n % 2 == 0)
            {
```

```
                n /= 2;
            }
            else
            {
                flag=false;
                break;
            }
        }
        if (flag)
        {
            System.out.println(temp + " is an Ugly number.");
        }
        else
        {
            System.out.println(temp + " is not an Ugly number.");
        }
        sc.close();
    }
}
```

Result
Enter a number: 6
6 is an Ugly number.

Enter a number: 14
14 is not an Ugly number.

6.5 Sample Viva Voce Questions

1. **Question:**

   Answer:

2. **Question:**

   Answer:

3. **Question:**

   Answer:

4. **Question:**

   Answer:

6.6 Further Probing Experiments

1. The least common multiple, lowest common multiple, or smallest common multiple of two integers a and b, usually denoted by LCM(a, b), is the smallest positive integer that is divisible by both a and b.

   Write a Java Program to find the LCM of two given numbers

2. Two integers a and b are said to be relatively prime, mutually prime, or coprime if the only positive integer that divides both of them is 1. Example: 13 and 15 are co prime.

   Write a Java program to check whether the two given numbers are co-prime or not

EXERCISES ON STRING AND CHAR OPERATIONS

7.1 Introduction

This lab, focus on the fundamental concepts of strings and characters in the context of Java programming. Strings are an integral part of any programming language, and understanding how to work with them is crucial for developing applications that handle textual data efficiently. In Java, strings and characters offer a rich set of features and functions, making them powerful tools for developers. This lab will provide you with a comprehensive understanding of strings and characters, from basic operations to more advanced techniques. By the end of this lab, you will be well-equipped to manipulate and work with text in Java, a skill that is vital for a wide range of software development tasks.

7.2 Prelab Preparation

7.2.1 Objective

By the end of this lab, students should be able to:

1. Understand the basics of strings in Java, including string declaration, initialization, and essential string operations like concatenation and comparison.
2. Perform various operations on strings, such as modifying, extracting substrings, and transforming text to meet specific requirements.
3. Work with individual characters, including character representation, comparison, and manipulation.

7.2.2 Background

Strings and characters are integral components of programming and are particularly significant in Java. Understanding how to work with them is fundamental for developing Java applications.

User Interaction: Strings are frequently used to process and manipulate user input. Understanding how to work with strings is crucial for building interactive applications where users provide textual information.

Data Storage: Strings are used to store and represent data, both within the program and in external storage, such as databases and files. Effective string handling is essential for data management.

Text Processing: Many applications involve text processing, from simple search and replace operations to more complex data extraction and transformation tasks. Mastery of strings and characters is essential for these operations.

Output Formatting: Properly formatted strings are crucial for presenting information to users or generating reports. The ability to format strings according to specific requirements is a valuable skill.

When working with strings and characters in Java, there are several fundamental concepts and techniques that are essential for efficient text processing. These concepts form the core knowledge required for effective manipulation of strings and characters in Java. Here are some of the required concepts:

String Declaration and Initialization: Understanding how to declare and initialize strings is fundamental. In Java, strings can be declared using the String class, and they can be initialized using string literals or by creating new instances.

In Java, you can declare and initialize strings in several ways. Here are some common methods for declaring and initializing strings:

- Using String Literals:

  You can directly create a string using double-quoted string literals.

  ```
  String greeting = "Hello, World!";
  ```

- Using the new Keyword:

  You can create a string object using the new keyword.

  ```
  String message = new String("This is a new string.");
  ```

- Using Empty String:

  You can create an empty string by assigning an empty string literal to a variable.

  ```
  String emptyString = "";
  ```

- Null String:

  You can also create a string with a null value.

  ```
  String nullString = null;
  ```

String Concatenation: Concatenating (joining) strings together is a common operation. The + operator is often used for string concatenation, and the concat() method of the String class can also be used.

String Length: Finding the length of a string is essential for various operations. You can use the length() method to determine the number of characters in a string.

String Comparison: You'll often need to compare strings for equality. The equals() method is used to compare the content of two strings, while == is used to compare their references.

String Manipulation: Learn how to manipulate strings by using methods like substring(), charAt(), replace(), toLowerCase(), toUpperCase(), and many more. These methods enable you to modify and extract parts of strings.

String Parsing: Parsing involves breaking down a string into smaller components, such as tokens. The split() method and regular expressions are commonly used for string parsing.

String Formatting: Properly formatting strings is crucial for displaying information to users. You can use methods like String.format() or printf-style formatting to format strings according to specific requirements.

Character Handling: Understand the basics of working with individual characters in strings. This includes accessing characters by index, comparing characters, and converting between characters and their integer representations.

Escape Sequences: Learn common escape sequences used to represent special characters within strings, such as \n for a newline or \t for a tab.

7.2.3 Hardware/Software Requirements

To successfully complete this lab, students will need access to the following:

1. A computer with a suitable text editor or Integrated Development Environment (IDE) installed, such as Eclipse, IntelliJ IDEA, or Visual Studio Code.
2. Java Development Kit (JDK) installed on the computer. Ensure that the JDK is correctly configured, and the system's PATH variable is set to include the JDK's bin directory.

7.2.4 Pre-Lab Questions

1. Why are strings important for user interaction in Java applications?
2. How do you declare and initialize a string using string literals in Java?
3. Explain the difference between using string literals and the new keyword to create a string.
4. What method is used to find the length of a string in Java, and why is it important?

7.3 Experiment Procedure

1. Ensure you have access to a Java development environment on your computer. Open your Java IDE or text editor for coding.
2. Create a new Java class with a suitable name in your IDE or text editor.
3. Write Java code to solve the problem
4. Test your code with different input values to ensure it behaves as expected. Debug and fix any issues that arise during testing.
5. Document the problem statement, the approach used to solve it, and the code you've written in your lab notebook or report.
6. Experiment with variations of the problem, explore different scenarios, and modify your code to handle various cases.

7.4 Experiments, Results & Discussion

7.4.1 ReverseString (String & char)

Aim: Write a program called ReverseString, which prompts user for a String, and prints the reverse of the String by extracting and processing each character.

Program

```
/**
*author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/

import java.util.Scanner;

public class ReverseString {
    public static void main(String args[]){
        // Define variables
        String inStr;        // input String
        int inStrLen;        // length of the input String

        Scanner in = new Scanner(System.in);
        // Prompt and read input as "String"
        System.out.print("Enter a String: ");
        inStr = in.next();   // use next() to read a String
        inStrLen = inStr.length();

        // Use inStr.charAt(index) in a loop to extract each character
        // The String's index begins at 0 from the left.
        // Process the String from the right
        System.out.print("The reverse of the String " + inStr + " is ");
        for (int charIdx = inStrLen - 1; charIdx >= 0; --charIdx) {
            // charIdx = inStrLen-1, inStrLen-2, ... ,0
            System.out.print(inStr.charAt(charIdx));
        }

    }
}
```

Result
Enter a String: abcdefghijk
The reverse of the String abcdefghijk is kjihgfedcba

7.4.2 CountVowelsDigits (String & char)

Aim: Write a program called CountVowelsDigits, which prompts the user for a String, counts the number of vowels (a, e, i, o, u, A, E, I, O, U) and digits (0-9) contained in the string, and prints the counts and the percentages (rounded to 2 decimal places).

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
```

```
* IARE, Dundigal.
* @author cvrpa
*/

import java.util.Scanner;

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

        // Prompt the user for input
        System.out.print("Enter a string: ");
        String input = in.nextLine();

        int vCount = 0;
        int dCount = 0;
        // Count vowels
        String vowels = "aeiouAEIOU";
        for (int i = 0; i < input.length(); i++) {
            char c = input.charAt(i);
            if (vowels.indexOf(c) != -1) {
                vCount++;
            }
            if (Character.isDigit(c)) {
                dCount++;
            }
        }

        // Calculate percentages
        int totalCharacters = input.length();
        double vowelPercentage = (double) vCount / totalCharacters * 100;
        double digitPercentage = (double) dCount / totalCharacters * 100;

        // Print the results
        System.out.printf("Number of vowels: %d (%.2f%%)\n",vCount,
           vowelPercentage);
        System.out.printf("Number of digits: %d (%.2f%%)\n", dCount,
           digitPercentage);

        // Close the scanner
        in.close();
    }
}
```

Result
Enter a string: testing123456
Number of vowels: 2 (15.38%)
Number of digits: 6 (46.15%)

7.4.3 PhoneKeyPad (String & char)

Aim: On you phone keypad, the alphabets are mapped to digits as follows:

ABC(2), DEF(3), GHI(4), JKL(5), MNO(6), PQRS(7), TUV(8), WXYZ(9).

Write a program called PhoneKeyPad, which prompts user for a String (case insensitive),

and converts to a sequence of keypad digits.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept CSE
* IARE, Dundigal.
* @author cvrpa
*/

import java.util.Scanner;

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

        // Prompt the user for input
        System.out.print("Enter a string: ");
        String input = in.nextLine().toUpperCase(); // Convert to uppercase
             for case insensitivity

        // Convert the input string to keypad digits
        StringBuilder result = new StringBuilder();
        System.out.print("The sequence of keypad digits: ");
        for (int i = 0; i < input.length(); i++) {
            char c = input.charAt(i);
            switch(c){
                case 'A': case 'B': case 'C':
                System.out.print(2); break;
                case 'D': case 'E': case 'F':
                System.out.print(3); break;
                case 'G': case 'H': case 'I':
                System.out.print(4); break;
                case 'J': case 'K': case 'L':
                System.out.print(5); break;
                case 'M': case 'N': case 'O':
                System.out.print(6); break;
                case 'P': case 'Q': case 'R': case 'S':
                System.out.print(7); break;
                case 'T': case 'U': case 'V':
                System.out.print(8); break;
                case 'W': case 'X': case 'Y': case 'Z':
                System.out.print(9); break;
                default:
                System.out.print("");
            }
        }
        // Close the scanner
        in.close();
    }
}
```

Result

Enter a string: abcdxyhjs
The sequence of keypad digits: 222399457

7.4.4 Caesar's Code (String & char)

Aim: Caesar's Code is one of the simplest encryption techniques. Each letter in the plaintext is replaced by a letter some fixed number of position (n) down the alphabet cyclically. In this exercise, we shall pick n=3. That is, 'A' is replaced by 'D', 'B' by 'E', 'C' by 'F', ..., 'X' by 'A', ..., 'Z' by 'C'.

Write a program called CaesarCode to cipher the Caesar's code. The program shall prompt user for a plaintext string consisting of mix-case letters only; compute the ciphertext; and print the ciphertext in uppercase.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/

import java.util.Scanner;

public class CaesarsCode {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a plaintext string: ");

        String plaintext = sc.nextLine();
        plaintext = plaintext.toUpperCase();

        String ciphertext = " ";

        for (int i = 0; i < plaintext.length(); i++) {
            char originalChar = plaintext.charAt(i);

            if (Character.isLetter(originalChar)) {
                char shiftedChar = (char) (originalChar + 3);
                ciphertext += shiftedChar;
            } else {
                // If the character is not a letter, add it unchanged to
                   the ciphertext
                ciphertext += originalChar;
            }
        }
        System.out.println("The ciphertext string is:" + ciphertext);
        sc.close();
    }
}
```

Result

Enter a plaintext string: Testing
The ciphertext string is: WHVWLQJ

7.4.5 Decipher Caesar's Code (String & char)

Aim: Write a program called DecipherCaesarCode to decipher the Caesar's code described in the previous exercise. The program shall prompts user for a ciphertext string consisting of mix-case letters only; compute the plaintext; and print the plaintext in uppercase.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/

import java.util.Scanner;

public class DecipherCaesarCode {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a plaintext string: ");

        String plaintext = sc.nextLine();
        plaintext = plaintext.toUpperCase();

        String ciphertext = " ";

        for (int i = 0; i < plaintext.length(); i++) {
            char originalChar = plaintext.charAt(i);

            if (Character.isLetter(originalChar)) {
                char shiftedChar = (char) (originalChar - 3);
                ciphertext += shiftedChar;
            } else {
                // If the character is not a letter, add it unchanged to
                   the ciphertext
                ciphertext += originalChar;
            }
        }
        System.out.println("The ciphertext string is:" + ciphertext);
        sc.close();
    }
}
```

Result
Enter a plaintext string: wHVwLQJThe ciphertext string is: TESTING

7.4.6 Exchange Cipher (String & char)

Aim: This simple cipher exchanges 'A' and 'Z', 'B' and 'Y', 'C' and 'X', and so on. Write a program called ExchangeCipher that prompts user for a plaintext string consisting of mix-case letters only. You program shall compute the ciphertext; and print the ciphertext in uppercase.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/

import java.util.Scanner;

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

```
        System.out.print("Enter a plaintext string: ");

        String plaintext = sc.nextLine();
        plaintext = plaintext.toUpperCase();

        String ciphertext = " ";

        for (int i = 0; i < plaintext.length(); i++) {
            char originalChar = plaintext.charAt(i);

            if (Character.isLetter(originalChar)) {
                char shiftedChar = (char) ('A' + 'Z' - originalChar);
                ciphertext += shiftedChar;
            } else {
                // If the character is not a letter, add it unchanged to
                   the ciphertext
                ciphertext += originalChar;
            }
        }
        System.out.println("The ciphertext string is:" + ciphertext);
        sc.close();
    }
}
```

Result
Enter a plaintext string: abcXYZ
The ciphertext string is: ZYXCBA

7.4.7 TestPalindromicWord (String & char)

Aim: A word that reads the same backward as forward is called a palindrome, e.g., "mom", "dad", "racecar", "madam", and "Radar" (case-insensitive).

Write a program called TestPalindromicWord, that prompts user for a word and prints ""xxx" is—is not a palindrome".

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/

import java.util.Scanner;

public class TestPalindromicPhrase {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a phrase: ");
        String phrase = scanner.nextLine();
        scanner.close();
        boolean flag = true;

        phrase = phrase.replaceAll("[^a-zA-Z]", "").toLowerCase();
        int length = phrase.length();
        int fIdx = 0, bIdx = length - 1;
```

```
        while (fIdx < bIdx) {
            char forwardChar = phrase.charAt(fIdx);
            char backwardChar = phrase.charAt(bIdx);

            if (forwardChar != backwardChar) {
                flag = false;
                break;
            }
            ++fIdx;
            --bIdx;
        }

        if (flag) {
            System.out.println("The phrase is a palindromic phrase.");
        } else {
            System.out.println("The phrase is not a palindromic phrase.");
        }
    }
}
```

Result
Enter a word: maDaM
maDaM is a palindromic word.

7.4.8 CheckBinStr (String & char)

Aim: The binary number system uses 2 symbols, 0 and 1. Write a program called CheckBinStr to verify a binary string. The program shall prompt user for a binary string; and decide if the input string is a valid binary string.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/

import java.util.Scanner;

public class CheckBinStr {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a binary string: ");
        String binaryStr = sc.nextLine();
        boolean flag = true;

        for (char c : binaryStr.toCharArray()) {
            if (c != '0' && c != '1') {
                flag = false;
                break;
            }
        }

        if (flag) {
            System.out.println("\"" + binaryStr + "\" is a binary string");
        } else {
```

```
            System.out.println("\"" + binaryStr + "\" is NOT a binary
                string");
        }
        sc.close();
    }
}
```

Result
Enter a binary string: 10101100
”10101100” is a binary string

Enter a binary string: 10201011
”10201011” is NOT a binary string

7.4.9 CheckHexStr (String & char)

Aim: The hexadecimal (hex) number system uses 16 symbols, 0-9 and A-F (or a-f). Write a program to verify a hex string. The program shall prompt user for a hex string; and decide if the input string is a valid hex string.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/

import java.util.Scanner;

public class CheckHexStr {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a hex string: ");
        String hexStr = sc.nextLine();
        boolean flag = true;

        for (char c : hexStr.toCharArray()) {
            if (!Character.isDigit(c) && !((c >= 'A' && c <= 'F') || (c >=
                'a' && c <= 'f'))) {
                flag = false;
            }
        }

        if (flag) {
            System.out.println("\"" + hexStr + "\" is a hex string");
        } else {
            System.out.println("\"" + hexStr + "\" is NOT a hex string");
        }

        sc.close();
    }
}
```

Result
Enter a Hexadecimal string: 123aBc
The equivalent binary for hexadecimal ”123aBc” is: 0001 0010 0011 1010 1011 1100

Enter a Hexadecimal string: 123aBc
The equivalent binary for hexadecimal "123aBc" is: 0001 0010 0011 1010 1011 1100

7.4.10 Bin2Dec (String & char)

Aim: Write a program called Bin2Dec to convert an input binary string into its equivalent decimal number.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/

import java.util.Scanner;

public class Bin2Dec {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a Binary string: ");
        String binaryStr = scanner.nextLine();
        boolean flag = true;

        int decimal = 0;
        int binaryLength = binaryStr.length();
        int power = 0;

        // Iterate through the binary string from right to left
        for (int i = binaryLength - 1; i >= 0; i--) {
            char digit = binaryStr.charAt(i);

            // Check if the character is '0' or '1'
            if (digit == '0' || digit == '1') {
                int bitValue = Character.getNumericValue(digit);
                decimal += bitValue * Math.pow(2, power);
                power++;
            } else {
                flag = false;
                break;
            }
        }

        if(flag){
            System.out.println("The equivalent decimal number for binary
               \"" + binaryStr + "\" is: " + decimal);
        }
        else{
            System.out.println("error: invalid binary string \"" +
               binaryStr + "\"");
        }
        scanner.close();
    }
}
```

Result
Enter a Binary string: 1011
The equivalent decimal number for binary "1011" is: 11

Enter a Binary string: 1234
error: invalid binary string "1234"

7.4.11 Hex2Dec (String & char)

Aim: Write a program called Hex2Dec to convert an input hexadecimal string into its equivalent decimal number.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/

import java.util.Scanner;

public class Hex2Dec {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a Hexadecimal string: ");
        String hexStr = scanner.nextLine();

        int decimal = 0;

        // Convert the hex string to uppercase for consistency
        hexStr = hexStr.toUpperCase();

        for (int i = 0; i < hexStr.length(); i++) {
            char digit = hexStr.charAt(i);

            if (Character.isDigit(digit)) {
                decimal = decimal * 16 + (digit - '0');
            } else if (digit >= 'A' && digit <= 'F') {
                decimal = decimal * 16 + (digit - 'A' + 10);
            } else {
                // Invalid hexadecimal digit found
                decimal = -1;
                break;
            }
        }

        if (decimal != -1) {
            System.out.println("The equivalent decimal number for
               hexadecimal \"" + hexStr + "\" is: " + decimal);
        } else {
            System.out.println("error: invalid hexadecimal string \"" +
               hexStr + "\"");
        }
        scanner.close();
    }
}
```

Result Enter a Hexadecimal string: 1a
The equivalent decimal number for hexadecimal "1A" is: 26

Enter a Hexadecimal string: abcd
The equivalent decimal number for hexadecimal "ABCD" is: 43981

Enter a Hexadecimal string: 1g5
error: invalid hexadecimal string "1G5"

7.4.12 Oct2Dec (String & char)

Aim: Write a program called Oct2Dec to convert an input Octal string into its equivalent decimal number.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/

import java.util.Scanner;

public class Oct2Dec {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter an Octal string: ");
        String octalStr = scanner.nextLine();

        int decimal = 0;

        for (int i = 0; i < octalStr.length(); i++) {
            char digit = octalStr.charAt(i);

            if (digit >= '0' && digit <= '7') {
                decimal = decimal * 8 + (digit - '0');
            } else {
                // Invalid octal digit found
                decimal = -1;
                break;
            }
        }

        if (decimal != -1) {
            System.out.println("The equivalent decimal number \"" +
               octalStr + "\" is: " + decimal);
        } else {
            System.out.println("error: Invalid Octal string \"" + octalStr
               + "\"");
        }
        scanner.close();
    }
}
```

Result

Enter an Octal string: 234
The equivalent decimal number "234" is: 156

Enter an Octal string: 148
error: Invalid Octal string "148"

7.5 Sample Viva Voce Questions

1. Give an example of a string concatenation operation in Java.
2. Discuss the concept of character manipulation in strings and provide an example.
3. How do you compare two strings for equality in Java, and what's the distinction between equals() and == for strings?
4. How can you extract a substring from a given string, and what method is used for this purpose?

7.6 Further Probing Experiments

1. A phrase that reads the same backward as forward is also called a palindrome, e.g., "Madam, I'm Adam", "A man, a plan, a canal - Panama!" (ignoring punctuation and capitalization).

   Write a program to check for palindromic phrase.

2. Write a Java program that takes a sentence as input, splits it into individual words using spaces as separators, and then displays both the original sentence and the total word count. For example, if the user enters 'Hello, World!,' the program should output 'Original Sentence: Hello, World! Total Words: 2.' Your program should handle typical sentence structures and spaces as separators.

EXERCISES ON ARRAYS

8.1 Introduction

Arrays are fundamental data structures that play a pivotal role in Java and many other programming languages. Arrays allow us to store and manipulate collections of data, making them a crucial component in a programmer's toolkit. In this lab, we will delve into the world of arrays in Java, exploring the concepts of array declaration, initialization, and manipulation. We will learn about the various types of arrays, multidimensional arrays, and how to work with them effectively. Arrays are the building blocks of many advanced data structures and algorithms, making this lab essential for your journey as a Java programmer.

8.2 Prelab Preparation

8.2.1 Objective

By the end of this lab, students should be able to:

1. Demonstrate a clear understanding of array basics, including declaration, initialization, and indexing in Java.
2. Effectively work with different array types, such as one-dimensional and multidimensional arrays, for data organization.
3. Perform array manipulation tasks, including adding, modifying, and deleting elements, as well as implementing common array operations like sorting and searching.
4. Apply their knowledge of arrays to solve practical programming problems, enhancing their problem-solving skills.

8.2.2 Background

Arrays are fundamental data structures in programming, and in Java, they play a critical role in organizing and managing data. Arrays allow us to store collections of values of the same data type, providing an efficient means to access and manipulate this data.

Declaration and Initialization:

In Java, arrays are declared by specifying the data type of the elements they will hold, followed by square brackets ([]) to denote that it's an array. For example, to declare an array of integers: int[] myArray;. Arrays can be initialized with values, either during declaration or separately.

```
int[] numbers = {1, 2, 3, 4, 5};
```

Indexing:

Arrays are zero-indexed, which means the first element is accessed at index 0, the second at index 1, and so on. Proper indexing is essential for working with arrays.

```
int firstNumber = numbers[0]; // Access the first element.
```

Array Length:

To find the number of elements in an array, you can use the length attribute.

```
int arrayLength = numbers.length; // Gets the length of the array.
```

One-Dimensional and Multidimensional Arrays:

Java supports both one-dimensional arrays (vectors) and multidimensional arrays (matrices). Multidimensional arrays can have multiple rows and columns.

```
int[][] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
```

Array Manipulation:

Arrays can be manipulated by adding, modifying, or removing elements. Common array operations include sorting and searching for specific values within an array.

```
int[] extendedArray = Arrays.copyOf(numbers, numbers.length + 1);
extendedArray[numbers.length] = 6;
```

Arrays are the foundation for many advanced data structures and algorithms. They provide a simple yet powerful way to manage large amounts of data, making them essential for tasks such as data storage, sorting, searching, and more. Understanding arrays is crucial for any Java programmer, as it opens the door to efficient data management and problem-solving in a wide range of software development projects

8.2.3 Hardware/Software Requirements

To successfully complete this lab, students will need access to the following:

1. A computer with a suitable text editor or Integrated Development Environment (IDE) installed, such as Eclipse, IntelliJ IDEA, or Visual Studio Code.
2. Java Development Kit (JDK) installed on the computer. Ensure that the JDK is correctly configured, and the system's PATH variable is set to include the JDK's bin directory.

8.2.4 Pre-Lab Questions

1. What is an array in Java, and why are they important in programming?
2. How do you declare and initialize a one-dimensional array in Java?
3. How do you access elements in an array?
4. What is the role of the length attribute in arrays, and how do you use it?

8.3 Experiment Procedure

1. Ensure you have access to a Java development environment on your computer. Open your Java IDE or text editor for coding.
2. Create a new Java class with a suitable name in your IDE or text editor.

3. Write Java code to solve the problem
4. Test your code with different input values to ensure it behaves as expected. Debug and fix any issues that arise during testing.
5. Document the problem statement, the approach used to solve it, and the code you've written in your lab notebook or report.
6. Experiment with variations of the problem, explore different scenarios, and modify your code to handle various cases.

8.4 Experiments, Results & Discussion

8.4.1 PrintArray (Array)

Aim: Write a program called PrintArray which prompts user for the number of items in an array (a non-negative integer), and saves it in an int variable called NUM_ITEMS. It then prompts user for the values of all the items and saves them in an int array called items. The program shall then print the contents of the array in the form of [x1, x2, ..., xn].

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/

import java.util.Scanner;

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

        // Prompt the user for the number of items
        System.out.print("Enter the number of items: ");
        int NUM_ITEMS = scanner.nextInt();

        // Create an array to store the items
        int[] items = new int[NUM_ITEMS];

        // Prompt the user for the values of all items
        System.out.print("Enter the value of all items (separated by space)
            : ");
        for (int i = 0; i < NUM_ITEMS; i++) {
            items[i] = scanner.nextInt();
        }

        // Print the array's contents
        System.out.print("The values are: [");
        for (int i = 0; i < NUM_ITEMS; i++) {
            System.out.print(items[i]);
            if (i < NUM_ITEMS - 1) {
                System.out.print(", ");
            }
        }
        System.out.println("]");
        scanner.close();
```

```
    }
}
```

Result
Enter the number of items: 5
Enter the value of all items (separated by space): 23 45 34 67 58
The values are: [23, 45, 34, 67, 58]

8.4.2 PrintArrayInStars (Array)

Aim: Write a program called PrintArrayInStars which prompts user for the number of items in an array (a non-negative integer), and saves it in an int variable called NUM_ITEMS. It then prompts user for the values of all the items (non-negative integers) and saves them in an int array called items. The program shall then print the contents of the array in a graphical form, with the array index and values represented by number of stars.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/
import java.util.Scanner;

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

        // Prompt the user for the number of items
        System.out.print("Enter the number of items: ");
        int NUM_ITEMS = scanner.nextInt();

        // Create an array to store the items
        int[] items = new int[NUM_ITEMS];

        // Prompt the user for the values of all items
        System.out.print("Enter the value of all items (separated by space)
            : ");
        for (int i = 0; i < NUM_ITEMS; i++) {
            items[i] = scanner.nextInt();
        }

        // Print the array's contents using stars
        for (int i = 0; i < NUM_ITEMS; i++) {
            System.out.print(i + ": ");
            for (int j = 0; j < items[i]; j++) {
                System.out.print("*");
            }
            System.out.print("(" + items[i] + ")\n");
        }
        scanner.close();
    }
}
```

Result

Enter the number of items: 5
Enter the value of all items (separated by space): 3 5 2 0 6
0: ***(3)
1: *****(5)
2: **(2)
3: (0)
4: ******(6)

8.4.3 GradesStatistics (Array)

Aim: Write a program which prompts user for the number of students in a class (a non-negative integer), and saves it in an int variable called numStudents. It then prompts user for the grade of each of the students (integer between 0 to 100) and saves them in an int array called grades. The program shall then compute and print the average (in double rounded to 2 decimal places) and minimum/maximum (in int).

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/
import java.util.Scanner;

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

        // Prompt the user for the number of students
        System.out.print("Enter the number of students: ");
        int numStudents = scanner.nextInt();

        // Create an array to store the grades
        int[] grades = new int[numStudents];

        // Prompt the user for the grades of each student
        for (int i = 0; i < numStudents; i++) {
            System.out.print("Enter the grade for student " + (i + 1) + ":
                ");
            grades[i] = scanner.nextInt();
        }

        // Calculate the average, minimum, and maximum grades
        double sum = 0;
        int minimum = Integer.MAX_VALUE;
        int maximum = Integer.MIN_VALUE;

        for (int grade : grades) {
            sum += grade;
            if (grade < minimum) {
                minimum = grade;
            }
            if (grade > maximum) {
                maximum = grade;
```

```
            }
        }

        double average = sum / numStudents;

        // Print the results
        System.out.printf("The average is: %.2f\n", average);
        System.out.println("The minimum is: " + minimum);
        System.out.println("The maximum is: " + maximum);

        scanner.close();
    }
}
```

Result
Enter the number of students: 5
Enter the grade for student 1: 56
Enter the grade for student 2: 67
Enter the grade for student 3: 78
Enter the grade for student 4: 59
Enter the grade for student 5: 46
The average is: 61.20
The minimum is: 46
The maximum is: 78

8.4.4 Hex2Bin (Array for Table Lookup)

Aim: Write a program called Hex2Bin that prompts user for a hexadecimal string and print its equivalent binary string.

Hints 1. Use an array of 16 Strings containing binary strings corresponding to hexadecimal number 0-9A-F (or a-f), as follows

```
    final String[] HEX\_BITS = {
        "0000", "0001", "0010", "0011",
        "0100", "0101", "0110", "0111",
        "1000", "1001", "1010", "1011",
        "1100", "1101", "1110", "1111"};
```

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/
import java.util.Scanner;

public class Hex2Bin {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        final String[] HEX_BITS = {
            "0000", "0001", "0010", "0011",
            "0100", "0101", "0110", "0111",
            "1000", "1001", "1010", "1011",
            "1100", "1101", "1110", "1111"
```

```
        };

        System.out.print("Enter a Hexadecimal string: ");
        String  temp= scanner.nextLine(); // Convert to uppercase for
            consistency
        String hexString = temp.toUpperCase();
        StringBuilder binaryStringBuilder = new StringBuilder();

        for (char hexDigit : hexString.toCharArray()) {
            if (hexDigit >= '0' && hexDigit <= '9') {
                int index = hexDigit - '0'; // Convert from character
                    '0'-'9' to integer 0-9
                binaryStringBuilder.append(HEX_BITS[index]);
                binaryStringBuilder.append(" ");
            } else if (hexDigit >= 'A' && hexDigit <= 'F') {
                int index = hexDigit - 'A' + 10; // Convert from character
                    'A'-'F' to integer 10-15
                binaryStringBuilder.append(HEX_BITS[index]);
                binaryStringBuilder.append(" ");
            } else {
                System.out.println("Invalid hexadecimal digit: " + hexDigit
                    );
                scanner.close();
                return;
            }
        }

        String binaryString = binaryStringBuilder.toString();
        System.out.println("The equivalent binary for hexadecimal \"" +
            temp + "\" is: " + binaryString);
        scanner.close();
    }
}
```

Result
Enter a Hexadecimal string: 1abc
The equivalent binary for hexadecimal "1abc" is: 0001 1010 1011 1100

8.4.5 Dec2Hex (Array for Table Lookup)

Aim: Write a program called Dec2Hex that prompts user for a positive decimal number, read as int, and print its equivalent hexadecimal string.

Program

```
/**
* @author Dr.C V Rama Padmaja
* Associate Professor, Dept of CSE,
* IARE, Dundigal.
*/
import java.util.Scanner;

public class Dec2Hex {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a decimal number: ");
        int decimalNumber = scanner.nextInt();
```

```
        if (decimalNumber < 0) {
            System.out.println("Please enter a positive decimal number.");
        } else {
            String hexadecimalString = Integer.toHexString(decimalNumber).
                toUpperCase();
            System.out.println("The equivalent hexadecimal number is " +
                hexadecimalString);
        }
        scanner.close();
    }
}
```

Result
Enter a decimal number: 1234
The equivalent hexadecimal number is 4D2

Enter a decimal number: 120
The equivalent hexadecimal number is 78

8.5 Sample Viva Voce Questions

1. What is the significance of zero-based indexing in arrays?
2. Explain the importance of proper indexing and what happens when you access an out-of-bounds index.
3. How do you initialize a multidimensional array in Java, and what's the purpose of such arrays?
4. Can you have an array without specifying its size during declaration? If yes, how?

8.6 Further Probing Experiments

1.
2.
3.
4.