Big Data Frameworks CSE3120

Lab – 1 Java Programs Experiment

Name: Naveen Nidadavolu
Roll No: 22MIA1049

1. Display odd numbers between 1 -100

Aim: To write a Java program that displays all odd numbers between 1 and 100.

Algorithm

1. Start.
2. Set the range limit n = 100.
3. Use a for loop to iterate through numbers from 1 to n.
4. Inside the loop, check if a number is odd using the condition i % 2 != 0:
    - If true, print the number.
5. Stop.

Program:

```
class Main {
    public static void main(String[] args) {
        System.out.print("Displaying odd numbers between 1 -100:\n");
        for(int i = 1; i <= 100; i++) {
            if(i % 2 != 0) {
                System.out.print(i + "\t");
            }
        }
    }
}
```

Output

```
Displaying odd numbers between 1 -100:
1   3   5   7   9   11  13  15  17  19  21  23  25  27  29  31  33  35  37
    39  41  43  45  47  49  51  53  55  57  59  61  63  65  67  69  71  73
    75  77  79  81  83  85  87  89  91  93  95  97  99
=== Code Execution Successful ===
```

Result: The program successfully displays all odd numbers between 1 and 100.

2. Sum of odd numbers between 1 -100

Aim: To write a Java program that displays sum of odd numbers between 1 and 100.

Algorithm/Procedure

1. Start.

2. Initialize a variable sum = 0 to store the sum of odd numbers.
3. Loop through numbers from i = 1 to i = 100 using a for loop:
4. For each i, check if it is an odd number using i % 2 != 0.
5. If true (the number is odd), add i to sum.
6. After the loop ends, print the value of sum.
7. Stop.

Program

```java
//2. Sum of odd numbers between 1 -100
class Main {
    public static void main(String[] args) {
        int sum = 0;
        for(int i=1; i<=100;i++){
            if(i%2 != 0){
                sum = sum + i;
            }
        }
        System.out.print("Sum of the first 100 odd numbers  :"+sum);
    }
}
```

Output

```
Sum of the first 100 odd numbers  :2500
=== Code Execution Successful ===
```

Result: The sum of all odd numbers between 1 and 100 is 2500. The code execution is successful.

3. Program to check the given number is Palindrome or not

Aim: To write a Java program to check whether a given number is a palindrome or not.

Algorithm/Procedure

1. Start.
2. Import the Scanner class for user input.
3. Initialize two variables:
    - reversen = 0 (to store the reversed number).
    - remainder (to store each digit of the number).
4. Accept an integer input n from the user.
5. Store the original value of n in a new variable originalnum.
6. Use a while loop to reverse the number:
    - Extract the last digit using remainder = n % 10.
    - Append the digit to the reversed number using reversen = (reversen * 10) + remainder.
    - Remove the last digit from n using n = n / 10.
7. Compare originalnum and reversen:
    - If they are equal, print that the number is a palindrome.

- Otherwise, print that the number is not a palindrome.

8. Stop.

Program

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        int remainder, reversen = 0;
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the number:\n");
        int n = input.nextInt();

        int originalnum = n;
        while (n != 0) {
            remainder = n % 10;
            reversen = (reversen * 10) + remainder;
            n = n / 10;
        }

        System.out.println("The original number is " + originalnum);
        System.out.println("The reversed number is " + reversen);

        if (originalnum == reversen) {
            System.out.println("The given number is a palindrome");
        } else {
            System.out.println("The given number is not a palindrome");
        }
    }
}
```

Output

```
Enter the number:
121
The original number is 121
The reversed number is 121
The given number is a palindrome

=== Code Execution Successful ===
```

Result: The program successfully checks whether the given number is a palindrome or not.

Q4. Program to print patterns of numbers and stars

```
*
* *
* * *
* * * *
```

Aim: To write a Java program that prints a right-angled triangle pattern of stars.

Algorithm/Procedure

1. **Start.**
2. Declare an integer n for the number of rows.
3. Use two nested for loops:
    - The outer loop runs from i = 0 to i < n (controls the rows).
    - The inner loop runs from j = 0 to j <= i (controls the columns and prints stars).
4. In each iteration of the inner loop, print a star (*).
5. Move to the next line after completing a row.
6. **Stop.**

Program:

```
class Main{
    public static void main(String[] args){
        int n= 5;
        for(int i=0;i<n;i++){
            for(int j=0;j<=i;j++){
                System.out.print("*");
            }
            System.out.println("");
        }
    }
}
```

Output:

```
*
**
***
****
*****

=== Code Execution Successful ===
```

Result: The program successfully prints the right-angled triangle pattern of stars.

5. Print numbers in triangle and pyramid vice

1
121
12321
1234321
123454321

Aim: To write a Java program that prints a numerical pattern in a triangular and pyramid structure.

Algorithm

1. **Start**
2. Take an integer input n from the user (number of rows).
3. For each row i from 1 to n (outer loop):
    - Print numbers in **increasing order** from 1 to i (first inner loop).
    - Print numbers in **decreasing order** from i-1 back to 1 (second inner loop).

- Move to the next line after completing the row.

4. **Stop**

Program

```java
import java.util.Scanner;
class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the n value:\n");
        int n = input.nextInt();
        for(int i=1;i<=n;i++){
            for(int j=1;j<=i;j++){
                System.out.print(j);
            }
            for(int k=i;k>1;k--){
                System.out.print(k-1);
            }
            System.out.println();
        }
    }
}
```

Output:

```
Enter the n value:
5
1
121
12321
1234321
123454321

=== Code Execution Successful ===
```

Result: The program successfully prints the triangular pyramid structure with numbers as expected.

6. Find largest and smallest number in an array in java

Aim: To write a Java program that finds the largest and smallest numbers in a given array.

Algorithm

1. Start.
2. Take input for the size of the array n from the user.
3. Declare an integer array arr of size n.
4. Use a loop to input n elements into the array from the user.
5. Initialize two variables:
   - max to the smallest possible value (Integer.MIN_VALUE).
   - min to the largest possible value (Integer.MAX_VALUE).
6. Use a loop to iterate through each element of the array:

- If the current element is greater than max, update max to the current element.
- If the current element is smaller than min, update min to the current element.

7. After the loop, print the values of min (smallest) and max (largest).
8. Stop.

Program

```java
//6. Find largest and smallest number in an array in java
import java.util.Scanner;
class Main{
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        System.out.print("Enter the size of the array:");
        int n= in.nextInt();
        int arr[] = new int[n];

        for(int i=0;i<n;i++){
            arr[i]=in.nextInt();
        }

        int max = Integer.MIN_VALUE;
        int min = Integer.MAX_VALUE;

        for(int i=0;i<n;i++){
            if(arr[i]>max){
                max=arr[i];
            }
            if(arr[i]<min){
                min=arr[i];
            }
        }
        System.out.println("Smallest: "+min+"\nLargest: "+max);
    }
}
```

Output

```
Enter the size of the array:
5
12 343 45 4 6
Largest: 4
Smallest: 343

=== Code Execution Successful ===
```

Result: The program successfully finds the largest and smallest numbers in the array.