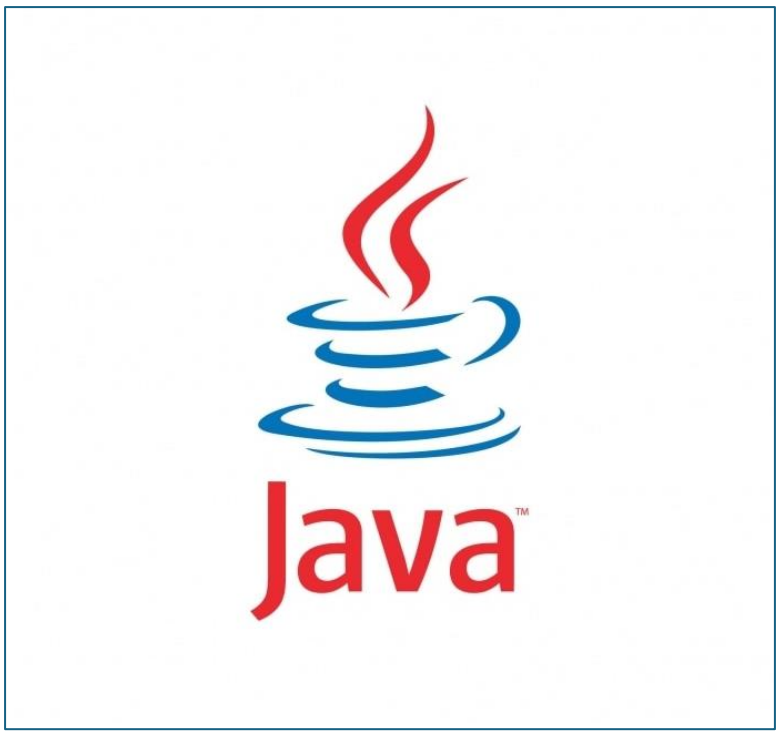

Scottish High International School

Computer Science Project

Name: Aaryan Ved Bhalla

Class: XII-A

Session: 2024-2025

Acknowledgement

I would like to thank the ISC Council board for giving me the opportunity to make this wonderful project, which would make sure that the research skills of the students are also being taken into account apart from the examination scores.

I would also like to thank my School Principal Mr. Sanjay Sachdeva and my Computer Science teacher, Ms. Surbhi Mathur for guiding me, supporting me and telling me how my project should come out better and giving all the necessary help required.

Lastly, I would also like to thank my parents for supporting and encouraging me throughout the development of this project.

Index

Inheritance Programs		
22	Rev → Palin	Accessing Rev Super Class to Find if a number is Palindrome or not.
23	Student → Marks	Accessing Student Super Class to present marks and grade of a Student
24	ISC_Scores → BestFour	Accessing ISC_Scores Class to present the four best marks scored and their respective subjects
25	Vehicle → Car	Accessing Vehicle Class to Display the Brand Name, Year of Release and No. of Doors

Step 1: Start of algorithm

Step 2: Input a string and store it in the variable 's'

Step 3: Create a Scanner 'sc' to read the words from the string

Step 4: Declare and initialize variables: n=0, p=0, sum=0, k=0, l=0, ts="", x=""

Step 5: Create arrays p1[] to store potential values and s1[] to store words

Step 6: Use a while loop to read words from the string using 'sc' and store them in the array s1[]

Step 7: Display the original string by iterating through the s1[] array

Step 8: Use a loop to calculate the potential values for each word and store them in the p1[] array

Step 9: Use nested loops to arrange the words in ascending order based on their potential values

Step 10: Display the new string after sorting

Step 11: End of algorithm

```java
//Potential in Acending Order
import java.util.*;
public class stringPotential
{
    public static void main(String args[])
    {
        Scanner in=new Scanner(System.in);
        System.out.println("Enter String: ");
        String s=in.nextLine();
        Scanner sc=new Scanner(s);
        int n=0,p=0,sum=0,k=0,l=0;
        //char ch=' ';
        String ts="",x="";
        int p1[]=new int[10];
        String s1[]=new String[10];
        while(sc.hasNext())
        {
            s1[k]=sc.next();
            k++;
        }
        System.out.println("Original String: ");
        for(int i=0;i<k;i++)
        {
            System.out.print(s1[i] + " ");
        }
        for(int i=0;i<k;i++)
        {
            l=s1[i].length();
            l=l-1;
            for(int j=0;j<l;j++)
            {
                //ch=s1[i].charAt(j);
                sum=sum+(int)s1[i].charAt(j);
            }
            p1[i]=sum;
            sum=0;
        }
        for(int i=0;i<k;i++)
        {
            for(int j=i+1;j<k;j++)
            {
                if(p1[i]>p1[j])
                {
                    ts=s1[i];
                    s1[i]=s1[j];
                    s1[j]=ts;
                }
            }
        }
        System.out.println();
```



```java
        System.out.println("New String: ");
        for(int i=0;i<k;i++)
        {
            System.out.print(s1[i] + " ");
        }
    }
}
```

```
Enter String:
my name is aaryan
Original String:
my name is aaryan
New String:
is my name aaryan
```

Step 1: Start of algorithm

Step 2: Input a string and store it in the variable 's'

Step 3: Concatenate a space to the end of the string 's'

Step 4: Convert the entire string to lowercase using 'toLowerCase()'

Step 5: Declare and initialize an empty string 's3' to store the result

Step 6: Use a loop to iterate through each character in the string 's'

Step 7: Check if the current character is equal to the next character

Step 8: If equal, continue to the next iteration (skip duplicates)

Step 9: If not equal, append the current character to the string 's3'

Step 10: Display the modified string 's3' without duplicate words

Step 11: End of algorithm

```java
import java.util.*;
public class replRepeat
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter String: ");
        String s=sc.nextLine();
        s=s+" ";
        s=s.toLowerCase();
        String s3="";
        for(int i=0;i<s.length()-1;i++)
        {
            if(s.charAt(i)==s.charAt(i+1))
            {
                continue;
            }
            else
            {
                s3=s3+s.charAt(i);
            }
        }
        System.out.println(s3);
    }
}
```

```
Enter String:
aaryan bhalla
aryan bhala
```

Step 1: Start of algorithm

Step 2: Input a string and store it in the variable 's'

Step 3: Initialize variables: i=0, count=0, counts=0, l=length of the string, ch=' ', c=' '

Step 4: Convert the entire string to lowercase using 'toLowerCase()'

Step 5: Use a nested loop:

a. Outer loop: Iterate through each character ('c') from 'a' to 'z'

b. Inner loop: Iterate through each character ('ch') in the string 's'

Step 6: Inside the inner loop:

a. Check if 'ch' is equal to the current character 'c'

b. If true, increment 'count'

Step 7: After the inner loop, check if 'count' is greater than 0

a. If true, print the frequency of the character 'c'

Step 8: Reset 'count' to 0 for the next iteration

Step 9: After the outer loop, calculate the frequency of spaces by dividing 'counts' by 26 (assuming 26 letters in the alphabet)

Step 10: Display the frequency of spaces

Step 11: End of algorithm

```java
//frequency of each and every character in string including spaces
import java.util.*;
public class charFreq
{
    public static void main(String args[])
    {
        Scanner in=new Scanner(System.in);
        System.out.println("Enter String: ");
        String s=in.nextLine();
        int i=0,count=0,counts=0;
        int l=s.length();
        char ch=' ',c=' ';
        s=s.toLowerCase();
        for(c='a'; c<='z'; c++)
        {
            for(i=0;i<l;i++)
            {
                ch=s.charAt(i);
                if(ch==c)
                {
                    count++;
                }
                else if(ch==' ')
                {
                    counts++;
                }
            }
            if(count>0)
            {
                System.out.println(c+" appears " + count + " times.");
            }
            count=0;
        }
        System.out.println("Spaces appear " + (counts/26)+ " times.");
    }
}
```

```
Enter String:
aaryan ved bhalla
a appears 5 times.
b appears 1 times.
d appears 1 times.
e appears 1 times.
h appears 1 times.
l appears 2 times.
n appears 1 times.
r appears 1 times.
v appears 1 times.
y appears 1 times.
Spaces appear 2 times.
```

Step 1: Start of algorithm

Step 2: Input a string and store it in the variable 's'

Step 3: Convert the entire string to lowercase using 'toLowerCase()'

Step 4: Create a Scanner 'sc' to read words from the string

Step 5: Create an array 's1' to store words and initialize 'k' to 0

Step 6: Use a while loop to read words from the string using 'sc' and store them in the array 's1'

Step 7: Create a new array 's2' with a size of 'k' to store the words

Step 8: Copy the elements from 's1' to 's2'

Step 9: Use nested loops for sorting:

a. Outer loop: Iterate from i=0 to i<k

b. Inner loop: Iterate from j=i+1 to j<k

c. Compare words using 'compareTo()' and swap if necessary

Step 10: Display the sorted array 's2'

Step 11: End of algorithm

```java
import java.util.*;
public class ascendString
{
    public static void main(String args[])
    {
        Scanner in=new Scanner(System.in);
        System.out.println("Enter String: ");
        String s=in.nextLine();
        s=s.toLowerCase();
        Scanner sc=new Scanner(s);
        String s1[]=new String[10];
        int k=0;
        while(sc.hasNext())
        {
            s1[k]=sc.next();
            k++;
        }
        String s2[]=new String[k];
        for(int i=0;i<k;i++)
        {
            s2[i]=s1[i];
        }
        for(int i=0;i<=k;i++)
        {
            for(int j=i+1;j<k;j++)
            {
                if(s2[i].compareTo(s2[j])>0)
                {
                    String t=s2[i];
                    s2[i]=s2[j];
                    s2[j]=t;
                }
            }
        }
        for(int i=0;i<k;i++)
        {
            System.out.print(s2[i]+ " ");
        }
    }
}
```

```
Enter String:
my name is aaryan
aaryan is my name
```

Step 1: Start of algorithm

Step 2: Input a string and store it in the variable 's'

Step 3: Convert the entire string to lowercase using 'toLowerCase()'

Step 4: Create a Scanner 'sc' to read words from the string

Step 5: Create an array 's1' to store words and initialize 'n' to 0

Step 6: Use a while loop to read words from the string using 'sc' and store them in the array 's1'

Step 7: Display the total number of words in the array 's1'

Step 8: Initialize variables 'count' to 0 and 'f' to 0

Step 9: Use a loop to iterate through each word in the array 's1'

a. Get the length of the current word 'l'

b. Use nested loops to compare each character in the word:

- Outer loop (j): Iterate from 0 to l-1
- Inner loop (k): Iterate from 0 to l-1
  - Check if the character at position j is not equal to the character at position k+1
  - If true, increment 'f' and break from the inner loop

c. If 'f' is greater than 0, increment 'count'

d. Reset 'f' to 0

Step 10: Display the total number of unique words ('count')

Step 11: End of algorithm

```java
import java.util.*;
public class uniqWord
{
    public static void main(String args[])
    {
        Scanner in=new Scanner(System.in);
        System.out.println("Enter a String: ");
        String s=in.nextLine();
        s=s.toLowerCase();
        Scanner sc=new Scanner(s);
        String s1[]=new String[10];
        int n=0,l=0,f=0,count=0;
        while(sc.hasNext())
        {
            s1[n]=sc.next();
            n++;
        }
        System.out.println("Total No. of Words: " + n);
        for(int i=0;i<n;i++)
        {
            l=s1[i].length();
            for(int j=0; j<l; j++)
            {
                for(int k=0; k<l; k++)
                {
                    if(s1[i].charAt(j)!=s1[i].charAt(k+1))
                    {
                        f++;
                        break;
                    }
                    else
                    {
                        break;
                    }
                }
                if(f>0)
                {
                    count++;
                    break;
                }
            }
            f=0;
        }
        System.out.println("Total No. of Unique Words: " + count);
    }
}
```

```
Enter a String:
aaryan ved bhalla
Total No. of Words: 3
Total No. of Unique Words: 3
```

Step 1: Start of algorithm

Step 2: Input the size of the square array and store it in the variable 'n'

Step 3: Create a 2D array 'a' of size 'n x n' to represent the square matrix

Step 4: Declare and initialize variables: r1=0, r2=n-1, c1=0, c2=n-1, k=1

Step 5: Use a while loop to fill the array in a reverse spiral pattern until 'k' reaches 'n*n'

Step 6: Inside the while loop:

a. Fill the top row from column 'c1' to 'c2' with natural numbers

b. Fill the right column from row 'r1+1' to 'r2' with natural numbers

c. Fill the bottom row from column 'c2-1' to 'c1' with natural numbers

d. Fill the left column from row 'r2-1' to 'r1+1' with natural numbers

e. Update indices: increment 'c1', decrement 'c2', increment 'r1', decrement 'r2'

Step 7: Display the final array

Step 8: End of algorithm

```java
//Aaryan Ved Bhalla, XI-A, Reverse Spiral DDA Natural Number
import java.util.*;
public class revSpiralDDA
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Size of Array: ");
        int n=sc.nextInt();
        int a[][]=new int[n][n];
        int r1=0, r2=n-1, c1=0, c2=n-1, k=1;
        while(k<=(n*n))
        {
            for(int i=c1; i<=c2; i++)
            {
                a[c1][i]=k++;
            }
            for(int i=r1+1; i<=r2; i++)
            {
                a[i][r2]=k++;
            }
            for(int i=c2-1; i>=c1; i--)
            {
                a[c2][i]=k++;
            }
            for(int i=r2-1; i>=r1+1; i--)
            {
                a[i][r1]=k++;
            }
            c1++; c2--; r1++; r2--;
        }
        System.out.println("Final Array: ");
        for(int i=0; i<n; i++)
        {
            for(int j=0; j<n; j++)
            {
                System.out.print(a[i][j] + " ");
            }
            System.out.println();
        }
        sc.close();
    }
}
```

```
Enter Size of Array:
3
Final Array:
1 2 3
8 9 4
7 6 5
```

Step 1: Start of algorithm

Step 2: Input the number of people 'n'

Step 3: Create arrays 'name' and 'no' to store names and mobile numbers

Step 4: Use a loop to input names:

a. Loop from i=0 to i<=n

b. Input names and store them in the 'name' array

Step 5: Display a message to input mobile numbers

Step 6: Use a loop to input mobile numbers:

a. Loop from i=0 to i<n

b. Input mobile numbers and store them in the 'no' array

Step 7: Use nested loops for sorting:

a. Outer loop: Iterate from i=0 to i<=n

b. Inner loop: Iterate from j=i+1 to j<=n

c. Compare names using 'compareTo()' and swap if necessary

d. Also, swap corresponding mobile numbers

Step 8: Display the sorted names and corresponding mobile numbers

Step 9: End of algorithm

```java
//Sort Mobile Numbers and Names
import java.util.*;
public class cplxSelectionSort
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Number of People: ");
        int n=sc.nextInt();
        String name[]=new String[n+1];
        long no[]=new long[n];
        System.out.println("Enter Names: ");
        for(int i=0;i<=n;i++)
        {
            name[i]=sc.nextLine();
        }
        System.out.println("Enter Mobile Numbers: ");
        for(int i=0;i<n;i++)
        {
            no[i]=sc.nextLong();
        }
        for(int i=0;i<=n;i++)
        {
            for(int j=i+1;j<=n;j++)
            {
                if(name[i].compareTo(name[j])>0)
                {
                    String t=name[i];
                    long temp=no[i];
                    name[i]=name[j];
                    no[i]=no[j];
                    name[j]=t;
                    no[j]=temp;
                }
            }
        }
        System.out.println("Name: " + "\t" + "Mobile Number: ");
        for(int i=0;i<=n;i++)
        {
            System.out.println(name[i] + "\t" + no[i]);
        }
    }
}
```

```
Enter Number of People:
2
Enter Names:
aaryan
arsh
Enter Mobile Numbers:
57747
123123123
Name:    Mobile Number:
arsh     57747
aaryan   123123123
```

Step 1: Start of algorithm

Step 2: Input the size of the array and store it in the variable 'n'

Step 3: Create an array 'a' to store elements

Step 4: Display a message to enter array elements

Step 5: Use a loop to input elements:

a. Loop from i=0 to i<n

b. Input elements and store them in the array 'a'

Step 6: Initialize variables: i=0, flag=0

Step 7: Use a loop to check if elements are in ascending order:

a. Loop from i=0 to i<n-1

b. Check if a[i] > a[i+1], set 'flag' to 1 and break the loop

c. Otherwise, set 'flag' to 0

Step 8: Display whether the array is in ascending order or not based on 'flag'

Step 9: End of algorithm

```java
//Write a program to check wether the numbers existing in the array are in ascending order or not.
import java.util.*;
public class checkAscending
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Size of Array: ");
        int n=sc.nextInt();
        int a[]=new int[n];
        System.out.println("Enter Elements: ");
        int i=0;
        int flag=0;
        for(i=0; i<n;i++)
        {
            a[i]=sc.nextInt();
        }
        for(i=0; i<n-1;i++)
        {
            if(a[i]>a[i+1])
            {
                flag=1;
                break;
            }
            else
            {
                flag=0;
            }
        }
        if(flag==1)
            {
                System.out.println("It is not in Ascending Order");
            }
            else
            {
                System.out.println("It is in Ascending Order");
            }
    }
}
```

```
Enter Size of Array:
3
Enter Elements:
1
2
3
It is in Ascending Order
```

Step 1: Start of algorithm

Step 2: Define a class 'arrayMax' with instance variables 'm' and 'arr'

Step 3: Create a constructor to initialize 'm' and 'arr'

Step 4: Create methods:

a. 'readarray()' to input elements into the 2D array 'arr'

b. 'large()' to find and display the largest element in each row

c. 'display()' to print the entire array

Step 5: Inside 'readarray()', use nested loops to read elements into the 2D array 'arr'

Step 6: Inside 'large()', use nested loops to find the largest element in each row

Step 7: Inside 'display()', use nested loops to print the entire array

Step 8: In the 'main' method:

a. Input the array range 'c'

b. Create an object 'ob' of class 'arrayMax' with range 'c'

c. Call 'readarray()', 'large()', and 'display()' methods on 'ob'

Step 9: End of algorithm

```java
import java.util.*;
public class arrayMax
{
    int m;
    int arr[][];
    arrayMax(int mm)
    {
        m=mm;
        arr= new int[m][m];
    }
    void readarray()
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the Elements: ");
        for(int i=0; i<m;i++)
        {
            for(int j=0; j<m;j++)
            {
                arr[i][j]=sc.nextInt();
            }
        }
    }

    void large()
    {
        int max=0, i, j;
        for(i=0; i<m;i++)
        {
            max=arr[i][0];
            for(j=0; j<m;j++)
            {
                if(max<arr[i][j])
                {
                    max=arr[i][j];
                }
            }
            System.out.println("Largest Element in row "+ (i+1) + " is " + max);
        }
    }

    void display()
    {
        System.out.println("Array");
        for(int i=0; i<m;i++)
        {
            for(int j=0; j<m; j++)
            {
                System.out.print(arr[i][j] + " ");
            }
            System.out.println("");
```

```
        }
    }

    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Array Range: ");
        int c=sc.nextInt();
        arrayMax ob=new arrayMax(c);
        ob.readarray();
        ob.large();
        ob.display();
    }
}
```

```
Enter Array Range:
3
Enter the Elements:
1
2
3
4
5
6
7
8
9
Largest Element in row 1 is 3
Largest Element in row 2 is 6
Largest Element in row 3 is 9
Array
1 2 3
4 5 6
7 8 9
```

Step 1: Start of algorithm

Step 2: Declare and initialize a 3x3 matrix 'a' with given values

Step 3: Initialize variables: sumr=0, sumc=0, sumd=0, temp=0, flag=0

Step 4: Display the given matrix

Step 5: Use nested loops to calculate the sum of rows and columns:

a. Loop from i=0 to i<3

b. Inside the loop:

- Initialize sumr and sumc to 0

- Loop from j=0 to j<3:

* Add a[i][j] to sumr

* Add a[j][i] to sumc

Step 6: Use a loop to calculate the sum of the main diagonal:

a. Loop from i=0 to i<3

- Add a[i][i] to sumd

Step 7: Check if sumd is equal to sumc and sumc is equal to sumr:

a. If true, display "It is a Magic Square"

b. Otherwise, display "It is not a Magic Square"

Step 8: End of algorithm

```java
import java.util.*;
public class magiSq
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        //System.out.println("HAPPY BIRTHDAY AARYAN!!");
        System.out.println("Given Matrix");
        int a[][]={{2,7,6},{9,5,1},{4,3,8}};
        int sumr=0, sumc=0, sumd=0,temp=0, flag=0;
        for(int i=0;i<3;i++)
        {
            for(int j=0;j<3;j++)
            {
                sumr=+a[i][j];
                sumc=+a[j][i];
            }
        }
        for(int i=0;i<3;i++)
        {
            sumd=+a[i][i];
        }
        if(sumd==sumc && sumc==sumr)
        {
            System.out.println("It is a Magic Square");
        }
        else
        {
            System.out.println("It is not a Magic Square");
        }
    }
}
```

```
Given Matrix
Array:
2 7 6
9 5 1
4 3 8
It is a Magic Square
```

Step 1: Start of algorithm

Step 2: Define a class 'factRecur'

Step 3: Declare a method 'fact' inside the class:

a. Takes an integer 'n' as an argument

b. If 'n' is 0, return 1

c. Otherwise, return 'n * fact(n-1)'

Step 4: In the 'main' method:

a. Display a message to enter a number

b. Input a number 'n'

c. Create an object 'ob' of class 'factRecur'

d. Call the 'fact' method on 'ob' with 'n' as an argument

e. Display the result

Step 5: End of algorithm

```java
import java.util.*;
public class factRecur
{
    int fact(int n)
    {
        if(n==0)
        {
            return 1;
        }
        return n*fact(n-1);
    }
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Number: ");
        int n=sc.nextInt();
        int a=0;
        factRecur ob=new factRecur();
        System.out.println(ob.fact(n));
    }
}
```

```
Enter Number:
5
120
```

Step 1: Start of algorithm

Step 2: Define a class 'multRecur'

Step 3: Declare a method 'mult' inside the class:

a. Takes two integers 'x' and 'y' as arguments

b. If 'x' or 'y' is 0, return 0

c. Otherwise, return 'x + mult(x, y-1)'

Step 4: In the 'main' method:

a. Display a message to enter two numbers for multiplication

b. Input two numbers 'a' and 'b'

c. Create an object 'ob' of class 'multRecur'

d. Call the 'mult' method on 'ob' with 'a' and 'b' as arguments

e. Display the product

Step 5: End of algorithm

```java
import java.util.*;
public class multRecur
{
    int mult(int x, int y)
    {
        if(x==0 || y==0)
        {
            return 0;
        }
        else
        {
            return x+mult(x,y-1);
        }
    }
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Two Numbers to be Multiplied: ");
        int a=sc.nextInt();
        int b=sc.nextInt();
        multRecur ob=new multRecur();
        System.out.println("The Product is: " + ob.mult(a,b));
    }
}
```

```
Enter Two Numbers to be Multiplied:
2
5
The Product is: 10
```

Step 1: Start of algorithm

Step 2: Define a class 'fibRecur'

Step 3: Declare a method 'fib' inside the class:

a. Takes an integer 'n' as an argument

b. If 'n' is less than or equal to 1, return 'n'

c. Otherwise, return 'fib(n-1) + fib(n-2)'

Step 4: In the 'main' method:

a. Display a message to enter a number

b. Input a number 'n'

c. Create an object 'ob' of class 'fibRecur'

d. Call the 'fib' method on 'ob' with 'n' as an argument

e. Display the nth Fibonacci number

Step 5: End of algorithm

```java
import java.util.*;
public class fibRecur
{
    int fib(int n)
    {
        if(n<=1)
        {
            return n;
        }
        return fib(n-1)+fib(n-2);
    }
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Number: ");
        int n=sc.nextInt();
        fibRecur ob=new fibRecur();
        System.out.println("The " + n + "th Fibonacci Number is " + ob.fib(n));
    }
}
```

```
Enter Number:
10
The 10th Fibonacci Number is 55
```

Step 1: Start of algorithm

Step 2: Define a class 'b2dRecur'

Step 3: Declare an instance variable 'x' to track the position of digits

Step 4: Declare a method 'conv' inside the class:

a. Takes an integer 'n' as an argument

b. If 'n' is 0, return 0

c. Otherwise, calculate the decimal equivalent using recursion:

* Increment 'x' by 1

* Return (n % 10 * 2^x) + conv(n/10)

Step 5: In the 'main' method:

a. Display a message to enter a binary number

b. Input a binary number 'l'

c. Create an object 'ob' of class 'b2dRecur'

d. Call the 'conv' method on 'ob' with 'l' as an argument

e. Display the decimal equivalent by dividing the result by 2

Step 6: End of algorithm

```java
import java.util.*;

public class b2dRecur
{
    int x=0;
    int conv(int n)
    {
        if(n==0)
            return 0;
        else
        {
            x++;
            return ((n%10* (int)Math.pow(2,x))+ conv(n/10));
        }
    }
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter a No. to be Converted: ");
        int l=sc.nextInt();
        b2dRecur ob = new b2dRecur();
        int y=ob.conv(l);
        System.out.println("Decimal Number is: "+ y/2);
    }
}
```

```
Enter a No. to be Converted:
10111
Decimal Number is: 23
```

Step 1: Start of algorithm

Step 2: Define a class 'd2bRecur'

Step 3: Declare an instance variable 'x' to store the remainder

Step 4: Declare a method 'conv' inside the class:

a. Takes an integer 'n' as an argument

b. If 'n' is 0, return 0

c. Otherwise, calculate the binary equivalent using recursion:

* Set 'x' to n % 2

* Return x + (conv(n / 2) * 10)

Step 5: In the 'main' method:

a. Display a message to enter a decimal number

b. Input a decimal number 'a'

c. Create an object 'ob' of class 'd2bRecur'

d. Display the binary equivalent using the 'conv' method on 'ob' with 'a' as an argument

Step 6: End of algorithm

```java
import java.util.*;
public class d2bRecur
{
    int x=1;
    int conv(int n)
    {
        if(n==0)
            return 0;
        else
        {
            x=n%2;
            return x+(conv(n/2)*10);
        }
    }
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter No. to be Converted : ");
        int a =sc.nextInt();
        d2bRecur ob=new d2bRecur();
        System.out.println("Binary No. is: ");
        System.out.println(ob.conv(a));
    }
}
```

```
Enter No. to be Converted :
23
Binary No. is:
10111
```

Step 1: Start of algorithm

Step 2: Define a class 'powRecur'

Step 3: Declare an instance variable 'n' to store the base value

Step 4: Declare a method 'pow' inside the class:

a. Takes two integers 'b' (base) and 'i' (exponent) as arguments

b. If 'i' is 0, return 1

c. Otherwise, return 'b * pow(b, i-1)'

Step 5: In the 'main' method:

a. Display a message to enter a base number and its exponent

b. Input a base number 'n' and its exponent 'p'

c. Create an object 'ob' of class 'powRecur'

d. Call the 'pow' method on 'ob' with 'n' and 'p' as arguments

e. Display the result of 'n' to the power of 'p'

Step 6: End of algorithm

```java
import java.util.*;
public class powRecur
{
    int n;
    int pow(int b, int i)
    {
        if(i==0)
            return 1;
        else
            return b*pow(b,i-1);
    }
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Number and Power: ");
        int n=sc.nextInt();
        int p=sc.nextInt();
        powRecur ob=new powRecur();
        int ans=ob.pow(n,p);
        System.out.println(n+" to the power "+p+" = " + ans);
    }
}
```

```
Enter Number and Power:
2
3
2 to the power 3 = 8
```

Step 1: Start of algorithm

Step 2: Define a class 'primePalinGen'

Step 3: Declare instance variables 'start', 'end', and 'flag

Step 4: Create a constructor inside the class:

a. Takes two integers 'a' and 'b' as arguments

b. Initialize 'start' with 'a' and 'end' with 'b'

Step 5: Declare methods:

a. 'isPrime' to check if a number is prime

b. 'isPalin' to check if a number is a palindrome

c. 'generate' to generate and display prime palindrome numbers in the range [start, end]

Step 6: Inside 'isPrime' method:

a. Set 'flag' to 1

b. Use a loop to check for divisibility from 2 to i-1

c. If i is divisible by any number, set 'flag' to 0

d. Return 'flag'

Step 7: Inside 'isPalin' method:

a. Initialize variables 'temp', 'd', 'r', and 'flag'

b. Use a loop to reverse the digits of 'i'

c. If the reversed number is equal to 'i', set 'flag' to 1

d. Return 'flag'

Step 8: Inside 'generate' method:

a. Use a loop from 'start' to 'end'

b. Check if the number is both prime and palindrome using 'isPrime' and 'isPalin'

c. If true, display the number as a prime palindrome number

Step 9: In the 'main' method:

a. Display a message to enter start and end values

b. Input start 's' and end 'e'

c. Create an object 'ob' of class 'primePalinGen' with 's' and 'e'

d. Call the 'generate' method on 'ob'

Step 10: End of algorithm

```java
import java.util.*;
public class primePalinGen
{
    int start;
    int end;
    int flag;
    primePalinGen (int a, int b)
    {
        start=a;
        end=b;
    }
    int isPrime(int i)
    {
        flag =1;
        for(int j=2; j<i; j++)
        {
            if(i%j==0)
            {
                flag=0;
            }
        }
        return flag;
    }
    int isPalin(int i)
    {
        int temp=i, d=0, r=0, flag=0;
        while(i>0)
        {
            d=temp%10;
            r=(r*10)+d;
            temp=temp/10;
        }
        if(temp==i)
        {
            flag=1;
        }
        return flag;
    }
    void generate()
    {
        for(int j=start; j<=end; j++)
        {
            if(isPalin(j)==1 && isPrime(j)==1)
            {
                System.out.println(j + "is a prime palindrome number");
            }
        }
    }
    public static void main(String args[])
    {
```

```
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter start and end values: ");
        int s = sc.nextInt();
        int e = sc.nextInt();
        primePalinGen ob=new primePalinGen(s,e);
        ob.generate();
    }
    }
```

```
Enter start and end values
100
200
101
131
151
181
191
```

Step 1: Start of algorithm

Step 2: Declare variables 'a', 'b', 'c' to store Fibonacci series terms

Step 3: Display a message to enter 'n'

Step 4: Input 'n' from the user

Step 5: Display the initial terms of the Fibonacci series: a=0, b=1

Step 6: Use a loop to generate and display the Fibonacci series up to the nth term:

a. Loop from i=3 to i=n

b. Inside the loop:

- Calculate the next term 'c' as the sum of 'a' and 'b'
- Display 'c' followed by a comma
- Update 'a' to the value of 'b'
- Update 'b' to the value of 'c'

Step 7: End of algorithm

```java
import java.util.*;
public class fibSeries
{
    public static void main(String argsp[])
    {
        Scanner sc=new Scanner(System.in);
        int a=0;
        int b=1;
        int c=0;
        System.out.println("Enter n: ");
        int n=sc.nextInt();
        System.out.print(a + "," + b);
        for(int i=3;i<=n;i++)
        {
            c=a+b;
            System.out.print(","+c);
            a=b;
            b=c;
        }
    }
}
```

```
Enter n:
10
0,1,1,2,3,5,8,13,21,34
```

Step 1: Start of algorithm

Step 2: Initialize an empty string 's' to store user input for continuation

Step 3: Start a do-while loop:

a. Display the menu options: 1. Perfect No, 2. Armstrong No, 3. Prime No

b. Input the user's choice 'choice'

c. Input the number 'num'

Step 4: Use a switch statement based on the user's choice:

a. Case 1 (Perfect No):

- Initialize 'sum' to 0
- Loop from i=1 to i<6:
  * If num is divisible by i, add i to 'sum'
- If 'sum' equals 'num', display "Perfect No", otherwise "Not a Perfect No"

b. Case 2 (Armstrong No):

- Initialize 'sum' to 0
- Loop while num > 0:
  * Extract the last digit 'd' from num
  * Add d^3 to 'sum'
  * Update num to num/10
- If 'sum' equals 'k', display "Armstrong No", otherwise "Not an Armstrong No"

c. Case 3 (Prime No):

- Initialize 'count' to 0
- Loop from i=1 to i<=k:
  * If k is divisible by i, increment 'count'
- If 'count' equals 2, display "Prime No", otherwise "Not a Prime No"

d. Default case: Display "Invalid input" for any other choice

Step 5: Prompt the user with "Do you want to continue Yes/No"

a. Input the user's response into 's'

b. Continue the loop if 's' is "Yes" (case insensitive)

Step 6: End of algorithm

```java
import java.util.Scanner;
public class perfArmPrim
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        String s="";
        do{
            System.out.println("1. Perfect No\n2.Armstrong no\n3. Prime No");
            System.out.println("Enter your choice");
            int choice=sc.nextInt();
            System.out.println("Enter number");
            int num=sc.nextInt();
            int k=num;
            switch(choice)
            {
                case 1:
                    int sum=0;
                    for(int i=1;i<6;i++)
                    {
                        if(num%i==0)
                            sum=sum+i;
                    }
                    if(sum==k)
                        System.out.println("Perfect No");
                    else
                        System.out.println(" Not a Perfect No");
                    break;
                case 2:
                    sum=0;
                    while(num>0)
                    {
                        int d=num%10;
                        sum=sum+(int)Math.pow(d,3);
                        num=num/10;
                    }
                    if(sum==k)
                        System.out.println("Armstrong No");
                    else
                        System.out.println(" Not an Armstrong No");
                    break;
                case 3:
                    int count=0;
                    for(int i=1;i<=k;i++)
                    {
                        if(k%i==0)
                            count++;
                    }
                    if(count==2)
                        System.out.println("Prime No");
```

```
                    else
                        System.out.println(" Not a Prime No");
                    break;
                default:
                    System.out.println("Invalid input");
            }
            System.out.println("Do you want to continue Yes/No");
            s=sc.next();
        }
        while(s.equalsIgnoreCase("Yes"));
    }
}
```

```
1. Perfect No
2.Armstrong no
3. Prime No
Enter your choice
3
Enter number
2
Prime No
Do you want to continue Yes/No
Yes
1. Perfect No
2.Armstrong no
3. Prime No
Enter your choice
2
Enter number
45
 Not an Armstrong No
Do you want to continue Yes/No
No
```

Can only enter input while your program is r

Step 1: Start of algorithm

Step 2: Display the quadratic form: Ax^2 + Bx + C

Step 3: Input coefficients A, B, and C from the user

Step 4: Calculate the discriminant (d) using the formula d = B^2 - 4AC

Step 5: Initialize variables x1, x2, t, t1, and i

Step 6: If d > 0 (Real roots):

a. Calculate x1 and x2 using the quadratic formula

b. Display the roots: X1 and X2

Step 7: If d <= 0 (Imaginary roots):

a. Calculate the imaginary part (i) and the real part (t)

b. Round the imaginary part to two decimal places

c. Display the roots in the form X1 + iY and X2 - iY

Step 8: End of algorithm

```java
import java.util.*;
public class quadSolver
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Quadratic Form: ");
        System.out.println("Ax^2 + Bx + C");
        System.out.println("");
        System.out.println("Enter A: ");
        double a=sc.nextDouble();
        System.out.println("Enter B: ");
        double b=sc.nextDouble();
        System.out.println("Enter C: ");
        double c=sc.nextDouble();
        double b2=b*b;
        System.out.println("");
        System.out.println("Your Equation is: ");
        System.out.println((int)a+"x^2 + " +(int)b+"x + "+ (int)c +" = 0");
        System.out.println("");
        double d=b2-(4*a*c);
        double x1=0,x2=0,t=0,t1=0,i=0;
        String xi1="",xi2="";
        if(d>0)//Real
        {
            t=-b + Math.sqrt(d);
            x1=t/2*a;
            t1=-b - Math.sqrt(d);
            x2=t1/2*a;
            System.out.println("Roots of the Equation are: ");
            System.out.println("X1: "+x1+"\n"+"X2: "+x2);
        }
        else//Imaginary
        {
            i=Math.sqrt(d*-1);
            i=i/2*a;
            t=-b/2*a;
            i=(int)(i*100);
            i=i/100;
            //i=Math.round(i*100)/100;
            System.out.println("Roots of the Equation are: ");
            if(i==0.0)
            {
                System.out.print("X1 and X2: "+-b);
            }
            else
            {
                System.out.print("X1: "+t);
                System.out.println(" + "+"i"+i);
                System.out.print("X2: "+t);
                System.out.println(" - "+"i"+i);
            }
        }
    }
}
```

```
Quadratic Form:
Ax^2 + Bx + C

Enter A:
1
Enter B:
1
Enter C:
-12

Your Equation is:
1x^2 + 1x + -12 = 0

Roots of the Equation are:
X1: 3.0
X2: -4.0
```

Step 1:

Start of the algorithm

Step 2:

Define a class named Stack

Step 3:

Declare an integer array arr[] to store stack elements

Step 4:

Declare an integer variable top to track the top of the stack

Step 5:

Declare an integer variable size and initialize it to 10 (size of the stack)

Step 6:

Define a constructor Stack() for initializing the stack

- Step 6.1: Initialize the array arr with size size
- Step 6.2: Initialize top to -1

Step 7:

Define a method void push(int num) to add an element to the stack

- Step 7.1: Check if top is equal to size - 1
  - If true, print "Stack OVERFLOW"
  - Else, increment top and store num in arr[top]

Step 8:

Define a method int pop() to remove and return the top element of the stack

- Step 8.1: Check if top is equal to -1
  - If true, print "Stack UNDERFLOW" and return 0
  - Else, store the value of arr[top] in a variable n, decrement top, and return n

Step 9:

Define a method void display() to display all elements of the stack

- Step 9.1: Iterate from top to 0 and print each element arr[i]

Step 10: End of the algorithm

```
class Stack
{
    int arr[];
    int top;
    int size = 10;
    Stack()
    {
        arr = new int[size];
        top = -1;
    }
    void push(int num)
    {
        if (top == size - 1)
            System.out.println("Stack OVERFLOW");
        else
            arr[++top] = num;
    }
    int pop()
    {
        if (top == -1)
        {
            System.out.println("Stack UNDERFLOW");
            return 0;
        }
        else
        {
            int n = arr[top];
            top--;
            return n;
        }
    }
    void display()
    {
        for (int i = top; i <= 0; i++)
            System.out.println(arr[i]);
    }
}
```

```
Stack is:
12
33
222
124
23
1
```

Step 1:

Start of the algorithm

Step 2:

Define a class named Queue

Step 3:

Declare an integer variable size and initialize it to 10 (size of the queue)

Step 4:

Declare integer variables front and rear and initialize them to -1 (to track the front and rear of the queue)

Step 5:

Declare an integer array que[] to store queue elements

Step 6:

Define a constructor Queue() for initializing the queue

Step 6.1: Initialize the array que with size size

Step 7:

Define a method void insert(int x) to add an element to the queue

Step 7.1: Check if rear is equal to size - 1

If true, print "Queue Overflow"

Else, check if front is -1

If true, increment front and rear, and store x in que[rear]

Else, increment rear and store x in que[rear]

Step 8:

Define a method int delete() to remove and return the front element of the queue

Step 9:

Define a method void display() to display all elements of the queue

Step 10:

End of the algorithm

```java
class Queue
{
    int size = 10;
    int front = -1;
    int rear = -1;
    int que[];
    public Queue()
    {
        que = new int[size];
    }
    public void insert(int x)
    {
        if(rear == size-1)
        {
            System.out.println("Queue Overflow");
        }
        else if(front==-1)
        {
            front++;
            rear++;
            que[rear] = x;
        }
        else
        {
            rear++;
            que[rear] = x;
        }
    }
    public int delete()
    {
        if(front == -1 && rear == -1)
        {
            System.out.println("Queue Underflow");
            return -9999;
        }
        else if(front==rear)
        {
            int val = que[front];
            front--;
            rear--;
            return val; //return the deleted value
        }
        else
        {
            int temp = que[front];
            front++;
            return temp;
        }
    }
    public void display()
    {
        if(front==-1 && rear == -1)
            System.out.println("Queue is Empty");
        else
        {
            System.out.print("\nQueue is: ");
            for(int i=front; i<=rear; i++)
            {
                System.out.print(que[i] + " ");
            }
            System.out.println();
        }
    }
}
```

```
Queue is: 3 3 34 33
```

```
Queue is: 34 33
```

Step 1:

Start of the algorithm

Step 2:

Define a class named Strange

Step 3:

Declare an integer array ele[] to store elements

Step 4:

Declare an integer variable top to track the top of the stack

Step 5:

Declare an integer variable capacity and initialize it to 10 (default capacity of the stack)

Step 6:

Define a constructor Strange(int cap) for initializing the stack with a specified capacity

Step 6.1: Set capacity to the provided cap

Step 6.2: Initialize the array ele with size capacity

Step 6.3: Initialize top to -1

Step 7:

Define a method void pushItem(int value) to add an element to the stack

Step 8:

Define a method int popItem() to remove and return the top element of the stack

Step 8.1: Check if top is equal to -1

If true, print "Strange is empty. Returning -9999" and return -9999

Else, store the value of ele[top] in a variable n, decrement top, and return n

Step 9:

Define a method void display() to display all elements of the stack

Step 9.1: Print "Strange is:"

Step 9.2: Iterate from top to 0 and print each element ele[i]

Step 10:

End of the algorithm

```
class Strange
{
    int ele[];
    int top;
    int capacity = 10;
    Strange(int cap)
    {
        capacity=cap;
        ele = new int[capacity];
        top = -1;
    }
    void pushItem(int value)
    {
        if (top == capacity - 1)
            System.out.println("Strange is Full");
        else
            ele[++top] = value;
    }
    int popItem()
    {
        if (top == -1)
        {
            System.out.println("Strange is empty. Returning -9999");
            return -9999;
        }
        else
        {
            int n = ele[top];
            top--;
            return n;
        }
    }
    void display()
    {
        System.out.println("Strange is: ");
        for (int i = top; i >= 0; i--)
        {
            System.out.println(ele[i]);
        }
    }
}
```

```
Strange is:
10
8
9
6
7
5
3
4
2
1
```

Step 1:

Start of the algorithm

Step 2:

Define a class named RingGame

Step 3:

Declare an integer array ring[] to store rings

Step 4:

Declare an integer variable upper to track the top of the ring stack

Step 5:

Declare an integer variable max to store the maximum capacity of the ring stack

Step 6:

Define a constructor RingGame(int m) for initializing the ring stack with a specified capacity

Step 6.1: Set max to the provided m

Step 6.2: Initialize the array ring with size max

Step 6.3: Initialize upper to -1

Step 7:

Define a method void jump_in(int num) to add a ring to the stack

Step 7.1: Check if upper is equal to max - 1

If true, print "Column is Full. Start removing Rings."

Else, increment upper and store num in ring[upper]

Step 8:

Define a method int jump_out() to remove and return the top ring of the stack

Step 8.1: Check if upper is equal to -1

If true, print "Congratulations. The game is over." and return 0

Else, store the value of ring[upper] in a variable n, decrement upper, and return n

Step 9:

Define a method void display() to display all rings in the stack

Step 9.1: Print "Rings are:"

Step 9.2: Iterate from upper to 0 and print each element ring[i]

Step 10:

End of the algorithm

```java
class RingGame
{
    int ring[];
    int upper;
    int max;
    RingGame(int m)
    {
        max = m;
        ring = new int[max];
        upper = -1;
    }
    void jump_in(int num)
    {
        if (upper == max - 1)
            System.out.println("Coulmn is Full. Start removing Rings.");
        else
            ring[++upper] = num;
    }
    int jump_out()
    {
        if (upper == -1)
        {
            System.out.println("Congratulations. The game is over.");
            return 0;
        }
        else
        {
            int n = ring[upper];
            upper--;
            return n;
        }
    }
    void display()
    {
        System.out.println("Rings are: ");
        for (int i = upper; i >= 0; i--)
        {
            System.out.println(ring[i]);
        }
    }
}
```

```
Coulmn is Full. Start removing Rings.
Rings are:
34
32
31
Congratulations. The game is over.
Congratulations. The game is over.
```

Step 1:

Start of the algorithm

Step 2:

Define a class named Dequeue

Step 3:

Declare an integer array qrr[] to store elements

Step 4:

Declare integer variables lim, front, and rear to track the limits and the front and rear positions of the deque

Step 5:

Define a constructor Dequeue(int l) for initializing the deque with a specified limit

Step 5.1: Set lim to the provided l

Step 5.2: Initialize the array qrr with size lim

Step 5.3: Initialize front and rear to 0

Step 6:

Define a method void addFront(int v) to add an element to the front of the deque

Step 6.1: Check if front is equal to 0

If true, print "OVERFLOW FROM FRONT"

Else, decrement front and store v in qrr[front]

Step 7:

Define a method void addRear(int v) to add an element to the rear of the deque

Step 7.1: Check if rear is equal to lim

If true, print "OVERFLOW FROM REAR"

Else, store v in qrr[rear] and increment rear

Step 8:

Define a method int popFront() to remove and return the front element of the deque

Step 8.1: Check if front is less than rear

If true, store the value of qrr[front] in a variable d, increment front

If front is equal to rear, reset front and rear to 0

Return d

Else, return -999

Step 9:

Define a method int popRear() to remove and return the rear element of the deque

Step 9.1: Check if front is less than rear

If true, decrement rear, store the value of qrr[rear] in a variable d

If front is equal to rear, reset front and rear to 0

Return d

Else, return -999

Step 10:

Define a method void show() to display all elements of the deque

Step 10.1: Check if front is greater than or equal to rear

If true, print "DE-QUEUE EMPTY"

Else, iterate from front to rear and print each element qrr[i]

Step 11:

Define the main method to test the deque operations

Step 11.1: Create a Scanner object for input

Step 11.2: Prompt the user to enter the deque limit and read the input

Step 11.3: Create a Dequeue object with the provided limit

Step 11.4: Use a while(true) loop to present a menu and perform operations based on user input

Step 11.4.1: Display the menu options

Step 11.4.2: Read the user's choice

Step 11.4.3: Use a switch statement to execute the corresponding operation

Case 1: Prompt the user for an element and call addFront

Case 2: Prompt the user for an element and call addRear

Case 3: Call popFront and display the result

Case 4: Call popRear and display the result

Case 5: Call show

Default: Print "Bye!" and exit the loop

Step 12:

End of the algorithm

```
import java.util.*;
class Dequeue{
    int qrr[];
    int lim;
    int front;
    int rear;
    public Dequeue(int l){
        lim = l;
        qrr = new int[lim];
        front = 0;
        rear = 0;
    }
    public void addFront(int v){
        if(front == 0)
            System.out.println("OVERFLOW FROM FRONT");
        else
            qrr[--front] = v;
    }
    public void addRear(int v){
        if(rear == lim)
            System.out.println("OVERFLOW FROM REAR");
        else
            qrr[rear++] = v;
    }
    public int popFront(){
        if(front < rear){
            int d = qrr[front++];
            if(front == rear){
                front = 0;
                rear = 0;
            }
            return d;
        }
        return -999;
    }
    public int popRear(){
        if(front < rear){
            int d = qrr[--rear];
            if(front == rear){
                front = 0;
                rear = 0;
            }
            return d;
        }
        return -999;
    }
    public void show(){
        if(front > rear)
            System.out.println("DE-QUEUE EMPTY");
        else{
```

```
            for(int i = front; i < rear; i++)
                System.out.print(qrr[i] + " ");
            System.out.println();
        }
    }
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        System.out.print("Dequeue limit: ");
        int size = Integer.parseInt(in.nextLine());
        Dequeue dq = new Dequeue(size);
        while(true)
        {
            System.out.println("1. Add from front");
            System.out.println("2. Add from rear");
            System.out.println("3. Pop from front");
            System.out.println("4. Pop from rear");
            System.out.println("5. Display elements");
            System.out.print("Enter your choice: ");
            int choice = Integer.parseInt(in.nextLine());
            switch(choice)
            {
            case 1:
                System.out.print("Element to be added: ");
                int v = Integer.parseInt(in.nextLine());
                dq.addFront(v);
                break;
            case 2:
                System.out.print("Element to be added: ");
                v = Integer.parseInt(in.nextLine());
                dq.addRear(v);
                break;
            case 3:
                v = dq.popFront();
                if(v == -999)
                    System.out.println("Underflow from front");
                else
                    System.out.println(v + " popped");
                break;
            case 4:
                v = dq.popRear();
                if(v == -999)
                    System.out.println("Underflow from rear");
                else
                    System.out.println(v + " popped");
                break;
            case 5:
                dq.show();
                break;
            default:
                System.out.println("Bye!");
                return;
            }
        }
    }
}
```

Super Class: Rev

Step 1:

Start of the algorithm

Step 2:

Define a class named Rev

Step 3:

Define a method int rev(int n) to reverse the digits of an integer

Step 3.1: Initialize an integer variable r to 0

Step 3.2: Start a while loop that runs while n is greater than 0

Step 3.2.1: Calculate q as n % 10 (extract the last digit)

Step 3.2.2: Update r to r * 10 + q (build the reversed number)

Step 3.2.3: Update n to n / 10 (remove the last digit)

Step 3.3: Return r (the reversed number)

Step 4:

End of the superclass algorithm

Child Class: Palin

Step 5:

Define a class named Palin that extends Rev

Step 6:

Define the main method

Step 6.1: Create a Scanner object for input

Step 6.2: Prompt the user to enter a number

Step 6.3: Read the input number and store it in an integer variable nu

Step 6.4: Create an instance of Palin named ob

Step 6.5: Call the rev method from the superclass Rev using ob and store the result in an integer variable ni

Step 6.6: Compare nu with ni

If nu is equal to ni, print "It is Palindrome"

Else, print "It is not Palindrome"

Step 7:

End of the child class algorithm

```java
import java.util.*;
public class Rev
{
    int rev(int n)
    {
        int r=0;
        while(n>0)
        {
            int q=n%10;
            r=r*10+q;
            n=n/10;
        }
        return r;
    }
}
```



```java
import java.util.*;
public class Palin extends Rev
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter a number: ");
        int nu=sc.nextInt();
        Palin ob=new Palin();
        int ni=ob.rev(nu);
        if(nu==ni)
        {
            System.out.println("It is Palindrome");
        }
        else
        {
            System.out.println("It is not Palindrome");
        }
    }
}
```

```
Enter a number:
23432
It is Palindrome
Enter a number:
26963
It is not Palindrome
Enter a number:
67976
It is Palindrome
```

Super Class: Student

Step 1:

Start of the algorithm

Step 2:

Define a class named Student

Step 3:

Declare string variables name and dob to store the student's name and date of birth

Step 4:

Declare an integer variable r to store the student's roll number

Step 5:

Define a method void inputdata() to input the student's data

- Step 5.1: Create a Scanner object for input
- Step 5.2: Prompt the user to enter the name, date of birth, and roll number
- Step 5.3: Read the name using nextLine()
- Step 5.4: Read the date of birth using nextLine()
- Step 5.5: Read the roll number using nextInt()

Step 6:

Define a method void printdata() to print the student's data

- Step 6.1: Print the student's name
- Step 6.2: Print the student's date of birth
- Step 6.3: Print the student's roll number

Step 7:

End of the superclass algorithm

Child Class: Marks

Step 8:

Define a class named Marks that extends Student

Step 9:

Declare integer variables p, c, m, cts, and e to store marks in Physics, Chemistry, Mathematics, Computer Science, and English respectively

Step 10:

Declare float variables per and tot to store the percentage and total marks, initialized to 0

Step 11:

Declare a character variable gd to store the grade

Step 12:

Define a method void readdata() to input the student's marks

Step 12.1: Create a Scanner object for input

Step 12.2: Prompt the user to enter marks in Physics, Chemistry, Mathematics, Computer Science, and English

Step 12.3: Read the marks for each subject using nextInt()

Step 13:

Define a method void compute() to calculate the total marks, percentage, and grade

Step 13.1: Calculate tot as the sum of marks in all subjects (p, c, m, cts, and e)

Step 13.2: Calculate per as (tot * 100) / 500

Step 13.3: Determine the grade based on the percentage:

If per >= 90, set gd to 'A'

Else if per >= 60 and per < 90, set gd to 'B'

Else if per >= 40 and per < 60, set gd to 'C'

Else if per < 40, set gd to 'D'

Step 14:

Define a method void showdata() to display the student's data and marks

Step 14.1: Call printdata() to display the student's name, date of birth, and roll number

Step 14.2: Print the marks in Physics

Step 14.3: Print the marks in Chemistry

Step 14.4: Print the marks in Mathematics

Step 14.5: Print the marks in Computer Science

Step 14.6: Print the marks in English

Step 14.7: Print the percentage marks

Step 14.8: Print the grade

Step 15:

End of the child class algorithm

```java
import java.util.*;
class Student
{
    String name, dob;
    int r;
    void inputdata()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter name, date of birth and roll number");
        name = sc.nextLine();
        dob = sc.nextLine();
        r = sc.nextInt();
    }
    void printdata()
    {
        System.out.println("Name: "+name);
        System.out.println("Date of Birth: "+dob);
        System.out.println("Roll No.: "+r);
    }
}
```



```java
import java.util.*;
class Marks extends Student
{
    int p,c,m,cts,e;
    float per=0,tot=0;
    char gd;
    void readdata()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter marks in Physics, Chem, Math, Computer Science and
English");
        p = sc.nextInt();
        c = sc.nextInt();
        m = sc.nextInt();
        cts = sc.nextInt();
        e = sc.nextInt();
    }
    void compute()
    {
        tot = p+c+m+cts+e;
        per = (tot*100)/500;
        if(per>=90) gd='A';
        else if(per>=60 && per<90) gd='B';
        else if(per>=40 && per<60) gd='C';
        else if(per<40) gd='D';
    }
    void showdata()
    {
        printdata();
        System.out.println("Marks in Physics: "+p);
        System.out.println("Marks in Chemistry: "+c);
        System.out.println("Marks in Maths: "+m);
        System.out.println("Marks in Computer Science: "+cts);
        System.out.println("Marks in English: "+e);
        System.out.println("Percentage marks: "+per);
        System.out.println("Grade: "+gd);
    }
}
```

```
Name: Aaryan
Date of Birth: 14 Dec 2006
Roll No.: 2
Marks in Physics: 36
Marks in Chemistry: 26
Marks in Maths: 37
Marks in Computer Science: 40
Marks in English: 33
Percentage marks: 0.0
Grade: D
```

Super Class: ISC_Scores

Step 1:

Start of the algorithm

Step 2:

Define a class named ISC_Scores

Step 3:

Declare a protected 2D integer array number[][] of size 6x2 to store subject codes and marks

Step 4:

Define a method void getiscscores() to input the subject codes and marks

- Step 4.1: Create a Scanner object for input
- Step 4.2: Print "Enter subject codes and marks in 6 subjects:"
- Step 4.3: Use a for loop to iterate over 6 subjects
- Step 4.3.1: Read the subject code and store it in number[i][0]
- Step 4.3.2: Read the marks and store it in number[i][1]

Step 5:

Define a method int point(int sub_score) to calculate the points based on the subject score

- Step 5.1: Initialize an integer variable pt to 0
- Step 5.2: Use a for loop to iterate from 0 to 6
  - Step 5.2.1: If sub_score is greater than or equal to 100 - i * 10
  - Step 5.2.1.1: Set pt to i
  - Step 5.2.1.2: Break the loop
- Step 5.3: Return pt

Step 6:

End of the superclass algorithm

Child Class: BestFour

Step 7:

Define a class named BestFour that extends ISC_Scores

Step 8:

Declare integer variables t, temp, and c, initialized to 0, 0, and 2 respectively

Step 9:

Define a method void bestsubject() to calculate the best four subjects based on their scores

Step 9.1: Use a for loop to iterate over 6 subjects

Step 9.1.1: Add the points of number[i][1] to t using the point method

Step 9.2: Print "Total points obtained: " followed by t

Step 9.3: Use a nested for loop to sort the subjects based on their scores in descending order

Step 9.3.1: Outer loop from 0 to 6-1

Step 9.3.2: Inner loop from 0 to 6-1-i

Step 9.3.2.1: If number[j][1] is less than number[j+1][1]

Step 9.3.2.1.1: Use another loop from 0 to c to swap the elements

Step 9.3.2.1.1.1: Swap number[j][k] and number[j+1][k] using temp

Step 9.4: Print "Subject code with 4 best scores: "

Step 9.5: Use a for loop to print the subject codes and marks for the top 4 subjects

Step 10:

End of the child class algorithm

```java
import java.util.*;
class ISC_Scores
{
    protected int number[][] = new int[6][2];
    void getiscscores()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter subject codes and marks in 6 subjects:");
        for(int i=0;i<6;i++)
        {
            number[i][0]= sc.nextInt();
            number[i][1]= sc.nextInt();
        }
    }
    int point(int sub_score)
    {
        int pt=0;
        for(int i=0;i<=6;i++)
        {
            if(sub_score>=(100-i*10))
             {
               pt=i;
               break;
            }
        }
        return(pt);
    }
}
```



```java
class BestFour extends ISC_Scores
{
    int t=0, temp, c=2;
    void bestsubject()
    {
        for(int i=0;i<6;i++)
        {
            t = t+point(number[i][1]);
        }
        System.out.println("Total points obtained: ");
        System.out.println(t);
        for(int i=0;i<6-1;i++)
        {
            for(int j=0;j<(6-1-i);j++)
            {
                if(number[j][1]<number[j+1][1])
                {
                    for(int k=0;k<c;k++)
                    {
                        temp = number[j][k];
                        number[j][k]=number[j+1][k];
                        number[j+1][k]=temp;
                    }
                }
            }
        }
        System.out.println("Subject code with 4 best scores: ");
        for(int i=0;i<4;i++) System.out.println(number[i][0]+"\t\t"+number[i][1]);
    }
}
```

```
Total points obtained:
9
Subject code with 4 best scores:
4                   88
6                   79
3                   66
5                   37
```

Super Class: Vehicle

Step 1:

Start of the algorithm

Step 2:

Define a class named Vehicle

Step 3:

Declare string variable brand to store the brand of the vehicle

Step 4:

Declare an integer variable year to store the manufacturing year of the vehicle

Step 5:

Define a method void inputData(String b, int y) to input the vehicle's data

Step 5.1: Assign b to the brand variable

Step 5.2: Assign y to the year variable

Step 6:

Define a method void displayData() to display the vehicle's data

Step 6.1: Print the vehicle's brand

Step 6.2: Print the vehicle's manufacturing year

Step 7:

End of the superclass algorithm

Child Class: Car

Step 8:

Define a class named Car that extends Vehicle

Step 9:

Declare an integer variable doors to store the number of doors in the car

Step 10:

Define a method void inputCarData(String b, int y, int d) to input the car's data

Step 10.1: Call inputData(b, y) from the superclass Vehicle to set the brand and year

Step 10.2: Assign d to the doors variable

Step 11:

Define a method void displayCarData() to display the car's data

Step 11.1: Call displayData() from the superclass Vehicle to display the brand and year

Step 11.2: Print the number of doors in the car

Step 12:

Define the main method

Step 12.1: Create an instance of Car named myCar

Step 12.2: Call inputCarData("Toyota", 2020, 4) on myCar to set the car's data

Step 12.3: Call displayCarData() on myCar to display the car's data

Step 13:

End of the child class algorithm

```java
class Vehicle
{
    String brand;
    int year;
    void inputData(String b, int y)
    {
        brand = b;
        year = y;
    }
    void displayData()
    {
        System.out.println("Brand: " + brand);
        System.out.println("Year: " + year);
    }
}
```



```java
import java.util.*;
class Car extends Vehicle
{
    int doors;
    void inputCarData(String b, int y, int d)
    {
        inputData(b, y);
        doors = d;
    }
    void displayCarData()
    {
        displayData();
        System.out.println("Number of doors: " + doors);
    }
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        Car c = new Car();
        System.out.println("Enter Car Brand: ");
        String cn=sc.next();
        System.out.println("Enter Year of Release: ");
        int y=sc.nextInt();
        System.out.println("Enter No. of Doors");
        int nd=sc.nextInt();
        c.inputCarData(cn, y, nd);
        c.displayCarData();
    }
}
```

```
Enter Car Brand:
Toyota
Enter Year of Release:
2020
Enter No. of Doors
4
Brand: Toyota
Year: 2020
Number of doors: 4
```