Computer Project

Aanjishnu Bhattacharyya

23 June 2022

Table of Contents

Assignment 1

Write a program to accept a date in format dd/mm/yyyy and accept the name of the day on 1st January of the corresponding year. Find the day for the given date. Do validation check where required.
ex: input: date:5/7/2001
day on 1st January: Monday
output: day on 5/7/2001: Thursday

Algorithm

Date
Step 1: Start
Step 2: initialize name of week
Step 3: initialize number of days in a month
Step 4: initialize dd mm yy
Step 5: Accomodate for a Leapyear by adding one if a leapyear is encountered
Step 6: formats the string to be sensible for internal use
Step 7: End

generate_day
Step 1: Start
Step 2: genenrates the required date
Step 3: converting first_day_of_january from string to number
Step 4: loop through the days of week if it is found use the index
Step 5: if it is not foound bail out
Step 6: simple error handling for a malfromed input
Step 7: offsetting by day
Step 8: offsetting by month
Step 9: by looping through the days of the month
Step 10: returning a correctly formatted string for printing
Step 11: End

parse_input
Step 1: Start
Step 2: formatting accroding to dd/mm/yyyy
Step 3: End

Date
Step 1: Start
Step 2: manages main input and output
Step 3: End

main
Step 1: Start
Step 2: accept input properly
Step 3: System.out.print("day on 1st January: ");
Step 4: parsing the input into usable format
Step 5: Handles apparent error condition
Step 6: Handles apparent error condition
Step 7: outputs the required day
Step 8: End

Source Code

```
import java.util.Scanner;

public class Date
{
     // Stores the names of the days
     private String[] days_of_week = null;

     // number of days in a single month
     private int[] days_in_month = null;

     // Accepted from the user
     private int day;

     // Accepted from the user
     private int month;

     // Accepted from the user
     private int year;

     // First Day of January
     private String first_day_of_january;

     Date(int dd, int mm, int yyyy, String fdoj)
     {
          // initialize name of week
          this.days_of_week   =   new   String[]{"monday",   "tuesday",
"wednesday", "thursday", "friday", "saturday", "sunday"};

          // initialize number of days in a month
          this.days_in_month  =  new  int[]{31,  28,  31,  30,  31,  30,  31,
31, 30, 31, 30, 31};

          // initialize dd mm yy
          this.day = dd;
          this.month = mm;
          this.year = yyyy;

          // Accomodate for a Leapyear by adding one if a leapyear is
encountered
          if(yyyy % 4 == 0 && yyyy % 100 != 0 || yyyy % 400 == 0)
               this.days_in_month[1] += 1;

          // formats the string to be sensible for internal use
```

```
          this.first_day_of_january = fdoj.trim().toLowerCase();
     }

     String generate_day()
     {
          // genenrates the required date
          // converting first_day_of_january from string to number
          // loop through the days of week if it is found use the index
          // if it is not foound bail out

          // num first day of january falls on
          int num_fdoj = -1;
          for(int i = 0; i < this.days_of_week.length; i++)
          {
               if(this.days_of_week[i].equals(this.first_day_of_janu-
ary))
               {
                    num_fdoj = i;
                    break;
               }
          }

          // simple error handling for a malfromed input
          if(num_fdoj == -1)
               return null;

          // days from first january
          int days_from_1stjan = 0;

          // offsetting by day
          days_from_1stjan += this.day-1;

          // offsetting by month
          // by looping through the days of the month
          for(int i = 0; i < this.month-1; i++)
               days_from_1stjan += this.days_in_month[i];

          // correctly offsetting days of year with days of week and
converting to string using lookup table;
          String   new_day   =   this.days_of_week[(days_from_1stjan   +
num_fdoj)%this.days_of_week.length];

          // returning a correctly formatted string for printing
          return  (new_day.charAt(0)+"").toUpperCase()  +  new_day.sub-
string(1);
     }

     static int[] parse_input(String d)
     {
```

```
          // parses the string into a integer array delemetarized with
/
          String[] s = d.split("/");

          // initialize an int array
          int[]  di  =  new  int[]{Integer.parseInt(s[0]),  Integer.par-
seInt(s[1]), Integer.parseInt(s[2])};

          // formatting accroding to dd/mm/yyyy
          if(di[0] > 31 || di[0] < 1) return null;
          if(di[1] > 12 || di[1] < 1) return null;
          if(di[2] > 9999 || di[2] < 0) return null;

          return di;
     }

     // manages main input and output

}
import java.util.Scanner;
public class Date_main
{
     public static void main(String args[])
     {
          // input system
          Scanner sc = new Scanner(System.in);

          // accept input properly
          // System.out.print("date: ");
          String str_date = sc.next();

          // System.out.print("day on 1st January: ");
          // day of first january
          String jan_1st = sc.next();

          // parsing the input into usable format
          // date array
          int[] date = Date.parse_input(str_date);

          // Handles apparent error condition
          if(date == null)
          {
               System.err.println("Malformed Input");
               return;
          }

          // gnenrate object
          Date d = new Date(date[0], date[1], date[2], jan_1st);
```

```
          // get generated day
          String new_day = d.generate_day();

          // Handles apparent error condition
          if(new_day == null)
          {
               System.err.println("Input  is  either  out  of  bounds  or
nonsensical");
               return;
          }

          // outputs the required day
          System.out.println("day on "+str_date+": " + new_day);
     }
}
```

Varible Listing

Name	Function	Type	Scope
days_of_week	Stores the names of the days	String[]	Date
days_in_month	number of days in a single month	int[]	Date
day	Accepted from the user	int	Date
month	Accepted from the user	int	Date
year	Accepted from the user	int	Date
first_day_of_january	First Day of January	String	Date
num_fdoj	num first day of january falls on	int	generate_day
days_from_1stjan	days from first january	int	generate_day
new_day	correctly offsetting days of year with days of week and converting to string using lookup table;	String	generate_day
s	parses the string into a integer array delemetarized with /	String[]	parse_input
di	initialize an int array	int[]	parse_input

Assignment 2

Write a program to add two times given by the user in hour,min and seconds.

Class name: TimeAdd

Data member: hr(hour),min(minutes),sec(seconds)

Member methods:
TimeAdd(): DEFAULT constructor
void accept(): accept time from user
TimeAdd timeAdd(TimeAdd t): add two time objects return the final time value.
*

Algorithm

TimeAdd
Step 1: Start
Step 2: second
Step 3: initialize the time
Step 4: End

accept
Step 1: Start
Step 2: accept input from the user
Step 3: check if mins are between 0 and 60
Step 4: check if mins are between 0 and 60
Step 5: End

timeAdd
Step 1: Start
Step 2: add two time objects return final value
Step 3: generate a new timeadd object
Step 4: End

display
Step 1: Start
Step 2: display time in hour, mins and seconds
Step 3: End

main
Step 1: Start
Step 2: 3 value of Timeadd
Step 3: accept values for t's
Step 4: add time objects
Step 5: display the times
Step 6: End

Source Code

```
import java.util.Scanner;

public class TimeAdd
{
     int hr;        // hours
     int min;  // minuites
     int sec;  // second

     public TimeAdd()
     {
          // initialize the time
          this.hr = 0;
          this.min = 0;
          this.sec = 0;
     }

     void accept()
     {
          // input handler
          Scanner sc = new Scanner(System.in);

          // accept input from the user
          this.hr = sc.nextInt();
          this.min = sc.nextInt();
          this.sec = sc.nextInt();

          // check if mins are between 0 and 60
          if(this.min >= 60 || this.min < 0)
          {
               System.err.println("Minuites make no sense!");
               System.exit(1);
          }

          // check if mins are between 0 and 60
          if(this.sec >= 60 || this.sec < 0)
          {
               System.err.println("Seconds make no sense!");
               System.exit(1);
          }
     }

     TimeAdd timeAdd(TimeAdd t)
     {
```

```
          // add two time objects return final value
          // generate a new timeadd object
          // add and store the value of this and t
          TimeAdd x = new TimeAdd();
          x.hr = t.hr + this.hr;
          x.min = t.min + this.min;
          x.sec = t.sec + this.sec;
          return x;
     }

     void display()
     {
          // display time in hour, mins and seconds
          System.out.println("Hr:    "+this.hr+"Min:"+this.min+"    Sec:
"+this.sec);
     }


}
import java.util.Scanner;
public class TimeAdd_main
{
     public static void main(String args[])
     {
          // 3 value of Timeadd
          // Timeadds
          TimeAdd[]  t  =  new  TimeAdd[]{new  TimeAdd(),  new  TimeAdd(),
null};

          // accept values for t's
          t[0].accept();
          t[1].accept();

          // add time objects
          t[2] = t[0].timeAdd(t[1]);

          // display the times
          t[0].display();
          t[1].display();
          t[2].display();
     }
}
```

Varible Listing

Name	Function	Type	Scope
hr		int	TimeAdd
min	hours	int	TimeAdd
sec	minuites	int	TimeAdd
sc	input handler	Scanner	accept
x	add and store the value of this and t	TimeAdd	timeAdd

Assignment 3

Write a program to take lower and upper range from the user and print all the binodd numbers within that range. (A binodd number is a number whose binary equivalent have all the 1s present in the odd position of the binary number considering from MSB to LSB) Example: 17 is a binodd number as its binary equivalent is 10001 where 1s are in the position 1st and 5th position of the binary number which are odd position of the number.

Algorithm

isBinOdd
Step 1: Start
Step 2: loop until num not equals to zero
Step 3: two bits are checked
Step 4: if they are not 01 then its an anomly
Step 5: therefore return false
Step 6: otherwise return true
Step 7: End

main
Step 1: Start
Step 2: accpet upper limit
Step 3: accpet lower limit
Step 4: loop from lower limit to upper limit
Step 5: if any big odds are encountered print it
Step 6: End

Source Code

```
import java.util.Scanner ;

public class BinOdd
{
     static boolean isBinOdd(int num)
     {
          // loop until num not equals to zero
          // two bits are checked
          // if they are not 01 then its an anomly
          // therefore return false
          while(num != 0)
          {
               if((num & 3) != 1)
                    return false;
               num >>= 2;
          }

          // otherwise return true
          return true;
     }


}
import java.util.Scanner;
public class BinOdd_main
{
     public static void main(String args[])
     {
          // input handler
          Scanner sc = new Scanner(System.in);

          // accpet upper limit
          // upper limit
          int upper_limit = sc.nextInt();

          // accpet lower limit
          // lower limit
          int lower_limit = sc.nextInt();

          // loop from lower limit to upper limit
          // if any big odds are encountered print it
          while(lower_limit < upper_limit)
          {
```

```
                if(BinOdd.isBinOdd(lower_limit))
                     System.out.println(lower_limit);
                lower_limit++;
           }
     }
}
```

Varible Listing

Name	Function	Type	Scope

Assignment 4

Write a program to accept a square matrix CIR[][] of order MXM where M is no. of rows and no. of columns. Value of M varies from 2 <= M <= 10. Accept alphabet character values in UPPERCASE as input. Display appropriate mess for invalid input. Perform following tasks.

i) Display Original Matrix.
ii) Find the sum of Unicode values of the elements of four corners of the matrix.
iii) Rotate matrix 90 degrees anti-clockwise and display it

Example:

INPUT:

M = 3

A F D
D B T
C A A

OUTPUT:

Original Matrix:
A F D
D B T
C A A

Sum = 256

Final Matrix:
D T A
F B A
A D C

Algorithm

Matrix
Step 1: Start
Step 2: initializing M with M
Step 3: initializing mat of size M by M
Step 4: End

display
Step 1: Start
Step 2: loop accross the matrix elements
Step 3: print each element
Step 4: End

unicodeSum
Step 1: Start
Step 2: adding all the corners of the matrix
Step 3: returning the recorded sum
Step 4: End

rotateMat
Step 1: Start
Step 2: loop through the coloums from first to last
Step 3: loop through the rows from last to first
Step 4: rotate the mat matrix
Step 5: setting original matrix to rotated matrix
Step 6: End

readMat
Step 1: Start
Step 2: create input handler
Step 3: accept size as input
Step 4: loop through the matrix to generate the matrix by accepting input
Step 5: return the generated matrix
Step 6: End

main
Step 1: Start
Step 2: accpting input from user
Step 3: displaying original matrix
Step 4: displaying the sum of the corners
Step 5: roatate the actual matrix
Step 6: display the rotated matrix
Step 7: End

Source Code

```
import java.util.Scanner;

public class Matrix
{
     // stores the size of square matrix
     int M;

     // stores the matrix itself
     char[][] mat;

     Matrix(int M)
     {
          // initializing M with M
          this.M = M;

          // initializing mat of size M by M
          this.mat = new char[M][M];
     }

     void display()
     {
          // loop accross the matrix elements
          // print each element
          for(int i = 0; i < this.M; i++)
          {
               for(int j = 0; j < this.M; j++)
               {
                    System.out.print(this.mat[i][j]+" ");
               }
               System.out.println();
          }
     }

     int unicodeSum()
     {
          // stores the sum of the corners of the mat
          int sum = 0;

          // adding all the corners of the matrix
          sum += this.mat[0][0] + this.mat[M-1][0] + this.mat[0][M-1] +
this.mat[M-1][M-1];

          // returning the recorded sum
```

```
          return sum;
     }

     void rotateMat()
     {
          // rotated matrix
          char[][] rot_mat = new char[this.M][this.M];

          // loop through the coloums from first to last
          // loop through the rows from last to first
          // rotate the mat matrix
          for(int i = 0; i < M; i++)
          {
               for(int j = 0; j < M; j++)
               {
                    rot_mat[i][j] = this.mat[j][M-i-1];
               }
          }

          // setting original matrix to rotated matrix
          this.mat = rot_mat;
     }

     static Matrix readMat()
     {
          // create input handler
          // input handler
          Scanner sc = new Scanner(System.in);

          // accept size as input
          // Input from the user about the size of matrix
          int M = sc.nextInt();

          // matrix object for matrix operations
          Matrix m = new Matrix(M);

          // loop through the matrix to generate the matrix by accept-
ing input
          for(int i = 0; i < M; i++)
          {
               for(int j = 0; j < M; j++)
               {
                    m.mat[i][j] = sc.next().charAt(0);
               }
          }

          // return the generated matrix
          return m;
     }
```

```
}
import java.util.Scanner;
public class Matrix_main
{
     public static void main(String args[])
     {
          // accpting input from user
          // Matrix object
          Matrix m = Matrix.readMat();

          // displaying original matrix
          System.out.println("0riginal Matrix: ");
          m.display();

          // displaying the sum of the corners
          System.out.println("0um = "+m.unicodeSum());

          // roatate the actual matrix
          m.rotateMat();

          // display the rotated matrix
          System.out.println("0inal Matrix: ");
          m.display();
     }
}
```

Varible Listing

Name	Function	Type	Scope
M	stores the size of square matrix	int	Matrix
mat	stores the matrix itself	char[][]	Matrix
sum	stores the sum of the corners of the mat	int	unicodeSum
rot_mat	rotated matrix	char[][]	rotateMat
sc	input handler	Scanner	readMat
M	Input from the user about the size of matrix	int	readMat
m	matrix object for matrix operations	Matrix	readMat

Assignment 5

Design a class StringModify in your default package that will contain two methods that will work on string values. The method definitions of the class is given below:

i) StringModify(String st): parameterized constructor

ii) String insertStringAt(String w,int pos): to insert string w at valid position pos and returns final sentence without changing any other data.

iii) String deleteCharAt(char w,int pos): to delete character w from valid position pos and returns final sentence without changing any other data.

Write a possible menu in main method to implement the above logic for any random sentence by calling methods.

DO POSSIBLE CHECKING WHERE REQUIRED.

Algorithm

StringModify
Step 1: Start
Step 2: initializes the original
Step 3: End

insertStringAt
Step 1: Start
Step 2: checks if pos is a valid position
Step 3: if not return null
Step 4: append w at pos in st
Step 5: use the + string concatination operator
Step 6: End

deleteCharAt
Step 1: Start
Step 2: checks if pos is a valid position
Step 3: if not return null
Step 4: checks if the char at pos is actually the required char
Step 5: if yes then it returns modifined string otherwise just the original string
Step 6: End

main
Step 1: Start
Step 2: accept menu entry option
Step 3: if the option is rediculous just return
Step 4: accept a sentence
Step 5: End

Source Code

```
import java.util.Scanner;

public class StringModify
{
     // original string
     String st;

     StringModify(String st)
     {
          // initializes the original
          this.st = st;
     }

     String insertStringAt(String w, int pos)
     {
          // checks if pos is a valid position
          // if not return null

          if(pos < 0 || pos >= this.st.length())
               return null;

          // append w at pos in st
          // use the + string concatination operator

          return   this.st.substring(0,   pos)   +   w   +   this.st.sub-
string(pos);
     }

     String deleteCharAt(char w, int pos)
     {
          // checks if pos is a valid position
          // if not return null

          if(pos < 0 || pos >= this.st.length())
               return null;

          // checks if the char at pos is actually the required char
          // if yes then it returns modifined string otherwise just the
original string

          return    st.charAt(pos)    ==    w    ?    this.st.substring(0,
pos)+this.st.substring(pos+1) : this.st;
     }
```

```
}
import java.util.Scanner;
public class StringModify_main
{
     public static void main(String args[])
     {
          // Input Handler
          Scanner sc = new Scanner(System.in);

          // accept menu entry option
          // option of menuentry
          int opt = sc.nextInt();

          // if the option is rediculous just return
          if(opt != 1 && opt != 2)
               return;

          // accept a sentence
          // generate an object of StringModify using that sentence
          StringModify   s   =   new   StringModify(new   Scanner(Sys-
tem.in).nextLine());

          // the new string that needs to be attached or removed
          String w = sc.next();

          // position of the attachment or removal
          int pos = sc.nextInt();

          switch(opt)
          {
               case 1:
                    System.out.println(s.insertStringAt(w, pos));
                    break;

               case 2:
                    System.out.println(s.deleteCharAt(w.charAt(0),
pos));
                    break;
          }
     }
}
```

Varible Listing

Name	Function	Type	Scope
st	original string	String	StringModify

Assignment 6

Given two possible numbers M and N, such that M is between 100 and 10000 and N is less than 100. Find the smallest integer that greater than M and whose digits add up to N. For example, if M = 100 and N = 11, then the smallest integer greater than 100 whose digits add up to 11 is 119

Write a program to accept the numbers M and N from the user and print the smallest required number whose sum of all its digits is equal to N. Also, print the total number of digits presnet in the required number. The program should check for the validity of the inputs display an appropriate message for an invalid input.

Test your program with the sample data and some random data.

Example 1
INPUT:
M = 100
N = 11
OUTPUT:
The required Number = 119
Total number of digits = 3

Example 2
INPUT:
M = 1500
N = 25
OUTPUT:
The required Number = 1699
Total number of digits = 4

Example 3
INPUT:
M = 99
N = 11
OUTPUT:
INVALID INPUT

Algorithm

SumDigit
Step 1: Start
Step 2: initializing M with local M
Step 3: initializing N with local N
Step 4: End

digitSum
Step 1: Start
Step 2: create a duplicate of M
Step 3: while M is not 0 loop
Step 4: add all athe digits of M
Step 5: End

genNum
Step 1: Start
Step 2: check the value of M and N to make sure they are in range
Step 3: if not just return -1 to mark invalid input
Step 4: start a loop which to go from M till the upper bound
Step 5: check if any of the numbers is actually the required number
Step 6: if the number is found return the number of digits in the number
Step 7: store the number in M
Step 8: other wise return -1 marking an invalid input
Step 9: End

main
Step 1: Start
Step 2: accept input from the user
Step 3: use the input to initialize the object
Step 4: if the digit is -1 then it is invalid input
Step 5: End

Source Code

```
import java.util.Scanner;

public class SumDigit
{
     // lower bound of operations
     int M;

     // the number to be derived from digits
     int N;

     SumDigit(int M, int N)
     {
          // initializing M with local M
          this.M = M;

          // initializing N with local N
          this.N = N;
     }

     int digitSum()
     {
          // create a duplicate of M
          // local version of M for computation
          int M = this.M;

          // stores the sum of digits
          int sum = 0;

          // while M is not 0 loop
          // add all athe digits of M
          // return the added digits
          while(M != 0)
          {
               sum += M%10;
               M /= 10;
          }

          return sum;
     }

     int genNum()
     {
          int digits = -1;
```

```
          // check the value of M and N to make sure they are in range
          // if not just return -1 to mark invalid input
          if(this.M >= 10000 || this.M < 100 || this.N > 100)
               return -1;

          // start a loop which to go from M till the upper bound
          // check if any of the numbers is actually the required num-
ber
          while(this.M < 10000)
          {
               if(digitSum() == this.N)
               {
                    digits = (int)(Math.log10(this.M) + 1);
                    break;
               }
               this.M++;
          }

          // if the number is found return the number of digits in the
number
          // store the number in M
          // other wise return -1 marking an invalid input
          return digits;
     }


}
import java.util.Scanner;
public class SumDigit_main
{
     public static void main(String args[])
     {
          // input handler
          Scanner sc = new Scanner(System.in);

          // accept input from the user
          // use the input to initialize the object
          // call genNum and display the value
          SumDigit s = new SumDigit(sc.nextInt(), sc.nextInt());

          //stores the digits
          int digits = s.genNum();

          // if the digit is -1 then it is invalid input
          if(digits == -1)
          {
               System.out.println("INVALID INPUT");
          }
          else
```

```
            {
                System.out.println("The required Number = "+s.M);
                System.out.println("Total number of digits = "+digits);
            }
        }
}
```

Varible Listing

Name	Function	Type	Scope
M	lower bound of operations	int	SumDigit
N	the number to be derived from digits	int	SumDigit
M	local version of M for computation	int	digitSum
sum	stores the sum of digits	int	digitSum
digits	return the added digits	int	genNum

Assignment 7

Write a program to accetp a pragraph containing **TWO** senteces only. The sentences may be terminated by either '.', '?', or '!' only. Any other character may be ignored. The words are to be separated by single blank space and must be in UPPER CASE.

Perform the following tasks:
a) Check for the validity of the accepted paragraph for the number of sentecs and for the terminating charater.
b) Separate the two sentences from the paragraph and find common words in the two senteces with their frequency of occurence in the paragraph.
c) Display both the sentences separately along with common words and their frequency, in the format given below:

Test your program for the following data and some random data:

Example 1
INPUT:

IS IT RAINING? YOU MAY GET WET IF IT IS RAINING.

OUTPUT:

IS IT RAINING?
YOU MAY GET WET IF IT IS RAINING.
COMMON WORDS FREQUENCY
IS 2
IT 2
RAINING 2

Example 2
INPUT:

ARE YOU COMMING? I AM GETTINNG LATE.

OUTPUT:

ARE YOU COMMING?
I AM GETTINNG LATE.

NO COMMON WORDS

*

Algorithm

Sentences
Step 1: Start
Step 2: initializing the raw
Step 3: initializing sentences with the given criteria
Step 4: End

checkValidity
Step 1: Start
Step 2: check if there is exactly 2 sentences
Step 3: if not return false
Step 4: convert the raw input to upper case
Step 5: check if the upper cased string is exact match to the raw string
Step 6: if not return false
Step 7: if all conditions pass return true
Step 8: End

findWordFrequency
Step 1: Start
Step 2: gets the number of occurence of the particular word
Step 3: loop through the local raw until no more words are left
Step 4: check if the word matches if so increment f
Step 5: return frequeny of words
Step 6: End

getCommonWord
Step 1: Start
Step 2: loop through the words
Step 3: if a match is encountered print the word
Step 4: return the generated output
Step 5: End

main
Step 1: Start
Step 2: accept input
Step 3: print the required value
Step 4: End

Source Code

```
import java.util.Scanner;

public class Sentences
{
     // sotres the actual raw input
     String raw;

     // stores the 2 sentences
     String[] sentences;

     Sentences(String raw)
     {
          // initializing the raw
          this.raw = raw;

          // initializing sentences with the given criteria
          this.sentences = raw.split("\?|\.|!");
     }

     boolean checkValidity()
     {
          // check if there is exactly 2 sentences
          // if not return false
          if(this.sentences.length != 2)
               return false;

          // convert the raw input to upper case
          // check if the upper cased string is exact match to the raw
string
          // if not return false
          if(!this.raw.toUpperCase().equals(this.raw))
               return false;

          // if all conditions pass return true
          return true;
     }

     int findWordFrequency(String word)
     {
          // stores the frequency of words
          int f = 0;

          // the raw paragraph with a space at the end
```

```
     String raw = this.sentences[0]+" "+this.sentences[1]+" ";

     // stores the words present in raw
     String[] words = raw.split(" ");

     // gets the number of occurence of the particular word
     // loop through the local raw until no more words are left
     // check if the word matches if so increment f
     for(String w : words)
     {
          if(word.equals(w))
          {
               f++;
          }
     }

     // return frequeny of words
     return f;
}

String getCommonWord()
{
     // the raw paragraph with a space at the end
     String raw = this.sentences[0]+" "+this.sentences[1]+" ";

     // generate the words for the sentences
     String[] words1 = this.sentences[0].split(" ");

     // generate the words for the sentences
     String[] words2 = this.sentences[1].split(" ");

     // generated common words formated output
     String x = "";

     // loop through the words
     // if a match is encountered print the word
     for(String w1 : words1)
     {
          for(String w2 : words2)
          {
               if(w1.equals(w2))
               {
                    x += w1 + ""+ findWordFrequency(w1)+"0;
                    break;
               }
          }
     }

     // return the generated output
```

```
          return x;
     }


}
import java.util.Scanner;
public class Sentences_main
{
     public static void main(String args[])
     {
          // input handler
          Scanner sc = new Scanner(System.in);

          // setnece object
          Sentences se = new Sentences(sc.nextLine());

          // composited output
          String x = se.getCommonWord();

          // accept input
          // print the required value
          if(x.equals(""))
          {
               System.out.println("NO COMMON WORDS");
          }
          else
          {
               System.out.println(se.sen-
tences[0].trim()+se.raw.charAt(se.sentences[0].length()));
               System.out.println(se.sen-
tences[1].trim()+se.raw.trim().charAt(se.raw.trim().length()-1));
               System.out.println("0OMMON WORDSFREQUENCY");
               System.out.println(x);
          }
     }
}
```

Varible Listing

Name	Function	Type	Scope
raw	sotres the actual raw input	String	Sentences
sentences	stores the 2 sentences	String[]	Sentences
f	stores the frequency of words	int	findWordFrequency
raw	the raw paragraph with a space at the end	String	findWordFrequency
words	stores the words present in raw	String[]	findWordFrequency
raw	the raw paragraph with a space at the end	String	getCommonWord
words1	generate the words for the sentences	String[]	getCommonWord
words2	generate the words for the sentences	String[]	getCommonWord
x	generated common words formated output	String	getCommonWord

Assignment 8

A class Mix has been defined to mix two words, character by character, in the following manner:
The first character of the first word is followed by the first character of the second word and so on. If the words are of different length, the remaining characters of the longer word are put at the end.

Example: If the First word is "JUMP" and the second word is "STROLL", then the required word will be "JSUTMRPOLL"

Some of the memebers of the class are given below:

Class Name : **Mix**

Data member/instance variable:

wrd	:	to store a word
len	:	to store a word

Memeber functions/methods:

Mix()	:	default constructor to initialize the data members with legal initial value
void feedword()	:	to accept the word in UPPER case
void mix_word(Mix P, Mix Q)	:	mixes the words of object P and Q as stated above stores the resultant word in the current object
void display()	:	display the word

Specify the class Mix giving the details of the constructor(), void feedword(), void mix_word(Mix, Mix) and void display(). Define the main() function to create objects and call the functions accordingly to enable the task.

Algorithm

Mix
Step 1: Start
Step 2: Initialize values to default values
Step 3: initializing wrd with ""
Step 4: initializing len with 0
Step 5: End

feedword
Step 1: Start
Step 2: accept a singular word
Step 3: store the word in wrd
Step 4: check if the word is upper case
Step 5: if no kill the program
Step 6: otherwise initialize len
Step 7: End

mix_word
Step 1: Start
Step 2: loop throught the letter of P and Q until a boundary of the smaller is hit
Step 3: take the remaing value of P if any and add them to the wrd
Step 4: take the remaing value of Q if any and add them to the wrd
Step 5: correct the value of len
Step 6: End

display
Step 1: Start
Step 2: display the value of the word
Step 3: End

main
Step 1: Start
Step 2: accept input for P and Q
Step 3: call mix_word function to execute operation
Step 4: display computed value
Step 5: End

Source Code

```
import java.util.Scanner;

public class Mix
{
     // to store a word
     String wrd;

     // to sotre the length of the word
     int len;

     Mix()
     {
          // Initialize values to default values
          // initializing wrd with ""
          this.wrd = "";

          // initializing len with 0
          this.len = 0;
     }

     void feedword()
     {
          // accept a singular word
          // input handler
          Scanner sc = new Scanner(System.in);

          // store the word in wrd
          this.wrd = sc.next();

          // check if the word is upper case
          // if no kill the program
          if(!this.wrd.toUpperCase().equals(this.wrd))
               System.exit(1);

          // otherwise initialize len
          this.len = this.wrd.length();
     }

     void mix_word(Mix P, Mix Q)
     {
          // loop throught the letter of P and Q until a boundary of
the smaller is hit
          while(this.len < P.len && this.len < Q.len)
```

```
          {
               this.wrd                                                  +=
""+P.wrd.charAt(this.len)+Q.wrd.charAt(this.len);
               this.len++;
          }

          // take the remaing value of P if any and add them to the wrd
          if(this.len < P.len)
               this.wrd += P.wrd.substring(this.len);

          // take the remaing value of Q if any and add them to the wrd
          if(this.len < Q.len)
               this.wrd += Q.wrd.substring(this.len);

          // correct the value of len
          this.len = this.wrd.length();
     }

     void display()
     {
          // display the value of the word
          System.out.println(this.wrd);
     }


}
import java.util.Scanner;
public class Mix_main
{
     public static void main(String args[])
     {
          // P in the mix_word
          Mix P = new Mix();

          // Q in the mix_word
          Mix Q = new Mix();

          // the mix object for computation
          Mix m = new Mix();

          // accept input for P and Q
          P.feedword();
          Q.feedword();

          // call mix_word function to execute operation
          m.mix_word(P, Q);

          // display computed value
          m.display();
```

```
        }
}
```

Varible Listing

Name	Function	Type	Scope
wrd	to store a word	String	Mix
len	to sotre the length of the word	int	Mix
sc	input handler	Scanner	feedword

Assignment 9

Q9) Design a class FiboPrime which will display all the the Fibonacci numbers upto n terms which have atleast one prime digit in the number. For example 2,3,5,13,21 are some of the examples of Fibonacci numbers having atleast one prime digit in it.

Class name: FiboPrime

Data members:
n: number of terms

Method:
FiboPrime(int): constructor

int fibo(int n): returns nth Fibonacci number

void displayFiboPrimes(): Display all the Fibonacci numbers upto n terms which have atleast one digit as prime

boolean isPrime(int p): returns true or false if p is either prime or not.

You can add method(s) if required.

Algorithm

FiboPrime
Step 1: Start
Step 2: initializing fibo prime using n
Step 3: End

fibo
Step 1: Start
Step 2: loop until n is zero
Step 3: set a = a+b
Step 4: and b = a-b
Step 5: return b as the nth fibo number
Step 6: End

isPrime
Step 1: Start
Step 2: if p is 1 then its not prime
Step 3: loop through numbers starting from 2 till p
Step 4: if anyone is divisible by p return flase
Step 5: if all the conditions fail then it must be true
Step 6: End

hasPrime
Step 1: Start
Step 2: loop throught the digits of a number
Step 3: if a prime number is found return true
Step 4: otherwise return false
Step 5: End

displayFiboPrimes
Step 1: Start
Step 2: loop through all the fibo numbers until n terms
Step 3: if a prime fibo is encountered print it
Step 4: End

main
Step 1: Start
Step 2: create an object using user input
Step 3: call displayFiboPrimes using that object
Step 4: End

Source Code

```
import java.util.Scanner;

public class FiboPrime
{
     // number of terms
     int n;

     FiboPrime(int n)
     {
          // initializing fibo prime using n
          this.n = n;
     }

     int fibo(int n)
     {
          // second fibo number
          int a = 1;

          // first fibo number
          int b = 0;

          // loop until n is zero
          // set a = a+b
          // and b = a-b
          while(n != 0)
          {
               a = a+b;
               b = a-b;
               n--;
          }

          // return b as the nth fibo number
          return b;
     }

     boolean isPrime(int p)
     {
          // if p is 1 then its not prime
          if(p == 1) return false;

          // iterator from 2 until p
          int i = 2;
```

```
        // loop through numbers starting from 2 till p
        // if anyone is divisible by p return flase
        while(i != p)
        {
            if(p % i == 0)
                return false;
            i++;
        }

        // if all the conditions fail then it must be true
        return true;
    }

    boolean hasPrime(int p)
    {
        // loop throught the digits of a number
        // if a prime number is found return true
        // otherwise return false
        while(p != 0)
        {
            if(isPrime(p%10))
                return true;
            p/=10;
        }

        return false;
    }

    void displayFiboPrimes()
    {
        // iterator from 1 to n
        int i = 1;

        // fibo accumulator
        int fb = 1;

        // loop through all the fibo numbers until n terms
        // if a prime fibo is encountered print it
        while(i < n)
        {
            fb = fibo(i);
            if(hasPrime(fb))
                System.out.println(fb);
            i++;
        }
    }


}
```

```
import java.util.Scanner;
public class FiboPrime_main
{
     public static void main(String args[])
     {
          // Input handler
          Scanner sc = new Scanner(System.in);

          // create an object using user input
          // call displayFiboPrimes using that object
          new FiboPrime(sc.nextInt()).displayFiboPrimes();
     }
}
```

Varible Listing

Name	Function	Type	Scope
n	number of terms	int	FiboPrime
a	second fibo number	int	fibo
b	first fibo number	int	fibo
i	iterator from 2 until p	int	isPrime
i	iterator from 1 to n	int	displayFiboPrimes
fb	fibo accumulator	int	displayFiboPrimes

Assignment 10

Write a program to declare a matrix A[][] having order MxN(where M is no. of rows and N is no. of columns) where values of both M and N must be greater than 2 and less than 10.Allow the user to accept value for matrix. Perform the following tasks:
a) Display original matrix
b) Sort each odd row of the matrix in descending order using bubble sort algorithm and each even row of the matrix in ascending order using selection sort algorithm.
c) Display the final updated matrix.

Algorithm

MxN
Step 1: Start
Step 2: initializing A with a new matrix
Step 3: initializing rows and cols
Step 4: End

bsort
Step 1: Start
Step 2: loop through the arr
Step 3: check if any element is smaller is than the next element
Step 4: if it is then swap the elements
Step 5: End

ssort
Step 1: Start
Step 2: loop through the arr
Step 3: check if any element is bigger than the currently selected element
Step 4: End

sort
Step 1: Start
Step 2: loop through the rows
Step 3: sort the loops according to ther index
Step 4: if odd send them to bsort
Step 5: else send them to ssort
Step 6: this would sort the matrix
Step 7: End

display
Step 1: Start
Step 2: using an iterative forloop print all the values
Step 3: print a newline at the end of line
Step 4: End

main
Step 1: Start
Step 2: creating object of MxN
Step 3: take input from stdin
Step 4: display original matrix
Step 5: sort the original in the fation metioned matrix
Step 6: display the sorted matrix
Step 7: End

Source Code

```
import java.util.Scanner;

public class MxN
{
     // Original Matrix
     int[][] A;

     // number of rows
     int M;

     // number of cols
     int N;

     MxN(int M, int N)
     {
          // initializing A with a new matrix
          this.A = new int[M][N];

          // initializing rows and cols
          this.M = M;
          this.N = N;
     }

     void bsort(int[] arr)
     {
          // iterator
          int i = 0;

          // internal iterator
          int j = 0;


          // loop through the arr
          // check if any element is smaller is than the next element
          // if it is then swap the elements
          while(i < arr.length)
          {
               j = i;
               while(j < arr.length-1)
               {
                    if(arr[j+1] > arr[j])
                    {
                         // bubble sort arry element stuck
```

```
                    int x = arr[j+1];
                    arr[j+1] = arr[j];
                    arr[j] = x;
               }
               j++;
          }
          i++;
     }
}

void ssort(int[] arr)
{
     // iterator
     int i = 0;

     // internal iterator
     int j = 0;

     // minimum number index
     int jmin = 0;

     // loop through the arr
     // check if any element is bigger than the currently selected
element
     // if it is then swap the elements
     while(i < arr.length)
     {
          j = 0;
          jmin = 0;

          while(j < arr.length)
          {
               if(arr[i] < arr[jmin])
               {
                    jmin = j;
               }
               j++;
          }

          int x = arr[i];
          arr[i] = arr[jmin];
          arr[jmin] = x;

          i++;
     }
}

void sort()
{
```

```
          // iterator
          int i = 0;

          // loop through the rows
          // sort the loops according to ther index
          // if odd send them to bsort
          // else send them to ssort
          // this would sort the matrix
          while(i < M)
          {
               if((i+1) % 2 == 0)
                    ssort(this.A[i]);
               else
                    bsort(this.A[i]);
               i++;
          }
     }

     void display()
     {
          // index of the rows
          int i = 0;

          // index of the cols
          int j = 0;

          // using an iterative forloop print all the values
          // print a newline at the end of line
          while(i < M)
          {
               j = 0;
               while(j < N)
               {
                    System.out.print(j+" ");
                    j++;
               }
               System.out.println();
               i++;
          }
     }


}
import java.util.Scanner;
public class MxN_main
{
     public static void main(String args[])
     {
          // input handler
```

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

        // creating object of MxN
        // object of MxN
        MxN m = new MxN(sc.nextInt(), sc.nextInt());

        // row iterator
        int i = 0;

        // col iterator
        int j = 0;

        // take input from stdin
        for(i = 0; i < m.M; i++)
        {
             for(j = 0; j < m.N; j++)
             {
                  m.A[i][j] = sc.nextInt();
             }
        }

        // display original matrix
        m.display();

        // sort the original in the fation metioned matrix
        m.sort();

        // display the sorted matrix
        m.display();
    }
}
```

Varible Listing

Name	Function	Type	Scope
A	Original Matrix	int[][]	MxN
M	number of rows	int	MxN
N	number of cols	int	MxN
i	iterator	int	bsort
j	internal iterator	int	bsort
x	bubble sort arry element stuck	int	if
i	iterator	int	ssort
j	internal iterator	int	ssort
jmin	minimum number index	int	ssort
x	if it is then swap the elements	int	while
i	iterator	int	sort
i	index of the rows	int	display
j	index of the cols	int	display

Assignment 11

A superclass Binary has been defined to accept a binary number and a subclass ToHex has been defined to convert binary number into its equivalent hexadecimal number using short cut logic of combining bits. Some of the members of the class are given below:

Class name : Binary

Data members
n : stores the binary number

Member functions:
BinHex(int n) : constructor to initialize the data member
void display(): display the binary number

Class name: ToHex

Data member:
hex: to store hexadecimal number

Methods:
ToHex(...): parameterized constructor
void bin_hex() : calculates the hexadecimal equivalent of n and stores it in hex.(using short cut logic of combining bits)
void display() : displays the binary number and hexadecimal number. You can add any extra methods if required.
Using concept of inheritance write details of both the classes and write main method accordingly.
*

Algorithm

BinHex
Step 1: Start
Step 2: initialize n using local n
Step 3: End

display
Step 1: Start
Step 2: loop through n digits
Step 3: when n is zero exit
Step 4: print each digit of number
Step 5: display a new line at the end for pretty print
Step 6: End

ToHex
Step 1: Start
Step 2: initialize super object
Step 3: hexnumber version of n
Step 4: End

bin_hex
Step 1: Start
Step 2: until n is zero loop
Step 3: generate a number from binary encoded decimal number
Step 4: attaching the number after generation to the hex value
Step 5: remove 4 digits from the end of n
Step 6: End

display
Step 1: Start
Step 2: calling super’s display function
Step 3: a character mapper is used for prining
Step 4: loop through n digits base 16
Step 5: when n is zero exit
Step 6: print each digit of number
Step 7: display a subtle newline at the end
Step 8: End

main
Step 1: Start
Step 2: object is generated using user input
Step 3: execute bin_hex
Step 4: display the usable information

Step 5: End

Source Code

```
import java.util.Scanner;

class BinHex
{
     // stores the binary number
     int n;

     BinHex(int n)
     {
          // initialize n using local n
          this.n = n;
     }

     void display()
     {
          // local version of n
          int n = this.n;

          // output number
          String output = "";

          // loop through n digits
          // when n is zero exit
          // print each digit of number
          while(n != 0)
          {
               output = n%10 + output;
               n /= 10;
          }

          // display a new line at the end for pretty print
          System.out.println(output);
     }
}

public class ToHex extends BinHex
{
     // to store hexadecimal number
     int hex;

     ToHex(int n)
     {
          // initialize super object
```

```
          super(n);

          // hexnumber version of n
          this.hex = 0;
     }

     void bin_hex()
     {
          // local copy of n
          int n = super.n;

          // shift register
          int shl = 0;

          // until n is zero loop
          // generate a number from binary encoded decimal number
          // attaching the number after generation to the hex value
          // remove 4 digits from the end of n
          while(n != 0)
          {
               // decoded binary encoded decimal
               int number = (((n % 10000)/1000) << 3) |
                    (((n % 1000)/100) << 2) |
                    (((n % 100)/10) << 1) |
                    (n % 10);

               this.hex = this.hex | (number << shl);

               n /= 10000;
               shl += 4;
          }
     }

     void display()
     {
          // calling super's display function
          super.display();

          // output number
          String output = "";

          // a character mapper is used for prining
          // a hex character mapper
          char[] hex_map = {'0', '1', '2', '3', '4', '5', '6', '7',
'8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};

          // loop through n digits base 16
          // when n is zero exit
          // print each digit of number
```

```
          while(hex != 0)
          {
               output = hex_map[hex%16] + output;
               hex /= 16;
          }

          // display a subtle newline at the end
          System.out.println(output);
     }


}
import java.util.Scanner;
public class ToHex_main
{
     public static void main(String arg[])
     {
          // input handler
          Scanner sc = new Scanner(System.in);

          // object is generated using user input
          // to hex object
          ToHex th = new ToHex(sc.nextInt());

          // execute bin_hex
          th.bin_hex();

          // display the usable information
          th.display();
     }
}
```

Varible Listing

Name	Function	Type	Scope
n	stores the binary number	int	BinHex
n	local version of n	int	display
output	output number	String	display
hex	to store hexadecimal number	int	BinHex
n	local copy of n	int	bin_hex
shl	shift register	int	bin_hex
number	decoded binary encoded dec-imal	int	while
output	output number	String	display
hex_map	a hex character mapper	char[]	display

Assignment 12

A super class Sentence accepts a sentence in uppercase terminated by â.â only. A sub-class Encrypt will encrypt the words in the sentence with a valid logic given below.

Class name: Sentence

Data members:
sen: accepts a sentence in uppercase and terminated by â.â only. Words in the sentence can be separated by one or more spaces.

Methods:
Sentence(String): constructor
void show(): update the sentence where each word will be separated by single space and terminated by â.â. Display the updated sentence.

Class name: Encrypt

Data member:
nsen: stores encrypted sentence

Methods:
Encrypt(...): constructor
void encrypt(): encrypt the words in the updated sentence as per logic given below:
i) For the word(s) starting with vowel, write the vowel then append consecutive consonants and vowels present in the word. Example say if the word is EXAMINATION then encrypted word will be EXAMINATINO
ii) For the word(s) starting with consonant, arrange the characters in the word in descending order as per ASCII value. Example say if the word is CONSTANT then encrypted word will be TTSONNCA.
Finally create the encrypted sentence with encrypted word terminated by â.â
void show(): display updated original and encrypted sentence.
*

Algorithm

Sentence
Step 1: Start
Step 2: initialize the sentence
Step 3: End

show
Step 1: Start
Step 2: display sen
Step 3: End

Encrypt
Step 1: Start
Step 2: initializing super class by sending original sentence
Step 3: initialize null sting for nsen
Step 4: End

encrypt
Step 1: Start
Step 2: split the sentence into ' '
Step 3: remove the '.' at the end becuase its of no use
Step 4: iterate over the words
Step 5: if the word starts with vowel
Step 6: then attach vowels and consonats one after another
Step 7: otherwise sort the whole chars in desending order
Step 8: bubble sort the chars
Step 9: End

show
Step 1: Start
Step 2: call the super show function
Step 3: display the encrypted sentence
Step 4: End

main
Step 1: Start
Step 2: test if the line ends with '.'
Step 3: if not nuke the program
Step 4: encrypt text supplied
Step 5: show the encrpyted text
Step 6: End

Source Code

```
import java.util.Scanner;

class Sentence
{
     // actual sentence
     String sen;

     Sentence(String sen)
     {
          // initialize the sentence
          this.sen = sen;
     }

     void show()
     {
          // display sen
          System.out.println(this.sen);
     }
}

public class Encrypt extends Sentence
{
     // stores encrypted sentence
     String nsen;

     Encrypt(String sen)
     {
          // initializing super class by sending original sentence
          super(sen);

          // initialize null sting for nsen
          this.nsen = "";
     }

     void encrypt()
     {
          // split the sentence into ' '
          // remove the '.' at the end becuase its of no use
          // words of the sentence ladies and gentle men
          String[]     words     =     super.sen.substring(0,     su-
per.sen.length()-1).split(" ");

          // words iterator controler
```

```
          int i = 0;

          // iterate over the words
          // if the word starts with vowel
          // then attach vowels and consonats one after another
          // otherwise sort the whole chars in desending order
          // bubble sort the chars
          // if word length is 1 then dont do anything just attach the
word
          while(i < words.length)
          {
               if(words[i].length() == 1)
               {
                    this.nsen += words[i] + " ";
               }
               else if("aeiouAEIOU".indexOf(words[i].charAt(0)) >= 0)
               {
                    // list of vowels
                    String vowels = "";

                    // list of consonants
                    String consonants = "";

                    // iterator j
                    int j = 0;
                    while(j < words[i].length())
                    {
                         if("aeiouAEIOU".indexOf(words[i].charAt(j))  >=
0)
                         {
                              vowels += words[i].charAt(j);
                         }
                         else
                         {
                              consonants += words[i].charAt(j);
                         }
                         j++;
                    }

                    j = 0;
                    while(j   <   vowels.length()   &&   j   <   conso-
nants.length())
                    {
                         this.nsen += vowels.charAt(j);
                         this.nsen += consonants.charAt(j);
                         j++;
                    }

                    if(j < vowels.length())
```

```
                    this.nsen += vowels.substring(j);

               if(j < consonants.length())
                    this.nsen += consonants.substring(j);

               this.nsen += " ";
          }
          else
          {
               char[] letters = words[i].toCharArray();

               // iterator k
               int k = 0;

               // iterator l
               int l = 0;

               for(k = 0; k < letters.length; k++)
               {
                    for(l = 0; l < letters.length-1; l++)
                    {
                         if(letters[l] < letters[l+1])
                         {
                              // duplicate letters
                              char x = letters[l];
                              letters[l] = letters[l+1];
                              letters[l+1] = x;
                         }
                    }
               }

               this.nsen += new String(letters);

               this.nsen += " ";
          }
          i++;
     }
     this.nsen += ".";
}

void show()
{
     // call the super show function
     super.show();

     // display the encrypted sentence
     System.out.println(this.nsen);
}
```

```
}
import java.util.Scanner;
public class Encrypt_main
{
     public static void main(String args[])
     {
          // input handler
          Scanner sc = new Scanner(System.in);

          // line input
          String line = sc.nextLine();

          // test if the line ends with '.'
          // if not nuke the program
          if(line.charAt(line.length()-1) != '.')
               return;

          // encrypt object creation
          Encrypt e = new Encrypt(line);

          // encrypt text supplied
          e.encrypt();

          // show the encrpyted text
          e.show();
     }
}
```

Varible Listing

Name	Function	Type	Scope
sen	actual sentence	String	Sentence
nsen	stores encrypted sentence	String	Sentence
words	words of the sentence ladies and gentle men	String[]	encrypt
i	words iterator controler	int	encrypt
vowels	list of vowels	String	if
consonants	list of consonants	String	if
j	iterator j	int	if
letters	if word length is 1 then dont do anything just attach the word	char[]	else
k	iterator k	int	else
l	iterator l	int	else
x	duplicate letters	char	if

Assignment 13

A superclass Number is defined to accept number of terms and also calculate the factorial of a number. Define a subclass Series to find the product of the series
P = x * x2/3! * x4/4! * x8/5! * x16/6! ..... n terms
The details of the members of both classes are given below:

Class name: Number

Data member/instance variable:
n: to store an integer number

Member functions/methods:
Number(int): constructor to initialize the data member
int factorial(int a): returns the factorial of a number(use recursion)
(factorial of n(n!) = 1 Ã 2 Ã 3 Ã â¦â¦ Ã n)
void display(): displays the value of n

Class name: Series

Data member/instance variable:
prod: to store the product of the series
x: accepts value of unknown variable x(in double)

Member functions/methods:
Series(â¦) : parameterized constructor to initialize the data members of both the classes
void calProd(): calculates the PRODUCT of the given series
void display(): displays the data members of both the classes
*

Algorithm

Number
Step 1: Start
Step 2: initialize the value of n using local value
Step 3: End

factorial
Step 1: Start
Step 2: if a == 1 return a
Step 3: if that is not the case multiply a with the return value of factorial(a-1)
Step 4: End

display
Step 1: Start
Step 2: print the value of n
Step 3: End

Series
Step 1: Start
Step 2: initialize the super class object
Step 3: initialize the local value of x
Step 4: initialize prod to 1 (-_-)
Step 5: End

calProd
Step 1: Start
Step 2: loop using the iterator
Step 3: generate the product using the fomulae provided
Step 4: End

display
Step 1: Start
Step 2: call super's display function
Step 3: print the product of the value
Step 4: End

main
Step 1: Start
Step 2: generates the series from user input
Step 3: calculate the product
Step 4: display the product
Step 5: End

Source Code

```
import java.util.Scanner;

class Number
{
     // to store an integer number
     int n;

     Number(int n)
     {
          // initialize the value of n using local value
          this.n = n;
     }

     int factorial(int a)
     {
          // if a == 1 return a
          if(a == 1) return a;

          // if that is not the case multiply a with the return value
of factorial(a-1)
          return a * factorial(a-1);
     }

     void display()
     {
          // print the value of n
          System.out.println(this.n);
     }
}

public class Series extends Number
{
     // to store the product of the series
     int prod;

     // accepts value of unknown variable x(in double)
     int x;

     Series(int n, int x)
     {
          // initialize the super class object
          super(n);
```

```
          // initialize the local value of x
          this.x = x;

          // initialize prod to 1 (-_-)
          this.prod = 1;
     }

     void calProd()
     {
          // create an iterator
          int i = 1;

          // loop using the iterator
          // generate the product using the fomulae provided
          while(i <= n)
          {
               this.prod *= Math.pow(x, i)/factorial(i);
          }
     }

     void display()
     {
          // call super’s display function
          super.display();

          // print the product of the value
          System.out.println(this.prod);
     }


}
import java.util.Scanner;
public class Series_main
{
     public static void main(String args[])
     {
          // input handler
          Scanner sc = new Scanner(System.in);

          // generates the series from user input
          // series object
          Series s = new Series(sc.nextInt(), sc.nextInt());

          // calculate the product
          s.calProd();

          // display the product
          s.display();
     }
```

```
}
```

Varible Listing

Name	Function	Type	Scope
n	to store an integer number	int	Number
prod	to store the product of the series	int	Number
x	accepts value of unknown variable x(in double)	int	Number
i	create an iterator	int	calProd