CS261 ASSIGNMENT 1

NAME:

ARCHIT AGRAWAL

ROLL NO. :

202052307

SECTION:

A

1. Differentiate between structured programming and object-oriented programming, with the help of the codes. Use two codes to show the varied features.

|  |  |
| --- | --- |
| **STRUCTURED**  **PROGRAMMING** | **OBJECT-ORIENTED**  **PROGRAMMING** |
| It is a subset of procedural programming. | It relies on concept of objects that contain data and code. |
| It follows top-down approach. | It follows bottom-up approach. |
| Programs are divided into small programs or functions. | Programs are divided into objects or entities. |
| It is all about facilitating creation of programs with readable code and reusable components. | It is all about creating objects that usually contain both functions and data. |
| Its main aim is to improve and increase quality, clarity, and development time of computer program. | Its main aim is to improve and increase both quality and productivity of system analysis and design. |
| Structured Programming is **less** secure as there is no way of **data hiding**. | Object Oriented Programming is more secure as having data hiding feature. |
| Structured Programming provides **less** **reusability**, more function dependency. | Object Oriented Programming provides more reusability, less function **dependency**. |
| It provides less flexibility and abstraction as compared to object-oriented programming. | It provides more flexibility and abstraction as compared to structured programming. |
| In this, methods are written globally and code lines are processed one by one i.e., Run sequentially. | In this, method works dynamically, make calls as per need of code for certain time. |
| Examples include Pascal, C etc. | Examples include Java, C++ etc. |

Writing the codes for checking if a number is palindrome or not in both Java and C.

**Java (Object – Oriented)**

import java.util.\*;

public class PalindromeCheck{

    int num;  //instance variable

    //method to find reverse of a number

    public int reverse(int n){

        int rev = 0;

        while(n > 0){

            rev = rev \* 10 + n % 10;

            n /= 10;

        }

        return rev;

    }

    //method to find if the number is palindrome or not

    public boolean isPalindrome(int n){

        if(n == reverse(n)) return true; //if number is equal to its reverse, then it is palindrome, otherwise it is not.

        return false;

    }

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);

        //creating object to use methods of this class

        PalindromeCheck obj = new PalindromeCheck();

        //taking input and initialising the instance variable of the object

        System.out.println("Enter a number to check if it is palindrome or not : ");

        obj.num = sc.nextInt();

        //calling isPalindrome method

        if(obj.isPalindrome(obj.num)){

            System.out.println(obj.num + " is a palindrome.");

        } else {

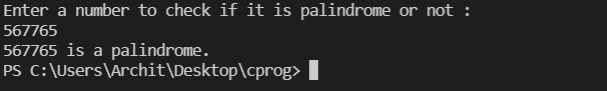
            System.out.println(obj.num + " is not a palindrome.");

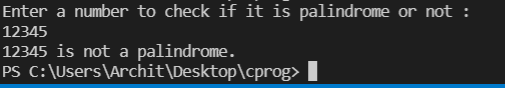
        }

    }

}

**Output:**





**C (Structured Programming)**

#include<stdio.h>

int main(){

    int num;

    //taking input

    printf("Enter a number to check if it is a palindrome or not ");

    scanf("%d", &num);

    //calling isPalindrome method

    if(isPalindrome(num) == 1){

        printf("%d is a palindrome.", num);

    } else {

        printf("%d is not a palindrome.", num);

    }

}

//finding reverse of a number

int reverse(int n){

    int rev = 0;

    while(n > 0){

        rev = rev \* 10 + n % 10;

        n /= 10;

    }

    return rev;

}

//checking if a number is Palindrome or not

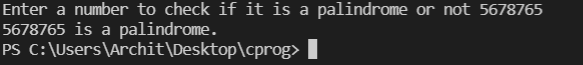
int isPalindrome(int n){

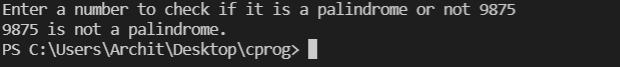
    if(reverse(n) == n) return 1;  //if reverse of num is equal to num then it is palindrome

    return 0;

}

**Output:**





**Observation:**

In Java, we needed to create an object to call the respective methods while in C, there was no such need. The design changes completely. In structured programming we follow Top-Down approach. We just create a function at top and call it simply, but in the object-oriented programming the design is such that code follows bottom-up approach.

We define a class in object-oriented programming we get many benefits like availability of a default constructor, use of instance variable also helps a lot, we can keep all functions of one feature in a single class so the code becomes clean, but this feature wasn’t available in structured programming.

1. Write a program in JAVA to reverse a number using:
2. Loop
3. Recursion
4. Using Loop:

Approach: In this approach, we initiate the value of rev to 0 and update the value of num in each iteration of while loop by num/10. The loop runs till num is greater than 0. In each iteration, we update rev by rev\*10 and add the unit digit of num to rev. When num = 0, then the value of rev is the reverse of the number.

Dry Run:

Input = 34657

Step 1: num = 34657 and rev = 0

[ rev = (0 \* 10) + 7 = 7]

Step 2: num = 3465 and rev = 7

[ rev = (7 \* 10) + 5 = 75]

Step 3: num = 346 and rev = 75

[ rev = (75 \* 10) + 6 = 756]

Step 4: num = 34 and rev = 756

[ rev = (756 \* 10) + 4 = 7564]

Step 5: num = 3 and rev = 7564

[ rev = (7564 \* 10) + 3 = 75643]

1. Using Recursion:

Approach: In this approach, we simply maintain a global variable revRec which stores the number reversed till now and each recursive call is made for n/10 till n becomes 0.

We can do this by extracting the unit digit of the number and then add it to revRec.

The point to be noted here is that we have to multiply revRec by 10 before adding the digit to it in order to update the place value of digits in the reverse so far.

Dry Run:

Input = 12345

Step 1: reverseRecursion(12345) and revRec = 0

Step 2: reverseRecursion (1234) and revRec = 5

Step 3: reverseRecursion (123) and revRec = 54

Step 4: reverseRecursion (12) and revRec = 543

Step 5: reverseRecursion (1) and revRec = 5432

Step 6: reverseRecursion (0) and revRec = 54321

Step 7: return revRec = 54321

public class Reverse{

    public int reverseLoop(int n){

        int rev = 0;

        //rev contains the number reversed till that iteration

        while(n > 0){

            rev = rev \* 10 + n % 10;

            n /= 10;

        }

        return rev;

    }

    int revRec = 0;

    public int reverseRecursion(int n){

        // base condition to end the recursive call

        if(n == 0) return revRec;

        else{

            //revRec stores the number reversed till now

            revRec = (revRec \* 10) + (n % 10);

            // recursive calling of the function

            reverseRecursion(n/10);

        }

        return revRec;

    }

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);

        //creating object of reverse class

        Reverse obj = new Reverse();

        //taking input

        System.out.println("Enter a number");

        int n = sc.nextInt();

        //reversing it using loop

        System.out.println("Original Number : " + n + " :: Reverse Number : "+ obj.reverseLoop(n));

        System.out.println("Enter another number");

        //taking another input

        n = sc.nextInt();

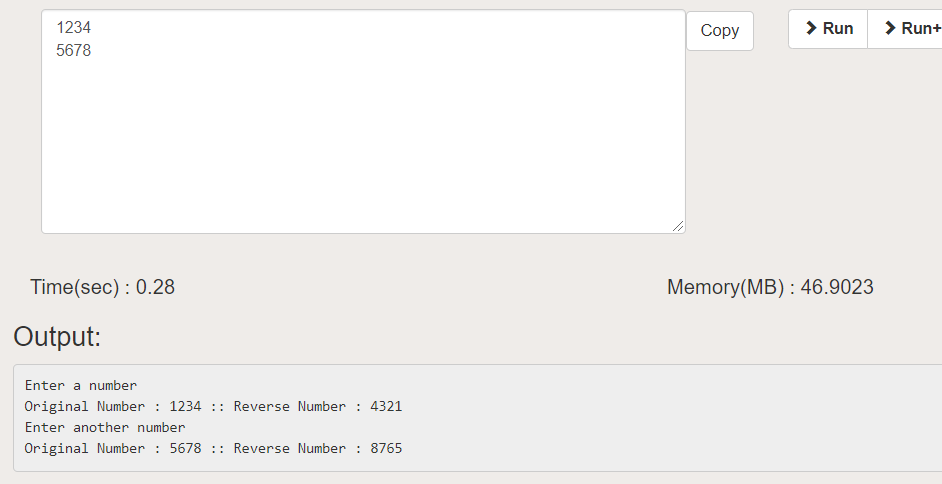
        //reversing it using recursion

        System.out.println("Original Number : "+  n + " :: Reverse Number : "+ obj.reverseRecursion(n));

    }

}





1. Write a program in JAVA to implement Linear Search using Arrays.

**Approach**: We begin with index 0 of the array and compare each element in the array with the target element(the element to be found). If found, we return the index at which the element is present, else we return -1. As -1 cannot be an index of array, hence it means that the element is not found.

import java.util.\*;

public class LinearSearch {

    public int search(int[] arr, int num){

        //checking if num is present in the array

        //if present, return the first index at which it is present

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

            if(arr[i] == num) return i;

        }

        return -1;

    }

    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);

        //creating object of the class

        LinearSearch obj = new LinearSearch();

        System.out.print("Enter total number of elements of array: ");

        int n = sc.nextInt();

        //inputting array elements

        int[] arr = new int[n];

        System.out.println("Enter the elements of the array: ");

        for (int i = 0; i < n; i++) {

            arr[i] = sc.nextInt();

        }

        //input value to search

        System.out.println("Enter value to search");

        int val = sc.nextInt();

        int index = obj.search(arr, val);

        //Printing results

        if (index == -1)

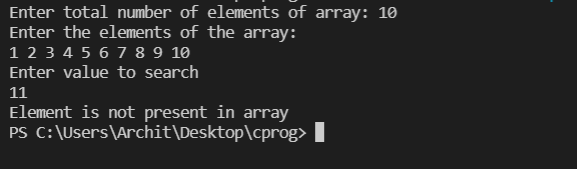
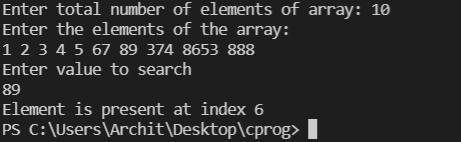
            System.out.print("Element is not present in array");

        else

            System.out.print("Element is present at index " + index);

    }

}

1. Write a program to create a room class, the attributes/variables of this class are roomNo, roomType, and roomArea. In this class, the member functions are setData and displayData. The program should perform some tasks such as assigning roomArea, printing room details, etc.

**Approach**:

In this, we create a class named as ‘Room’ and then we create three instance variables in it named roomNo, roomArea, roomType. This class has a default constructor and another constructor that can assign a specific roomNo at the the time of creating the object. After this we created a method called setData(int num , String type, double area) which basically sets the values of the three instance variables equal to the values of its parameters. We can use either Math.random() and Math.round() for roomNo and roomArea or we can also take them as input from the user.

The display() method displays the details of the object of class Room.

import java.util.\*;

public class Room {

    //declaring instance variables

    static int roomNo;

    static String roomType;

    static double roomArea;

    //default constructor for class Room

    //it assigns roomNo = 0

    //roomType = ""

    //roomArea = 0

    Room(){

        this.roomNo = 0;

        this.roomType = "";

        this.roomArea = 0.0;

    }

    //another constructor

    //it assigns roomNo = number

    Room(int number){

        this.roomNo = number;

    }

    // method to set data

    public void setData(int num, String type, double area) {

        roomNo = num;

        roomType = type;

        roomArea = area;

    }

    //method to display the details of room

    public void displayData() {

        System.out.println("The Room Number : " + roomNo);

        System.out.println("The Room Type : " + roomType);

        System.out.println("The Room Area : " + roomArea + " sq.ft");

    }

    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);

        Room room1 = new Room(); //creating object of Room class

        System.out.println("Inputting Details for Room 1");

        System.out.println("Enter room number : ");

        //inputting room number from user

        int n = sc.nextInt();

        room1.roomNo = n;

        System.out.println("Enter room type : ");

        //inputting room type from user

        sc.nextLine();

        String roomType = sc.nextLine();

        //assigning room area using random method

        double roomArea = 1000 + Math.round(1000 \* Math.random());

        //calling setData to set the data for the room1 object

        room1.setData(roomNo, roomType, roomArea);

        System.out.println("\*\*\* Details of room1 \*\*\*");

        room1.displayData();

        //creating room2 object with roomNo 52

        Room room2 = new Room(52);

        System.out.println();

        System.out.println();

        System.out.println("Inputting other details for Room 2");

        System.out.println("Enter room type : ");

        //inputting room type from user

        roomType = sc.nextLine();

        //assigning room area using random method

        roomArea = 1000 + Math.round(1000 \* Math.random());

        //calling setData to set the data for the room2 object

        room2.setData(roomNo, roomType, roomArea);

        System.out.println("\*\*\* Details of room2 \*\*\*");

        room2.displayData();

    }

}

**Output** :

