COMPUTER SCIENCE PROJECT 12th

Sri Pandey <srichapandey2003@gmail.com> Mon, Jan 1, 2024 at 6:14 PM
To: Sri Pandey <srichapandey2003@gmail.com>

COMPUTER APPLICATION PROJECT

ISC SESSION :- 2021 – 2022

BY :- SRICHA PANDEY

CLASS :- XII – 'B'

ROLL NO. :- 67

ADMISSION NO. :- 30016

ACKNOWLEDGEMENT

This project would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study.

First and foremost, I express utmost gratitude to our Computer Teacher, Mrs. S. Waqar whose inputs and encouragement has been my inspiration as I hurdle over the obstacles in the completion of this project.

I would specially like to thank our Principal Mrs. V. Eusebius, who gave me the golden opportunity to do this wonderful project.

I would also like to thank my parents and tutors who helped me a lot in finalizing this project within the limited time frame.

SRICHA PANDEY

INDEX

	store them in file names.txt.		
14.	**Read names of 5 students through keyword and store them in file names.txt.**		
15.	**Read rollno and marks of 5 students and write them on a file namely "stu.dat".**		
16.	**Read data from a binary file "stu.dat" that stores rollno and marks of 5 students.**		

PROGRAMS ON STRINGS

PROGRAM 1. **Write a program that encodes a word in PigLatin.**

ALGORITHM OF THE PROGRAM :-

STEP 1 : CREATE A CLASS .

STEP 2 : CREATE A METHOD.

STEP 3 : INPUT THE STRING.

STEP 4 : COUNT THE NUMBER OF VOWELS PRESENT IN THE STRING.

STEP 5 : ARRANGE THE LETTERS TO ENCODE THE STRING IN PIGLATIN.

STEP 6 : PRINT THE STRING IN PIGLATIN.

STEP 7 : EXIT.

PROGRAM :-

```
import java.util.*;
```

```
public class Piglatin
{
   public static void main(String[] args)
   {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter a word :- ");
      String word = sc.next().toUpperCase();
      int o = word.indexOf('O');
      int e = word.indexOf('E');
      int i = word.indexOf('I');
      int a = word.indexOf('A');
      int u = word.indexOf('U');
      if(a == -1)
      a = word.length();
      if(e == -1)
      e = word.length();
      if(i == -1)
      i = word.length();
      if(o == -1)
      o = word.length();
      if(u == -1)
      u = word.length();
      int min1 = (((a<e)?a:e)<i)?((a<e)?a:e):i;
      int min = (((min1<o)?min1:o)<u)?((min1<o)?min1:o):u;
      String w = word.substring(min)+word.substring(0,min)+"AY";
      System.out.println("Word in piglatin :- "+w);
   }
}
```

VARIABLE DESCRIPTION

NAME OF THE VARIABLE	*DATA TYPE*	*DESCRIPTION*
word	**STRING**	**Stores the entered word.**
o	**INTEGER**	**Stores the frequency of the letter 'O'.**
e	**INTEGER**	**Stores the frequency of the letter 'E'.**
i	**INTEGER**	**Stores the frequency of the letter 'I'.**
a	**INTEGER**	**Stores the frequency of the letter 'A'.**
u	**INTEGER**	**Stores the frequency of the letter 'U'.**
min	**INTEGER**	**Stores the minimum frequency of the letters.**
min1	**INTEGER**	**Stores the minimum frequency of the letters.**
w	**STRING**	**Stores the arranged PigLatin word.**

PROGRAM 2. **Write a program to accept a string and display :**

(i) The number of Lowercase Characters.

(ii) The number of Uppercase Characters.

(iii) The number of Special Characters.

(iv) The number of digits present in the string.

ALGORITHM OF THE PROGRAM :-

STEP 1 : CREATE A CLASS .

STEP 2 : CREATE A METHOD.

STEP 3 : INPUT THE STRING.

STEP 4 : COUNT THE NUMBER OF LOWERCASE CHARACTERS PRESENT IN THE STRING.

STEP 5 : COUNT THE NUMBER OF UPPERCASE CHARACTERS PRESENT IN THE STRING.

STEP 6 : COUNT THE NUMBER OF SPECIAL CHARACTERS PRESENT IN THE STRING.

STEP 7 : COUNT THE NUMBER OF DIGITS PRESENT IN THE STRING.

STEP 8 : DISPLAY THE CALCULATED CHARACTERS AND DIGITS.

STEP 9 : EXIT.

PROGRAM :-

```
import java.util.*;

public class Recordofstring

{

   public static void main(String args[])
```

```
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a sentence :-");
        String s = sc.nextLine();
        int u = 0, l = 0, sp = 0, d = 0;
        int len = s.length();
        for(int i=0; i<len; i++)
        {
            if(s.charAt(i) >= 'A'&& s.charAt(i) <= 'Z')
            u = u+1;
            else if(s.charAt(i) >= 'a'&& s.charAt(i) <= 'z')
            l = l+1;
            else if(s.charAt(i) >= '0' && s.charAt(i) <= '9')
            d = d+1;
            else
            sp = sp+1;
        }
        System.out.println("The number of special characters :- "+sp);
        System.out.println("The number of lowercase characters :- "+l);
        System.out.println("The number of uppercase characters :- "+u);
        System.out.println("The number of digits present in the string :- "+d);
    }
}
```

VARIABLE DESCRIPTION

NAME OF THE VARIABLE	*DATA TYPE*	*DESCRIPTION*
s	**STRING**	**Stores the entered sentence.**
u	**INTEGER**	**Stores the frequency of the uppercase characters.**
l	**INTEGER**	**Stores the frequency of the lowercase characters.**
sp	**INTEGER**	**Stores the frequency of the special characters.**
d	**INTEGER**	**Stores the frequency of the digits present.**
len	**INTEGER**	**Stores the length of the entered sentence.**
i	**INTEGER**	**Used in the for loop.**

PROGRAM 3. **Write a program that reads a string and a word separately. The program then finds the frequency of the word in the string.**

ALGORITHM OF THE PROGRAM :-

STEP 1 : CREATE A CLASS .

STEP 2 : CREATE A METHOD.

STEP 3 : INPUT THE STRING.

STEP 4 : INPUT THE WORD TO SEARCH FOR.

STEP 5 : FIND THE FREQUENCY OF THE ENTERED WORD IN THE ENTERED STRING.

STEP 6 : PRINT THE FERQUENCY.

STEP 7 : EXIT.

PROGRAM :-

```
import java.util.*;
public class Fequency
{
   public static void main(String args[])
   {
      Scanner sc = new Scanner(System.in);
      int length, count = 0;
      char tmp;
      String tmpW = "";
      System.out.println("Enter a string : ");
      String searchstring = sc.nextLine();
      searchstring = searchstring.trim() + " ";
      length = searchstring.length();
      System.out.println("Enter the word to search for : ");
      String searchword = sc.next();
      for(int i = 0; i < length; i++)
      {
         tmp = searchstring.charAt(i);
         if(tmp == ' ')
         {
            if(tmpW.compareToIgnoreCase(searchword) == 0)
            {
               count++;
            }
            tmpW = "";
         }
```

```
        else
        tmpW = tmpW + tmp;
      }
      System.out.println("Frequency of '"+searchword+"' is : "+count);
      sc.close();
   }
}
```

VARIABLE DESCRIPTION

NAME OF THE VARIABLE	*DATA TYPE*	*DESCRIPTION*
searchstring	**STRING**	**Stores the entered sentence.**
searchword	**STRING**	**Stores the entered word.**
tmpW	**STRING**	**Used to find the frequency of the entered word.**
tmp	**CHARACTER**	**Used to find the frequency of the entered word.**
count	**INTEGER**	**Stores the frequency of the entered word.**
length	**INTEGER**	**Stores the length of the entered sentence.**
i	**INTEGER**	**Used in the for loop.**

PROGRAM 4. **Write a program to accept a word and check if it is a Palindrome or not.**

ALGORITHM OF THE PROGRAM :-

STEP 1 : CREATE A CLASS .

STEP 2 : CREATE A METHOD.

STEP 3 : INPUT THE WORD.

STEP 4 : CHECK WHETHER THE ENTERED WORD IS A PALINDROME OR NOT.

STEP 5 : DISPLAY THE RESULT.

STEP 6 : EXIT.

PROGRAM :-

```
import java.util.*;
public class Palindrome
{
   public static void main(String args[])
   {
      Scanner sc = new Scanner(System.in);
      int length;
      String reverseString = "";
      System.out.println("Enter a word : ");
      String inputString = sc.nextLine();
      length = inputString.length();
      for(int i = length-1; i >= 0; i--)
      {
         reverseString = reverseString + inputString.charAt(i);
      }
```

```
    if(reverseString.equalsIgnoreCase(inputString))
    System.out.println("'"+inputString.toUpperCase()+"' is Palindrome.");
    else
    System.out.println("'"+inputString.toUpperCase()+"' is not Palindrome.");
    sc.close();
  }
}
```

VARIABLE DESCRIPTION

NAME OF THE VARIABLE	*DATA TYPE*	*DESCRIPTION*
inputString	**STRING**	**Stores the entered word.**
reverseString	**STRING**	**Stores the reverse word of the entered word.**
length	**INTEGER**	**Stores the length of the entered word.**
i	**INTEGER**	**Used in the for loop.**

PROGRAMS ON NUMBERS

PROGRAM 1. **Write a Menu driven program to perform the two operations :**

1. To find whether a number is a composite number.

2. To find the smallest digit in a number.

After making a selection, the program should ask the user to input a number and perform the requested operation. An appropriate message should be displayed for an invalid choice.

ALGORITHM OF THE PROGRAM :-

STEP 1 : CREATE A CLASS .

STEP 2 : CREATE A METHOD.

STEP 3 : DISPLAY THE MENU.

STEP 4 : INPUT THE SELECTED CHOICE.

STEP 5 : CREATE TWO CASES FOR THE TWO DIFFERENT CHOICES.

STEP 6 : INPUT THE NUMBER TO PERFORM THE SELECTED OPERATION.

STEP 7 : DEFINE CASE 1 TO CHECK WHETHER THE ENTERED NUMBER IS A COMPOSITE NUMBER OR NOT.

STEP 8 : DEFINE CASE 2 TO FIND THE SMALLEST DIGIT PRESENT IN THE ENTERED NUMBER.

STEP 9 : DISPLAY THE RESULT.

STEP 10 : EXIT.

PROGRAM :-

```
import java.util.*;
public class CompositeandSmallest
{
   public static void main()
   {
      Scanner sc = new Scanner(System.in);
      System.out.println("To find whether a number is a composite number press 1.");
      System.out.println("To find the smallest digit in a number press 2.");
      int c = sc.nextInt();
      switch(c)
      {
         case 1:{ System.out.print("Enter the number :- ");
         int count = 0;
         int n = sc.nextInt();
         for(int i = 1; i <= n; i++)
         {
            if(n%i == 0)
            count++;
         }
         if(count >= 3)
         System.out.println("Entered number is a composite number.....");
```

```
        else
        System.out.println("Entered number is not a composite number.....");}
        break;
        case 2 : System.out.print("Enter the number :- ");
        int n = sc.nextInt();
        int num = 0;
        int min = 0;
        int j = 10;
        while(n!=0)
        {
           num = n%10;
           min = Math.min(j,num);
           j = min;
           n = n/10;
        }
        System.out.println("The smallest digit in a number :- "+min);
        break;
        default : System.out.println("Entered the wrong choice.....");
      }
   }
}
```

VARIABLE DESCRIPTION

NAME OF THE VARIABLE	*DATA TYPE*	*DESCRIPTION*
c	**INTEGER**	**Stores the entered choice.**
count	**INTEGER**	**Stores the count of the specific condition .**
n	**INTEGER**	**Used to store the entered number.**
num	**INTEGER**	**Used in the while loop.**
min	**INTEGER**	**Stores the minimum value in the while loop.**
j	**INTEGER**	**Stores the value stored in the variable min.**

i	INTEGER	Used in the for loop.

PROGRAM 2. **Write a program to determine if an entered number is a Pronic number.**

ALGORITHM OF THE PROGRAM :-

STEP 1 : CREATE A CLASS .

STEP 2 : CREATE A METHOD.

STEP 3 : INPUT THE NUMBER.

STEP 4 : CHECK WHETHER THE ENTERED NUMBER IS A PRONIC NUMBER OR NOT.

STEP 5 : DISPLAY THE RESULT.

STEP 6 : EXIT.

PROGRAM :-

```
public class Pronic
{
   public static void main(String[] args)
   {
      System.out.print("Enter a number :- ");
      Scanner sc = new Scanner(System.in);
      int num = sc.nextInt();
      int j = -1;
      for(int i = 0; i < num; i++)
      {
         j = i+1;
         if((i*j)==num)
         {
            System.out.println(i+" * "+j+" = "+num);
```

```
            j = -1;
            break;
          }
        }
        if(j == -1)
        {
          System.out.println("Entered number is a pronic number.....");
        }
        else
        System.out.println("Entered number is not a pronic number.....");
    }
}
```

VARIABLE DESCRIPTION

NAME OF THE VARIABLE	*DATA TYPE*	*DESCRIPTION*
num	**INTEGER**	**Stores the entered number.**
j	**INTEGER**	**Stores the number (i+1).**
i	**INTEGER**	**Used in the for loop.**

PROGRAM 3. **Write a program to determine if an entered 10 - digit ISBN code is legal.**

ALGORITHM OF THE PROGRAM :-

STEP 1 : CREATE A CLASS .

STEP 2 : CREATE A METHOD.

STEP 3 : INPUT THE 10 - DIGIT ISBN CODE.

STEP 4 : USING WHILE LOOP CHECK WHETHER THE ENERTED ISBN CODE IS LEGAL OR NOT.

STEP 5 : DISPLAY THE RESULT.

STEP 6 : EXIT.

PROGRAM :-

```
import java.util.*;
class Isbn
{
   public static void main(String[] args)
   {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter the 10 - digit ISBN code :- ");
      long num  = sc.nextLong();
      long n = 0;
      long sum = 0;
      int m = 11;
      while(num != 0)
      {
         n = num%10;
         num = num/10;
         m--;
         sum += (n*m);
      }
      if((sum%11)==0)
      System.out.println("Legal ISBN");
      else
      System.out.println("Illegal ISBN");
   }
}
```

VARIABLE DESCRIPTION

NAME OF THE VARIABLE	*DATA TYPE*	*DESCRIPTION*
num	**LONG**	**Stores the entered 10 - digit ISBN code.**
n	**LONG**	**Used in the while loop to extract the number.**
sum	**LONG**	**Used in the while loop to store the sum of the numbers.**
m	**INTEGER**	**Used to run the while loop.**

PROGRAM 4. **Write a program to determine if an entered even number is a Goldbach number. An appropriate message should be displayed for an invalid input.**

ALGORITHM OF THE PROGRAM :-

STEP 1 : CREATE A CLASS .

STEP 2 : CREATE A METHOD.

STEP 3 : INPUT THE NUMBER.

STEP 4 : CHECK WHETHER THE ENTERED NUMBER IS A GOLDBACH NUMBER OR NOT.

STEP 5 : DISPLAY THE RESULT.

STEP 6 : EXIT.

PROGRAM :-

```
import java.util.*;
public class Goldbach
{
   public static void main(String[] args)
   {
```

```
Scanner sc = new Scanner(System.in);
System.out.println("Enter the number : ");
int n = sc.nextInt();
int po[] = new int[n/2];
int i,j,s = 0, c,k = 0, r;
if(n%2 == 0)
{
   for(i = 1; i <= n; i+=2)
   {
      c=0;
      for(j = 1; j <= i; j++)
      {
         if(i%j == 0)
         c++;
      }
      if(c==2)
      {
         po[k] = i;
         k++;
      }
   }
   for(i=0;i<n/2;i++)
   {
      r = po[i];
      for(j=i+1; j<n/2; j++)
      {
         s = r+po[j];
         if(s==n)
         {
            System.out.println("It is a Goldbach Number.");
```

```
                System.exit(0);
            }
            else
            s=r;
          }
        }
        System.out.println("It is not a Goldbach Number.");
      }
      else
      System.out.println("INVALID INPUT! Please input an even number.");
   }
}
```

VARIABLE DESCRIPTION

NAME OF THE VARIABLE	*DATA TYPE*	*DESCRIPTION*
n	**INTEGER**	**Stores the entered number.**
po[]	**INTEGER**	**An array that stores the element n/2.**
i	**INTEGER**	**Used in the for loop.**
j	**INTEGER**	**Used in the for loop.**
c	**INTEGER**	**It is used for counting.**
k	**INTEGER**	**It is used for checking.**
s	**INTEGER**	**Stores the value of r + po[j].**
r	**INTEGER**	**Stores the value of po[i].**

PROGRAMS ON RECURSION

PROGRAM 1. **Write a program that takes an integer value and converts it to equivalent binary number.**

ALGORITHM OF THE PROGRAM :-

STEP 1 : CREATE A CLASS .

STEP 2 : CREATE THE RECURSIVE METHOD TO PRINT THE BITS IN THE CORRECT ORDER.

STEP 3 : CREATE A METHOD TO CONVERT THE ENTERED INTEGER VALUE TO EQUIVALENT BINARY NUMBER AND ALSO RETURN THE BINARY NUMBER AS LONG.

STEP 4 : CREATE A METHOD TO DISPLAY THE RESULT.

STEP 5 : CREATE THE MAIN METHOD TO CALL ALL THE METHODS.

STEP 6 : EXIT.

PROGRAM :-

```
import java.util.*;
public class BinaryConverter
{
   int number;
   long binNumber;
   BinaryConverter(int n)
   {
      number = n;
      binNumber = convert(n);
   }
   long convert(int n)
   {
      if(n==0)return 0;
      return convert(n/2)*10 + (n%2);
   }
   void display()
   {
      System.out.println("Number : "+number);
      System.out.println("Binary Equivalent : "+binNumber);
   }
   public static void main(String args[])
   {
      Scanner sc =  new Scanner(System.in);
      System.out.print("Enter number to be converted into binary equivalent : ");
      int N = sc.nextInt();
      BinaryConverter binum = new BinaryConverter(N);
      binum.display();
   }
}
```

VARIABLE DESCRIPTION

NAME OF THE VARIABLE	*DATA TYPE*	*DESCRIPTION*
N	**INTEGER**	**Stores the entered integer value.**
number	**INTEGER**	**Stores the value of the variable 'n'.**
n	**INTEGER**	**Used in the methods.**
binNumber	**LONG**	**Stores the converted binary equivalent.**

PROGRAM 2. **Write a recursive program that takes an integer input N and computes an approximation using following recursive formula :**

f(N) = 1 if N = 0

= 1 + 1/f(-1) if N > 0

ALGORITHM OF THE PROGRAM :-

STEP 1 : CREATE A CLASS .

STEP 2 : CREATE THE RECURSIVE METHOD TO PRINT THE APPROXIMATION USING THE GIVEN RECURSIVE FORMULA.

STEP 3 : CREATE THE MAIN METHOD TO CALL THE RECURSIVE METHOD.

STEP 4 : USING THE MAIN METHOD INPUT THE INTEGER TO CALL THE RECURSIVE METHOD.

STEP 5 : DISPLAY THE RESULT.

STEP 6 : EXIT.

PROGRAM :-

```
import java.util.*;

class Recursion2

{

   public static double function(int n)

   {

      if(n==0) return 1;

      return 1.0 + 1.0/ function(n-1);

   }

   public static void main(String args[])
```

```
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter value of N to compute f(N) : ");
        int N = sc.nextInt();
        System.out.println("The value of f("+N+") is : "+function(N));
    }
}
```

VARIABLE DESCRIPTION

NAME OF THE VARIABLE	*DATA TYPE*	*DESCRIPTION*
N	**INTEGER**	**Stores the entered integer value.**
n	**INTEGER**	**Used in the recursive method.**

PROGRAM 3. **Write a program that takes a positive integer N as a command-line argument and prints out all partitions of N.**

ALGORITHM OF THE PROGRAM :-

STEP 1 : CREATE A CLASS .

STEP 2 : CREATE A METHOD TO SEND THE PARAMERTERS TO THE RECURSIVE METHOD.

STEP 3 : CREATE THE RECURSIVE METHOD TO FIND THE NUMBER OF PARTITIONS OF N AND ALSO TO REPRESENT THEM .

STEP 4 : CREATE THE MAIN METHOD TO CALL THE METHODS.

STEP 5 : USING THE MAIN METHOD INPUT THE INTEGER TO CALL THE METHODS.

STEP 6 : DISPLAY THE RESULT.

STEP 7 : EXIT.

PROGRAM :-

```
import java.util.*;
class Recursion3
{
   public static void Recursion3(int n)
   {
      Recursion3(n,n,"");
   }
   public static void Recursion3(int n, int max, String prefix)
   {
      if(n==0)
      {
         System.out.println(prefix);
         return;
      }
      for(int i = Math.min(max,n); i >= 1; i--)
      {
         Recursion3(n-i,i,prefix+" "+i);
      }
   }
   public static void main(String[] args)
   {
```

```
    Scanner sc =  new Scanner(System.in);
    System.out.print("Enter N to be partitioned : ");
    int N = sc.nextInt();
    System.out.println("All possible partitions of "+N+"are : ");
    Recursion3(N);
  }
}
```

VARIABLE DESCRIPTION

NAME OF THE VARIABLE	*DATA TYPE*	*DESCRIPTION*
N	**INTEGER**	**Stores the entered integer value.**
n	**INTEGER**	**Used in the methods.**
i	**INTEGER**	**Used in the for loop.**
max	**INTEGER**	**Used in the recursive method.**
prefix	**STRING**	**Used in the recursive method.**

PROGRAM 4. **Write a recursive function for printing fibonacci series up to 10 terms.**

ALGORITHM OF THE PROGRAM :-

STEP 1 : CREATE A CLASS .

STEP 2 : CREATE THE RECURSIVE METHOD TO PRINT THE FIBONACCI SERIES.

STEP 3 : CREATE THE MAIN METHOD TO CALL THE RECURSIVE METHOD.

STEP 4 : DISPLAY THE RESULT.

STEP 5 : EXIT.

PROGRAM :-

```
class fib
{
  int fibo(int n)
  {
    if(n==1)
```

```
        return 0;
        else if(n==2)
        return 1;
        else if(n > 2)
        return fibo(n-1) + fibo(n-2);
        else
        return - 1;
    }
    public static void main(String args[])
    {
        int i, term;
        fib obj1 = new fib();
        System.out.println("The fibonacci series up to 10 terms :- ");
        for(i = 1; i <= 10; ++i)
        {
            term = obj1.fibo(i);
            System.out.print(term+" ");
        }
    }
}
```

VARIABLE DESCRIPTION

NAME OF THE VARIABLE	*DATA TYPE*	*DESCRIPTION*
n	**INTEGER**	**Used in the recursive method.**
i	**INTEGER**	**Used in the for loop.**
term	**INTEGER**	**Stores the calculated fibonacci series.**

PROGRAMS ON OPERATIONS ON FILES

PROGRAM 1. **Read names of 5 students through keyword and store them in file names.txt.**

ALGORITHM OF THE PROGRAM :-

STEP 1 : CREATE A CLASS .

STEP 2 : USING THE MAIN METHOD, CREATE THE OBJECTS OF THE BUFFER CLASS

TO PERFORM THE TASKS.

STEP 3 : USING THE MAIN METHOD INPUT THE NAMES OF 5 STUDENTS.

STEP 4 : STORE THE NAMES IN THE FILE.

STEP 5 : EXIT.

PROGRAM :-

```
import java.io.*;
public class IO
{
   static String fileName = ("names.txt");
   static InputStreamReader isr = new InputStreamReader(System.in);
   static BufferedReader stdin = new BufferedReader(isr);
   public static void main(String args[])
   {
      try{
         FileWriter fw = new FileWriter(fileName);
         BufferedWriter bw = new BufferedWriter(fw);
         PrintWriter outFile = new PrintWriter(bw);
         for(int i = 0; i<5; i++)
         {
            System.out.print("Enter Name :- ");
            String name = stdin.readLine();
            outFile.println(name);
         }
         outFile.close();
      }
      catch(IOException e)
      {
         System.err.println(e);
      }
   }
```

}

VARIABLE DESCRIPTION

NAME OF THE VARIABLE	*DATA TYPE*	*DESCRIPTION*
name	**STRING**	**Stores the entered names.**
i	**INTEGER**	**Used in the for loop.**
fileName	**STRING**	**Static variable used to store the file name.**

PROGRAM 2. **Read names of 5 students through keyword and store them in file names.txt.**

ALGORITHM OF THE PROGRAM :-

STEP 1 : CREATE A CLASS .

STEP 2 : USING THE MAIN METHOD, CREATE THE OBJECTS OF THE BUFFER CLASS TO PERFORM THE TASK.

STEP 3 : USING THE BUFFER CLASS READ THE NAMES OF 5 STUDENTS STORED IN THE FILE.

STEP 4 : DISPLAY THE RESULTS.

STEP 5 : EXIT.

PROGRAM :-

```
import java.io.*;
class Example
{
   public static void main(String args[])throws IOException
   {
      FileReader file = new FileReader("names.txt");
```

```
    BufferedReader fileInput = new BufferedReader(file);
    String text;
    int i = 0;
    while((text = fileInput.readLine()) != null)
    {
       i++;
       System.out.print("Name "+i+" : ");
       System.out.println(text);
    }
    fileInput.close();
  }
}
```

VARIABLE DESCRIPTION

NAME OF THE VARIABLE	*DATA TYPE*	*DESCRIPTION*
text	**STRING**	**Used to store the names from the file.**
i	**INTEGER**	**Used in the while loop.**

PROGRAM 3. **Read rollno and marks of 5 students and write them on a file namely "stu.dat".**

ALGORITHM OF THE PROGRAM :-

STEP 1 : CREATE A CLASS .

STEP 2 : USING THE MAIN METHOD, CREATE THE OBJECTS OF THE BUFFER CLASS TO PERFORM THE TASKS.

STEP 3 : USING THE MAIN METHOD INPUT THE ROLLNO AND MARKS OF 5 STUDENTS.

STEP 4 : WRITE THEM ON THE FILE.

STEP 5 : EXIT.

PROGRAM :-

```
import java.io.*;
public class BinaryOutput
{
   static String fileName = "stu.dat";
   static InputStreamReader isr = new InputStreamReader(System.in);
   static BufferedReader stdin = new BufferedReader(isr);
   public static void main(String[] args)
   { try
      { int rno; float marks;
        FileOutputStream fw = new FileOutputStream(fileName);
        DataOutputStream dw = new DataOutputStream(fw);
        for(int i = 0; i<5; i++)
        {
           System.out.print("Enter Rollno : ");
           rno = Integer.parseInt(stdin.readLine());
           System.out.print("Enter Marks : ");
           marks = Float.parseFloat(stdin.readLine());
           dw.writeInt(rno);
```

```
            dw.writeFloat(marks);
          }
          dw.close();
          fw.close();
        }
        catch(IOException e)
        {
          System.err.println(e);
        }
    }
}
```

VARIABLE DESCRIPTION

NAME OF THE VARIABLE	*DATA TYPE*	*DESCRIPTION*
rno	**INTEGER**	**Stores the rollno of the students.**
i	**INTEGER**	**Used in the for loop.**
fileName	**STRING**	**Static variable used to store the file name.**
marks	**FLOAT**	**Stores the marks of the students.**

PROGRAM 4. **Read data from a binary file "stu.dat" that stores rollno and marks of 5 students.**

ALGORITHM OF THE PROGRAM :-

STEP 1 : CREATE A CLASS .

STEP 2 : USING THE MAIN METHOD, CREATE THE OBJECTS OF THE BUFFER CLASS TO PERFORM THE TASK.

STEP 3 : USING THE BUFFER CLASS READ THE ROLLNO AND MARKS OF 5 STUDENTS STORED IN THE FILE.

STEP 4 : DISPLAY THE RESULTS.

STEP 5 : EXIT.

PROGRAM :-

```
import java.io.*;
public class BinaryInput
{
   static String fileName = "Stu.dat";
   public static void main(String[] args)
   {
      boolean EOF = false;
      try{
         FileInputStream fr = new FileInputStream(fileName);
         DataInputStream dr = new DataInputStream(fr);
         while(!EOF)
         {
            try
            {
               int rno; float marks;
               rno = dr.readInt();
               System.out.println("Rollno : "+rno);
               marks = dr.readFloat();
               System.out.println("Marks : "+marks);
            }
            catch (EOFException el)
             {
               System.out.println("end of file");
               EOF = true;
             }
            catch (IOException e)
            {
               System.err.println(e);
            }
```

```
        }
      }
      catch (FileNotFoundException e)
      {
        System.out.println("File not found");
      }
    }
}
```

VARIABLE DESCRIPTION

NAME OF THE VARIABLE	*DATA TYPE*	*DESCRIPTION*
rno	**INTEGER**	**Stores the rollno of the students from the file.**
EOF	**BOOLEAN**	**Used to run the while loop.**
fileName	**STRING**	**Static variable used to store the file name.**
marks	**FLOAT**	**Stores the marks of the students from the file.**

PROGRAMS ON DATA STRUCTURES

PROGRAM 1. Write a program to insert a node in a sorted linked list.

ALGORITHM OF THE PROGRAM :-

STEP 1 : CREATE A CLASS .

STEP 2 : USING THE MAIN METHOD, CREATE THE OBJECTS OF THE BUFFER CLASS TO PERFORM THE TASK.

STEP 3 : USING THE BUFFER CLASS READ THE ROLLNO AND MARKS OF 5

STUDENTS STORED IN THE FILE.

STEP 4 : DISPLAY THE RESULTS.

STEP 5 : EXIT.

PROGRAM :-

```
class LinkedList
{
   Node head;  // head of list

   /* Linked list Node*/
   class Node
   {
      int data;
      Node next;
      Node(int d) {data = d; next = null; }
   }

   /* Inserts a new Node at front of the list. */
   public void push(int new_data)
   {
      /* 1 & 2: Allocate the Node &
               Put in the data*/
      Node new_node = new Node(new_data);

      /* 3. Make next of new Node as head */
      new_node.next = head;

      /* 4. Move the head to point to new Node */
      head = new_node;
   }
```

```
/* Inserts a new node after the given prev_node. */
public void insertAfter(Node prev_node, int new_data)
{
    /* 1. Check if the given Node is null */
    if (prev_node == null)
    {
        System.out.println("The given previous node cannot be null");
        return;
    }

    /* 2 & 3: Allocate the Node &
              Put in the data*/
    Node new_node = new Node(new_data);

    /* 4. Make next of new Node as next of prev_node */
    new_node.next = prev_node.next;

    /* 5. make next of prev_node as new_node */
    prev_node.next = new_node;
}

/* Appends a new node at the end.  This method is
   defined inside LinkedList class shown above */
public void append(int new_data)
{
    /* 1. Allocate the Node &
       2. Put in the data
       3. Set next as null */
    Node new_node = new Node(new_data);
```

```
    /* 4. If the Linked List is empty, then make the
          new node as head */
    if (head == null)
    {
        head = new Node(new_data);
        return;
    }

    /* 4. This new node is going to be the last node, so
          make next of it as null */
    new_node.next = null;

    /* 5. Else traverse till the last node */
    Node last = head;
    while (last.next != null)
        last = last.next;

    /* 6. Change the next of last node */
    last.next = new_node;
    return;
}

public void printList()
{
    Node tnode = head;
    while (tnode != null)
    {
        System.out.print(tnode.data+" ");
        tnode = tnode.next;
```

```
    }
  }

  public static void main(String[] args)
  {
    LinkedList llist = new LinkedList();

    // Insert 6.  So linked list becomes 6->NUllist
    llist.append(6);

    // Insert 7 at the beginning. So linked list becomes
    // 7->6->NUllist
    llist.push(7);

    // Insert 1 at the beginning. So linked list becomes
    // 1->7->6->NUllist
    llist.push(1);

    // Insert 4 at the end. So linked list becomes
    // 1->7->6->4->NUllist
    llist.append(4);

    // Insert 8, after 7. So linked list becomes
    // 1->7->8->6->4->NUllist
    llist.insertAfter(llist.head.next, 8);

    System.out.println("\nCreated Linked list is: ");
    llist.printList();
  }
}
```

PROGRAM 2:. Write a program to delete a node in a sorted linked list.

Algorithm

- **Create a class Node which has two attributes: data and next. Next is a pointer to the next node in the list.**
- **Create another class DeleteStart which has two attributes: head and tail.**
- **addNode() will add a new node to the list:**
  - **Create a new node.**
  - **It first checks, whether the head is equal to null which means the list is empty.**
  - **If the list is empty, both head and tail will point to a newly added node.**
  - **If the list is not empty, the new node will be added to end of the list such that tail's next will point to a newly added node. This new node will become the new tail of the list.**

a. deleteFromStart() will delete a node from the beginning of the list:

- **It first checks whether the head is null (empty list) then, display the message "List is empty" and return.**
- **If the list is not empty, it will check whether the list has only one node.**
- **If the list has only one node, it will set both head and tail to null.**
- **If the list has more than one node then, the head will point to the next node in the list and delete the old head node.**

a. display() will display the nodes present in the list:

- **Define a node current which will initially point to the head of the list.**
- **Traverse through the list till current points to null.**
- **Display each node by making current to point to node next to it in each iteration.**

program :

Program:

```
public class DeleteStart {


   //Represent a node of the singly linked list
```

```
class Node{
    int data;
    Node next;

    public Node(int data) {
        this.data = data;
        this.next = null;
    }
}

//Represent the head and tail of the singly linked list
public Node head = null;
public Node tail = null;

//addNode() will add a new node to the list
public void addNode(int data) {
    //Create a new node
    Node newNode = new Node(data);

    //Checks if the list is empty
    if(head == null) {
        //If list is empty, both head and tail will point to new node
        head = newNode;
        tail = newNode;
    }
    else {
        //newNode will be added after tail such that tail's next will point to newNode
        tail.next = newNode;
        //newNode will become new tail of the list
        tail = newNode;
```

```
    }
}

//deleteFromStart() will delete a node from the beginning of the list
public void deleteFromStart() {

    //Checks if the list is empty
    if(head == null) {
        System.out.println("List is empty");
        return;
    }
    else {
        //Checks whether the list contains only one node
        //If not, the head will point to next node in the list and tail will point to the new head.
        if(head != tail) {
            head = head.next;
        }
        //If the list contains only one node
        //then, it will remove it and both head and tail will point to null
        else {
            head = tail = null;
        }
    }
}

//display() will display all the nodes present in the list
public void display() {
    //Node current will point to head
    Node current = head;
    if(head == null) {
```

```
        System.out.println("List is empty");
        return;
    }
    while(current != null) {
        //Prints each node by incrementing pointer
        System.out.print(current.data + " ");
        current = current.next;
    }
    System.out.println();
}

public static void main(String[] args) {

    DeleteStart sList = new DeleteStart();

    //Adds data to the list
    sList.addNode(1);
    sList.addNode(2);
    sList.addNode(3);
    sList.addNode(4);

    //Printing original list
    System.out.println("Original List: ");
    sList.display();

    while(sList.head != null) {
        sList.deleteFromStart();
        //Printing updated list
        System.out.println("Updated List: ");
        sList.display();
```

```
    }
  }
}
```

PROGRAM 3. Write a program to traverse a list

Algorithm:

1. *ptr=start*
2. *repeat steps 3 and 4 untill ptr = NULL*
3. *print ptr.INFO*
4. *ptr-ptr.LINK*
5. *END.*

Program:

```
import java.io.*;
class Node
{
  protected int data;
  protected Node link;
  public Node()
  {
    link=null; data=0;
  }
  public Node(int d, Node n)
  {
    data=d; link=n;
  }
  public void setlink (Node n)
  {
    link=n;
  }
```

```
    public void setData(int d)
    {
        data=d;
    }
    public Node getlink()
{
    return link;
}
public int getData()
{
    return data;
}

}
class linkedList
{
    protected Node start;
    public linkedList()
    {
        start = null;
    }
    public boolean isEmpty()
    {
        return start == null;
    }
    public void Insert(int val)
    {
        Node nptr, ptr, save=null;
        nptr = new Node(val, null);
        boolean ins=false;
```

```
if (start==null)
start=nptr;
else if (val<=start.getData())
{
   nptr.setlink(start);
   start=nptr;
}
else {
   save=start;
   ptr=start.getlink();
   while(ptr!=null)
   {
      if(val>=save.getData()&&val<=ptr.getData())
      {
         save.setlink(nptr);
         nptr.setlink(ptr);
         ins=true;
         break;
      }
      else
      {
         save=ptr;
         ptr=ptr.getlink();
      }
   }
   if(ins==false)
   {
      save.setlink(nptr);
   }
}
```

```
    }
    public void Traverse()
    {
        Node ptr = start;
        System.out.print(start.getData()+" "+"-->");
        ptr = start.getlink();
        while(ptr.getlink()!= null)
        {
            System.out.print(ptr.getData()+" "+"-->");
            ptr = ptr.getlink();
        }
        System.out.print(ptr.getData()+" !!!!");
        System.out.println();
    }
}
    class linkListTest
    {
        protected static linkedList S;
        public static void main(String[] args)
        {
            int num;
            S = new linkedList();
            BufferedReader br = new BufferedReader(new InputStreamReader (System.in));
            System.out.println("..... Starting List Test for TRAVERSAL .....\n");
            for(int a = 0; a<7; ++a)
            {
                System.out.print("Enter a number : ");
                try
                {
```

```
            num = Integer.parseInt(br.readLine());
            S.Insert(num);
          }
          catch(Exception e)
          {
            System.out.println(e);
          }
        }
        System.out.println("\n Now the List is : ");
        S.Traverse();
        System.out.println("\n- - - - List Test Over - - - -");
    }
}
```

PROGRAM 4. Write a program to search for an item in a linked list.

Algorithm

- **Create a class Node which has two attributes: data and next. Next is a pointer to the next node in the list.**
- **Create another class SearchLinkedList which has two attributes: head and tail.**
- **addNode() will add a new node to the list:**
  - **Create a new node.**
  - **It first checks, whether the head is equal to null which means the list is empty.**
  - **If the list is empty, both head and tail will point to a newly added node.**
  - **If the list is not empty, the new node will be added to end of the list such that tail's next will point to a newly added node. This new node will become the new tail of the list.**

a. searchNode() will search for a node in the list:

- **Variable i will keep track of the position of the searched node.**
- **The variable flag will store boolean value false.**
- **Node current will point to head node.**
- **Iterate through the loop by incrementing current to current.next and i to i +**
- **Compare each node's data with the searched node if a match is found, set a flag to true.**
- **If the flag is true, display the position of the searched node.**
- **Else, display the message "Element is not present in the list".**

a. display() will display the nodes present in the list:

- **Define a node current which will initially point to head of the list.**
- **Traverse through the list till current points to null.**
- **Display each node by making current to point to node next to it in each iteration.**

Program:

```
public class SearchLinkedList {
   //Represent a node of the singly linked list
   class Node{
      int data;
      Node next;
      public Node(int data) {
         this.data = data;
         this.next = null;
      }
   }
   //Represent the head and tail of the singly linked list
   public Node head = null;
   public Node tail = null;
   //addNode() will add a new node to the list
   public void addNode(int data) {
```

```
    //Create a new node
    Node newNode = new Node(data);
    //Checks if the list is empty
    if(head == null) {
        //If list is empty, both head and tail will point to new node
        head = newNode;
        tail = newNode;
    }
    else {
        //newNode will be added after tail such that tail's next will point to newNode
        tail.next = newNode;
        //newNode will become new tail of the list
        tail = newNode;
    }
}
//searchNode() will search for a given node in the list
public void searchNode(int data) {
    Node current = head;
    int i = 1;
    boolean flag = false;
    //Checks whether list is empty
    if(head == null) {
        System.out.println("List is empty");
    }
    else {
        while(current != null) {
            //Compares node to be found with each node present in the list
            if(current.data == data) {
                flag = true;
                break;
```

```
            }
            i++;
            current = current.next;
        }
    }
    if(flag)
        System.out.println("Element is present in the list at the position : " + i);
    else
        System.out.println("Element is not present in the list");
}

public static void main(String[] args) {

    SearchLinkedList sList = new SearchLinkedList();

    //Add nodes to the list
    sList.addNode(1);
    sList.addNode(2);
    sList.addNode(3);
    sList.addNode(4);

    //Search for node 2 in the list
    sList.searchNode(2);
    //Search for a node  in the list
    sList.searchNode(7);
}}
```

PROGRAMS

ON

INHERITANCE

PROGRAM 1. Program to illustrate of **Multilevel Inheritance**

.

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class BabyDog extends Dog{
void weep(){System.out.println("weeping...");}
}
class TestInheritance2{
public static void main(String args[]){
BabyDog d=new BabyDog();
```

```
d.weep();
d.bark();
d.eat();
}}
```

PROGRAM 2. Program to show the working of inheritance.

```
class Teacher {
  private String designation = "Teacher";
  private String collegeName = "Beginnersbook";
  public String getDesignation() {
        return designation;
  }
  protected void setDesignation(String designation) {
        this.designation = designation;
  }
  protected String getCollegeName() {
        return collegeName;
  }
  protected void setCollegeName(String collegeName) {
        this.collegeName = collegeName;
  }
  void does(){
        System.out.println("Teaching");
  }
}

public class JavaExample extends Teacher{
  String mainSubject = "Physics";
  public static void main(String args[]){
        JavaExample obj = new JavaExample();
        /* Note: we are not accessing the data members
         * directly we are using public getter method
         * to access the private members of parent class
         */
        System.out.println(obj.getCollegeName());
        System.out.println(obj.getDesignation());
```

```
        System.out.println(obj.mainSubject);
        obj.does();
  }
}
```

PROGRAM 3. Getting access from a subclass method to a superclass method in the case when the names of the methods and the signatures of their parameters match. Keyword super.

```
// superclass
class A {
    void method() {
        // method of class A
        System.out.println("Method A");
    }
}

// subclass of class A
class B extends A {
    void method() {
        // class B method, overrides class A method
        System.out.println("Method B");
    }

    void method2() {
        System.out.println("Method2 - B");
        super.method(); // call to superclass A method
    }
}

public class Train01 {
    public static void main(String[] args) {
        B obj = new B(); // an instance of class B
        objB.method2();
    }
}
```

PROGRAM 4: **Method overriding in Java Inheritance**

```
class Parent {
```

```
    // private methods are not overridden
    private void m1()
    {
        System.out.println("From parent m1()");
    }

    protected void m2()
    {
        System.out.println("From parent m2()");
    }
}
  class Child extends Parent {
    // new m1() method
    // unique to Child class
    private void m1()
    {
        System.out.println("From child m1()");
    }

    // overriding method
    // with more accessibility
    @Override
    public void m2()
    {
```

```
        System.out.println("From child m2()");
    }
}

// Driver class
class Main {
    public static void main(String[] args)
    {
        Parent obj1 = new Parent();
        obj1.m2();
        Parent obj2 = new Child();
        obj2.m2();
    }
}
```

PROGRAMS

ON

2D - ARRAY

Program - 1 : Program of TwoDimensionalScanner using 2- D Array

Algorithm

1. **Read the row length, column length of an array using sc.nextInt() method of Scanner class.**

2) Declare the array with the dimension row, column.
3) for i=0 to i<row for j=0 to j<column sc.nextInt() reads the entered number and insert the element at a[i][j].

Program:

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

        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Row length of an array : ");
        int row=sc.nextInt();
        System.out.println("Enter column length of an array : ");
        int column=sc.nextInt();
        int a[][]=new int[row][column];//declaration
        System.out.print("Enter " + row*column + " Elements to Store in Array :\n");
      for (int i = 0; i < row; i++)
        {
          for(int j = 0; j < column; j++)
          {
         a[i][j] = sc.nextInt();
          }
        }
      System.out.print("Elements in Array are :\n");
      for (int i = 0; i < row; i++)
        {
          for(int j = 0; j < column; j++)
```

```
            {
              System.out.println("Row ["+i+"]:  Column ["+j+"] :"+a[i][j]);
            }
          }
    }
}
```

Program - 2 : Java Program to add two matrices

```
public class MatrixAdditionExample{
public static void main(String args[]){
//creating two matrices
int a[][]={{1,3,4},{2,4,3},{3,4,5}};
int b[][]={{1,3,4},{2,4,3},{1,2,4}};

//creating another matrix to store the sum of two matrices
int c[][]=new int[3][3];  //3 rows and 3 columns

//adding and printing addition of 2 matrices
for(int i=0;i<3;i++){
for(int j=0;j<3;j++){
c[i][j]=a[i][j]+b[i][j];    //use - for subtraction
System.out.print(c[i][j]+" ");
}
System.out.println();//new line
}
}}
```

Program - 3 : Java Program to multiply two matrices

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

```
//creating two matrices
int a[][]={{1,1,1},{2,2,2},{3,3,3}};
int b[][]={{1,1,1},{2,2,2},{3,3,3}};

//creating another matrix to store the multiplication of two matrices
int c[][]=new int[3][3];  //3 rows and 3 columns

//multiplying and printing multiplication of 2 matrices
for(int i=0;i<3;i++){
for(int j=0;j<3;j++){
c[i][j]=0;
for(int k=0;k<3;k++)
{
c[i][j]+=a[i][k]*b[k][j];
}//end of k loop
System.out.print(c[i][j]+" ");  //printing matrix element
}//end of j loop
System.out.println();//new line
}
}}
```

Program4:

Write a program in Java to enter natural numbers in a double dimensional array mxn (where m is the number of rows and n is the number of columns). Shift the elements of 4^{th} column into the 1^{st} column, the elements of 1^{st} column into the 2^{nd} column and so on. Display the new matrix.

```
import java.util.Scanner;

public class KboatSDAColShift
{
   public static void main(String args[]) {
      Scanner in = new Scanner(System.in);
      System.out.print("Enter number of rows (m): ");
      int m = in.nextInt();
```

```
        System.out.print("Enter number of columns (n): ");
        int n = in.nextInt();

        int arr[][] = new int[m][n];
        int newArr[][] = new int[m][n];

        System.out.println("Enter array elements");
        for (int i = 0; i < m; i++) {
            System.out.println("Enter Row "+ (i+1) + " :");
            for (int j = 0; j < n; j++) {
                arr[i][j] = in.nextInt();
            }
        }

        System.out.println("Input Array:");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(arr[i][j] + "\t");
            }
            System.out.println();
        }

        for (int j = 0; j < n; j++) {
            int col = j + 1;
            if (col == n) {
                col = 0;
            }
            for (int i = 0; i < m; i++) {
                newArr[i][col] = arr[i][j];
            }
        }

        System.out.println("New Shifted Array:");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(newArr[i][j] + "\t");
            }
            System.out.println();
        }
    }
}
```

MISCELLENEOUS

PROGRAMS

PROGRAM : 1 Java Coding to Get Current Date and Time

```
import java.time.format.DateTimeFormatter;
import java.time.LocalDateTime;
public class CurrentDateTimeExample1 {
  public static void main(String[] args) {
   DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
   LocalDateTime now = LocalDateTime.now();
   System.out.println(dtf.format(now));
  }
```

```
}
```

PROGRAM 2: Java Coding to Formatting Date and Time

```
import java.util.*;
import java.text.*;

public class DateDemo {

  public static void main(String args[]) {
    Date dNow = new Date( );
    SimpleDateFormat ft =
    new SimpleDateFormat ("E yyyy.MM.dd 'at' hh:mm:ss a zzz");

    System.out.println("Current Date: " + ft.format(dNow));
  }
}
```