Computer Science

Project Portfolio

Sarang, XII

2020-21

Contents

[Note: All programs below use objects]

Arithmetic Programs: pg

Strings:

Stacks and Queues:

1D Arrays:

2D Arrays:

Recursive Programs:

Inheritance, Abstract Classes & Interfaces:

Classes & Objects:

1. Fletcher Number

A fletcher Number is a number which is a prime number by itself and

when rotated (last digit becomes the first digit), all such rotated numbers (until you reach the original number) are also prime. Eg. 919 <> 991 <> 199 are all prime ∴ 919 is a Fletcher Number. This program checks if the input number is a Fletcher Number and returns true/false.

Code:

```
import java.util.Scanner;
public class FletcherNum {
   int fno;
   Scanner input=new Scanner(System.in);
   FletcherNum(){
      System.out.print("Enter a number: ");
      fno = input.nextInt();
   }
   boolean isPrime(int p, int i){
      if(p==2){ return true;}
      if(p%i==0){ return false;}
      if(p<i*i){ return true;}
      else{ return isPrime(p,++i);}
   }

   int rotate(int n){
      String s=Integer.toString(n), p="";int len =s.length();
      for(int x=0;x<len;x++){ p+=s.charAt((len+x-1)%len);}
```

```
        return Integer.parseInt(p);
    }
    boolean checkFletcher(){
        boolean b=true;
        int l=(Integer.toString(fno)).length(),m=fno;
        for(int x=0;x<l;x++){
            for(int y=0;y<x;y++){ m=rotate(m);}
            if(!isPrime(m,2)){
                b=false;
                break;
            }
        }return b;
    }

    public static void main(String[] args) {
        FletcherNum f = new FletcherNum();
        System.out.println(f.checkFletcher());
    }
}
```

Output:

```
(1) Enter a number: 453
False
(2) Enter a number: 379
True
```

2. Happy Number

A Happy number is a number in which the eventual sum of the square of the digits of the number is equal to 1. This program checks if an input number is a Happy Number.

Eg. 19 => $1^2 + 9^2$ => 82, → 82 => $8^2 + 2^2$ => 68, → 68 => $6^2 + 8^2$ => 100, → 100 => $1^2 + 0^2 + 0^2$ => 1

Code:

```
import java.util.Scanner;
public class HappyNumber {
   int n;
   HappyNumber(){n=0;}
   void getNum(int nn){n=nn;}
   int sum_sq_digits(int x){
      if(x==0){return 0;}
      else{ return ((x%10)*(x%10)+sum_sq_digits(x/10));}
   }
   void isHappy(){
      int s=sum_sq_digits(n);
      while(s>9){ s=sum_sq_digits(s);}
      if(s==1){System.out.print(n+" is");}
      else{System.out.print(n+" isn't");}
      System.out.println(" a Happy Number.");
   }
   public static void main(String[] args) {
      HappyNumber H = new HappyNumber();
```

```
        Scanner input=new Scanner(System.in);
        System.out.print("Enter a number: ");
        H.getNum(input.nextInt());H.isHappy();
    }
}
```

Output:

```
(1) Enter a number: 12
12 isn't a Happy Number.
(2) Enter a number: 19
19 is a Happy Number.
```

3. Armstrong Number

A number is said to be Armstrong if the sum of its digits raised to the power of length of the number is equal to the number. Eg:

371 = 33 + 73 + 13

1634 = 14 + 64 + 34 + 44

54748 = 55 + 45 + 75 + 45 + 85

Code:

```
import java.util.Scanner;
public class ArmstrongNum {
    int n,l;
    ArmstrongNum(int nn){
        n=nn; l=Integer.toString(n).length();
    }
    int sum_pow(int i){
        if(i==0){ return 0;}
        else{ return (int)(Math.pow(i%10,l)+sum_pow(i/10));}
    }
    void isArmstrong(){
        if(n==sum_pow(n)){ System.out.println(n+" is an Armstrong Number!");}
        else{ System.out.println(n+" isn't an Armstrong Number.");}
    }
    public static void main(String[] args) {
        Scanner input =new Scanner(System.in);
        System.out.print("Enter a number: ");
        int x=input.nextInt();
        ArmstrongNum a = new ArmstrongNum(x);
        a.isArmstrong();
    }
}
```

Output:

```
(1) Enter a number: 2345
2345 isn't an Armstrong Number.
(2) Enter a number: 1634
1634 is an Armstrong Number!
```

4. Midpoint of Two Points

Uses the Midpoint formula to find the midpoint of two given points.

Eg. Midpoint of (2,2) and (4,4) is (3,3).

Code:

```
import java.util.Scanner;
public class Point {
   int x,y;
   Point(){x=y=0;}
   Point(int x, int y){this.x=x;this.y=y;}
   void readPoint(){
      Scanner input=new Scanner(System.in);
      System.out.print("Enter x-coordinate: ");
      x=input.nextInt();
      System.out.print("Enter y-coordinate: ");
      y=input.nextInt();
   }
   Point midPoint(Point B){
      Point A=new Point(((B.x+x)/2),((B.y+y)/2));
      return A;
   }
   String displayPoint(){
      return("Point Coordinates: "+"("+(x)+","+(y)+")"+"");
   }
   public static void main(String[] args) {
      Point P=new Point();Point Q=new Point();
      P.readPoint();Q.readPoint();Point M=P.midPoint(Q);
      System.out.println(M.displayPoint());
}}
```

Output:

```
Enter x-coordinate: 2
Enter y-coordinate: -3
Enter x-coordinate: -4
Enter y-coordinate: 7
Point Coordinates: (-1,2)
```

5. Merging Words

The program merges two given words one letter at a time and returns the result in uppercase. Eg. “Hello” + “World” = “HWEOLRLLOD”

Code:

```
import java.util.Scanner;
public class WordMerge {
    String wrd;int len;
    WordMerge(){ wrd="";len=0;}
    void feedWord(){
        Scanner input=new Scanner(System.in);
        System.out.print("Enter word: ");
        wrd=input.next().toUpperCase();
        len=wrd.length();
    }
    void mixWord(WordMerge p, WordMerge q){
        int i = 0; int j = 0;
        while(true){
           if(i < p.len && j < q.len){
              this.wrd += p.wrd.charAt(i);
              this.wrd += q.wrd.charAt(j);
           }
           else{ break;}
           i++;j++;
        }
        if(i < p.len)
           this.wrd += p.wrd.substring(i);
        if(j < q.len)
           this.wrd += q.wrd.substring(i);
    }
    void display(){ System.out.println(wrd);}
    public static void main(String[] args) {
        WordMerge obj1 = new WordMerge();
        WordMerge obj2 = new WordMerge();
        WordMerge result = new WordMerge();
        obj1.feedWord();obj2.feedWord();
        result.mixWord(obj1, obj2);
        result.display();
    }}
```

Output:

```
Enter word: Hello
Enter word: World
HWEOLRLLOD
```

~~~~~~

6. Sorting Palindrome Words of a Sentence

Code:

```
import java.util.Scanner;
public class PalinSort {
   String data;
   Scanner input = new Scanner(System.in);
   PalinSort(){}
   void readData(){
      System.out.print("Enter a sentence: ");
      data=input.nextLine();
   }
   boolean isPalin(String word){
      int l=word.length();
      if(word.charAt(0)!=word.charAt(l-1)){
         return false;}
      else{
         if(l>2){
            return isPalin(word.substring(1, l-1));}
```
~~~~~~

```
            else{ return true;}
        }
    }
    void print(){
        String p="";String q="";
        String[] a = data.split(" ");
        for(int x=0;x<a.length;x++){
            if(isPalin(a[x])){ p+=(a[x]+" ");}
            else{ q+=(a[x]+" ");}
        }
        System.out.println("Palindrome sorted sentence is: "+p+q);
    }
    public static void main(String[] args) {
        PalinSort p = new PalinSort();
        p.readData();p.print();
    }
}
```

Output:

```
Enter a sentence: My dad is bob and he speaks malayalam
Palindrome sorted sentence is: dad bob malayalam My is and he speaks
```

7. Checking if a Word is a Palindrome using a Stack

Code:

```
import java.util.Scanner;
public class PalinCheck_Stack {
    public static void main(String[] args) {
        System.out.print("Enter a word: ");
        Scanner input = new Scanner(System.in);
        String s = input.nextLine();
        Stack st = new Stack(s.length());
        int i=0; String r="";
        while(!st.isFull()){
            st.push(Character.toString(s.charAt(i)));
            i++;
        } i=0;
        while(!st.isEmpty()){
            r+=st.pop();
            i++;
        }
        if(s.equalsIgnoreCase(r)){
            System.out.println(s+" is a palindrome");
        }
        else{
            System.out.println(s+" is not a palindrome");
}}}
```

Output:

```
(1) Enter a word: malayalam
malayalam is a palindrome
(2) Enter a word: banana
banana is not a palindrome
```

~~~~~~

8. Stack

A Stack is a LIFO data structure – new items are “pushed onto the top” and items are also removed from the top. (Eg. A stack of books.)

Code:

```
import java.util.Scanner;
public class Stack {
   int top, capacity;
   String[] stack;
   Stack(){
      top = -1;
      capacity = 10;
      stack = new String[capacity];}
   Stack(int c){
      top = -1;
      capacity = c;
      stack = new String[capacity];}
   public boolean isFull(){
```
~~~~~~

```
      return top == capacity-1;}
   public boolean isEmpty(){
      return top == -1;}
   public String push(String data){
      if(isFull()){
         System.out.println("Stack is full.");}
      return stack[++top] = data;}
   public String pop(){
      if(isEmpty()){
         System.out.println("Stack is empty.");}
      return stack[top--];}
   public static void main(String[] args) {
      Scanner input=new Scanner(System.in);
      Stack st = new Stack(25);
      String in;
      for(int x=0;x<4;x++){
         System.out.print("Enter some object names to push into the stack: ");
         in = input.next();st.push(in);
      }
      System.out.println("Your stack looks like...");
      System.out.println();
      for(int i=0; i<4; i++){
         System.out.println(st.pop());}
   }
}
```

Output:

```
Enter some object names to push into the stack: book
Enter some object names to push into the stack: phone
Enter some object names to push into the stack: paper
Enter some object names to push into the stack: shirt
Your stack looks like...

shirt
paper
phone
book
```

9. Linear Queue

A Queue is a FIFO data structure – the first item to enter the queue is the first to leave it. This is the linear variant (Eg. a queue of people.)

Code:

```
import java.util.NoSuchElementException;
public class LinearQueue {
   int front, rear, nums[];
   LinearQueue(int size){
      this.front=this.rear=-1;
      this.nums = new int[size];
   }
   public void enqueue(int data){
      if(isFull()){
         throw new IllegalStateException("Queue is full ...Can't enqueue more.");
      }
      if(isEmpty()){ front++;}
      nums[++rear]=data;
   }
   public int dequeue(){
      if(isEmpty()){
         throw new NoSuchElementException("Queue is empty ...Nothing to
dequeue.");
      }
      int temp=nums[front];
      if(front==rear){ front=rear=-1;}
      else{ front++;}
      return temp;
   }
   public boolean isEmpty(){
      return front==-1;
   }
   private boolean isFull(){
      return rear==nums.length-1;
   }
   public int peek(){
      if(isEmpty()){
}
            throw new NoSuchElementException("Queue is empty ...Nothing to peek into.");
        } return nums[front];
    }
    void display(){     //For debugging purposes
        System.out.print("Queue(<-- way): ");
        //System.out.println(Arrays.toString(nums));
        for(int i=front;i<=rear;i++){
            System.out.print(nums[i]+" ");
        }
    }

    public static void main(String[] args) {
        LinearQueue q = new LinearQueue(10);
        q.enqueue(20);q.enqueue(25);q.enqueue(30);q.enqueue(35);
        q.enqueue(40);q.display();System.out.println();
        System.out.println(q.dequeue()+" left the queue");
        System.out.println(q.dequeue()+" left the queue");
        System.out.println(q.peek()+" is first in queue");
        System.out.println(q.dequeue()+" left the queue");
        System.out.println(q.dequeue()+" left the queue");
        System.out.println(q.peek()+" is first in queue");
        q.display();
    }
}
```

Output:

```
Queue(<-- way): 20 25 30 35 40
20 left the queue
25 left the queue
30 is first in queue
30 left the queue
35 left the queue
40 is first in queue
Queue(<-- way): 40
```

~~~~~~

10. Circular Queue

A circular queue optimizes storage by connecting the end to the front.

Code:

```java
import java.util.NoSuchElementException;
public class CircularQueue {
   int front,rear,nums[];
   CircularQueue(int size){
      this.front=this.rear=-1;
      this.nums = new int[size];
   }
   public void enqueue(int data){
      if(isFull()){
         throw new IllegalStateException("Queue is full ...Can't enqueue more.");
```
~~~~~~

```
        }
        else if(isEmpty()){ front++;}
        rear=(rear+1)%nums.length;
        nums[rear]=data;
    }
    public int dequeue(){
        if(isEmpty()){
            throw new NoSuchElementException();
        }
        int temp=nums[front];
        if(front==rear){ front=rear=-1;}
        else{
            front=(front+1)%nums.length;
        } return temp;
    }
    public int peek(){
        if(isEmpty()){
            throw new NoSuchElementException();
        } return nums[front];
    }
    public boolean isEmpty(){
        return front==-1;
    }
    private boolean isFull(){
        return (rear+1)%nums.length==front;
```

```
   }
   void display(){    //For debugging purposes
      System.out.print("Queue(<-- way): ");
      if(front>rear){
         for(int i=front;i<nums.length;i++){
            System.out.print(nums[i]+" ");
         }
      }
      for(int i=0;i<=rear;i++){
         System.out.print(nums[i]+" ");
      }
   }
   public static void main(String[] args) {
      CircularQueue q = new CircularQueue(5);
      q.enqueue(20);q.enqueue(25);q.enqueue(30);
      q.enqueue(35);q.enqueue(40);q.display();System.out.println();
      System.out.println(q.dequeue()+" left the queue");
      System.out.println(q.dequeue()+" left the queue");
      System.out.println(q.peek()+" is first in queue");
      q.enqueue(45);q.enqueue(50);
      System.out.println(q.dequeue()+" left the queue");
      System.out.println(q.dequeue()+" left the queue");
      System.out.println(q.peek()+" is first in queue");
      q.display();
}}
```

Output:

```
Queue(<-- way): 20 25 30 35 40
20 left the queue
25 left the queue
30 is first in queue
30 left the queue
35 left the queue
40 is first in queue
Queue(<-- way): 40 45 50
```

~~~~~~

11. Difference of two Sets/Arrays

This program removes the elements of one set from another.

Code:

```
import java.util.*;
public class SetDiffer {
   int[] set; int size;
   SetDiffer(int size){
      this.size=size;
      set=new int[size];
   }
   void getSet(){
      Scanner input=new Scanner(System.in);
      for(int x=0;x<size;x++){
```
~~~~~~

```
      System.out.print("Enter element "+(x+1)+" of this set: ");
      set[x]=input.nextInt();
   }
   System.out.println();
}
SetDiffer subtract(SetDiffer A){
   String p="";boolean b;int i;
   for(int x=0;x<this.size;x++){
      i=this.set[x];b=false;
      for(int y=0;y<A.size;y++){
         if(i==A.set[y]){b=true;}
      }
      if(b==false){ p+=Integer.toString(i)+" ";}
   }
   String[] arr=p.split(" ");
   SetDiffer d=new SetDiffer(arr.length);
   for(int x=0;x<d.size;x++){
      d.set[x]=Integer.parseInt(arr[x]);
   } return d;
}
public String toString(){
   String s="[";
   for(int x=0;x<this.size;x++){
      s+=(Integer.toString(this.set[x]))+",";
   }
```

```
        s=s.substring(0, s.length()-1);
        s+="]"; return s;
    }
    public static void main(String[] args) {
        SetDiffer D1=new SetDiffer(5);D1.getSet();
        SetDiffer D2=new SetDiffer(5);D2.getSet();
        SetDiffer D3=D1.subtract(D2);
        System.out.println(D3);
    }
}
```

Output:

```
Enter element 1 of this set: 3
Enter element 2 of this set: 5
Enter element 3 of this set: 8
Enter element 4 of this set: 1
Enter element 5 of this set: 4

Enter element 1 of this set: 7
Enter element 2 of this set: 4
Enter element 3 of this set: 1
Enter element 4 of this set: 6
Enter element 5 of this set: 9

[3,5,8]
```

12. Inferring a Date

Infers the date and month from a given day number for a particular year.

Example: If day number is 64 and the year is 2020, then the corresponding date would be: March 4, 2020 i.e. (31 + 29 + 4 = 64).

Code:

```
import java.util.Scanner;
public class InferDate {
   int n,d,m,y;
   InferDate(){ n=0;d=0;m=1;y=0;}
   void accept(){
      Scanner input=new Scanner(System.in);
      System.out.print("Enter year: ");
      y=input.nextInt();
      System.out.print("Enter day number: ");
      n=input.nextInt();
      if(y%4==0 || (y%100==0) && (y%400==0)){
         if(n>366) {
            System.out.print("Invalid day.");
            System.exit(0);
         }
      }
      else if(n>365){
         System.out.print("Invalid day.");
         System.exit(0);
```

```
        }
    }
    void dayToDate(){
        int[] days={0,31,28,31,30,31,30,31,31,30,31,30,31};
        if((y%100==0) && (y%400==0) || (y%4==0) && (y%100!=0)){
            days[2]++;
        } int num=n;
        while(num>days[m]){
            num-=days[m]; m++;
        } d=num;
    }
    void display(){
        String[]
months={"","January","February","March","April","May","June","July","August
","September","October","November","December"};
        System.out.println("Day "+n+" is "+months[m]+" "+d+", "+y);
    }
    public static void main(String[] args) {
        InferDate x = new InferDate();
        x.accept();x.dayToDate();x.display();
    }
}
```

Output:

```
(1) Enter year: 2020
Enter day number: 64
```

```
Day 64 is March 4, 2020
(2) Enter year: 2100
Enter day number: 318
Day 318 is November 14, 2100
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

13. Adding two Time Instances

Code:

```
import java.util.Scanner;
public class AddTime {
  Scanner input = new Scanner(System.in);
  int[] t = new int[2];
  AddTime(){ t[0]=t[1]=0;}
  void readTime(){
    System.out.print("Enter Hours: ");
    this.t[0]=input.nextInt();
    System.out.print("Enter Minutes: ");
    this.t[1]=input.nextInt();
  }
  AddTime add(AddTime X){
    this.t[0]=(this.t[0]+X.t[0]+(this.t[1]+X.t[1])/60)%24;
    this.t[1]=(this.t[1]+X.t[1])%60;
    return this;
  }
```

```
  void displayTime(){
    System.out.println("The Time is "+this.t[0]+" Hours and "+this.t[1]+" Minutes.");
  }

  public static void main(String[] args) {
    AddTime at1 = new AddTime();at1.readTime();
    AddTime at2 = new AddTime();at2.readTime();
    AddTime at3 =at2.add(at1);
    at3.displayTime();
  }
}
```

Output:

```
(1) Enter Hours: 21
Enter Minutes: 37
(2) Enter Hours: 6
Enter Minutes: 41
The Time is 4 Hours and 18 Minutes.
```

14. Magic Square

An [nxn] square matrix is said to be a Magic Square, if the sum of each row, each column and each diagonal is same and equal to n(n^2+1)/2.

Code:

```
import java.util.Scanner;
public class MagicSquare {
   int n, m[][], MagicVal;
   MagicSquare(int n){
      this.n=n;
      m=new int[n][n];
      MagicVal=n*(n*n+1)/2;
   }
   void ReadMatrix(){
      Scanner input=new Scanner(System.in);
      for(int i=0;i<n;i++){
         for(int j=0;j<n;j++){
            System.out.print("Enter element "+(j+1)+" of row "+(i+1)+": ");
            m[i][j]=input.nextInt();
         }
      }
   }
   void PrintMatrix(){
      for(int i=0;i<n;i++){
         for(int j=0;j<n;j++){
```

```
         System.out.print(m[i][j]+" ");
      }
      System.out.println();
   }
}
boolean isMagic(){
   int d1sum=0;int d2sum=0;
   for(int j=0;j<n;j++){
         d1sum+=m[j][j];d2sum+=m[n-(j+1)][n-(j+1)];
      }
   if(d1sum != MagicVal){ return false;}
   if(d2sum != MagicVal){ return false;}
   else{
      for(int i=0;i<n;i++){
         int csum=0;int rsum=0;
         for(int j=0;j<n;j++){
            rsum+=m[i][j];csum+=m[j][i];
         }
         if(csum != MagicVal){ return false;}
         if(rsum != MagicVal){ return false;}
      } return true;
   }
}

public static void main(String[] args) {
        MagicSquare msq = new MagicSquare(3);
        msq.ReadMatrix();
        System.out.println("Original matrix:");
        msq.PrintMatrix();
        if(msq.isMagic()){
            System.out.println("Matrix is a magic square:");
        }
        else {
            System.out.println("Matrix is not a magic square:");
        }
        System.out.println("Magic value was "+msq.MagicVal);
    }
}
```

Output:

```
Enter element 1 of row 1: 2
Enter element 2 of row 1: 7
Enter element 3 of row 1: 6
Enter element 1 of row 2: 9
Enter element 2 of row 2: 5
Enter element 3 of row 2: 1
Enter element 1 of row 3: 4
Enter element 2 of row 3: 3
Enter element 3 of row 3: 8
```

```
Original matrix:
2 7 6
9 5 1
4 3 8
Matrix is a magic square:
The Magic value was 15
```

~~~~~~

15. Saddle Point of a Matrix

A Saddle point is an element of a matrix such that it is the minimum element for the row to which it belongs and the maximum element for the column to which it belongs. A Saddle point for a given matrix is always unique and a matrix may not have one. This program evaluates the Saddle point of a matrix, if any.

Code:

```
import java.util.Scanner;
public class SaddlePoint {
   int n,smat[][];
   SaddlePoint(int n){
      this.n=n;
      smat=new int[n][n];
   }
   void getMatrix(){
      Scanner input=new Scanner(System.in);
      System.out.println("Enter only +ve integers!");
      for(int i=0;i<n;i++){
```
~~~~~~

```
      for(int j=0;j<n;j++){
         System.out.print("Enter element "+(j+1)+" of row "+(i+1)+": ");
         smat[i][j]=input.nextInt();
      }
   }
}
int min(int r){
   int min=0;int minval=smat[r][0];
   for(int j=0;j<n;j++){
      if(smat[r][j]<minval){
         minval=smat[r][j];min=j;
      }
   } return min;
}
int max(int c){
   int max=0;int maxval=smat[0][c];
   for(int i=0;i<n;i++){
      if(smat[i][c]>maxval){
         maxval=smat[i][c];max=i;
      }
   } return max;
}
int saddle(){
   for(int i=0;i<n;i++){
      if(smat[i][min(i)]==smat[max(min(i))][min(i)]){
```

```
                return smat[i][min(i)];
            }
        } return -999;
    }
    void display(){
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                System.out.print(smat[i][j]+" ");}
            System.out.println();
        }
    }

    public static void main(String[] args) {
        SaddlePoint sp = new SaddlePoint(3);sp.getMatrix();
        System.out.println("Matrix");sp.display();
        if(sp.saddle()>0){
            System.out.println("has Saddle point "+sp.saddle());
        }
        else{
            System.out.println("has no Saddle point");
        }
    }
}
```

Output:

```
(1) Enter only +ve integers!

Enter element 1 of row 1: 4

Enter element 2 of row 1: 5

Enter element 3 of row 1: 6

Enter element 1 of row 2: 7

Enter element 2 of row 2: 8

Enter element 3 of row 2: 9

Enter element 1 of row 3: 5

Enter element 2 of row 3: 1

Enter element 3 of row 3: 3

Matrix

4 5 6

7 8 9

5 1 3

has Saddle point 7
```

```
(2) Enter only +ve integers!

Enter element 1 of row 1: 1

Enter element 2 of row 1: 8

Enter element 3 of row 1: 4

Enter element 1 of row 2: 2

Enter element 2 of row 2: 9

Enter element 3 of row 2: 7

Enter element 1 of row 3: 6

Enter element 2 of row 3: 3
Enter element 3 of row 3: 5
Matrix
1 8 4
2 9 7
6 3 5
Has no Saddle point
```

16. Rotating a Matrix Upwards

This program moves each row of the given matrix upwards a desired number of times. Eg: $\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}$ ===|`> $\begin{bmatrix}4 & 5 & 6\\ 7 & 8 & 9\\ 1 & 2 & 3\end{bmatrix}$ after one rotation.

Code:

```
import java.util.Scanner;
public class Matrix_Reorder {
   int n, m, mat[][], p;
   Matrix_Reorder(int m, int n, int p){
      this.n=n;this.m=m;this.p=p;
      mat=new int[m][n];
   }
   void getMatrix(){
      Scanner input=new Scanner(System.in);
      for(int i=0;i<m;i++){
         for(int j=0;j<n;j++){
            System.out.print("Enter element "+(j+1)+" of row "+(i+1)+": ");
            mat[i][j]=input.nextInt();
         }
      }
   }
   void rotateUp(){
      int[] temp,store;
      for(int x=0;x<p;x++){
         store=mat[0];
         for(int i=m-1;i>=0;i--){
            temp=mat[i];mat[i]=store;store=temp;
         }
      }
   }
   void display(){
      for(int i=0;i<m;i++){
         for(int j=0;j<n;j++){
            System.out.print(mat[i][j]+" ");
         }
         System.out.println();
      }
   }
   public static void main(String[] args) {
      Scanner input=new Scanner(System.in);
      System.out.print("Enter number of rows: ");
```

```
        int rows=input.nextInt();
        System.out.print("Enter number of columns: ");
        int cols=input.nextInt();
        System.out.print("Enter number of rotations: ");
        int rots=input.nextInt();
        Matrix_Reorder mr = new Matrix_Reorder(rows,cols,rots);
        mr.getMatrix();
        System.out.println("Original matrix:");mr.display();System.out.println();
        mr.rotateUp();System.out.println("After "+(mr.p)+" upward rotations:");
        mr.display();
    }
}
```

Output:

```
Enter element 1 of row 1: 1
Enter element 2 of row 1: 2
Enter element 3 of row 1: 3
Enter element 1 of row 2: 4
Enter element 2 of row 2: 5
Enter element 3 of row 2: 6
Enter element 1 of row 3: 7
Enter element 2 of row 3: 8
Enter element 3 of row 3: 9
```

```
Original matrix:

1 2 3

4 5 6

7 8 9

After 2 upward rotations:

7 8 9

1 2 3

4 5 6
```

17. Matrix of Prime Numbers

This program fills a mxn matrix with the first (mxn) prime numbers: 2, 3, 5, 7, 11...

Code:

```
import java.util.Scanner;
public class PrimeMatrix {
   int n, m, pmat[][], currentPrime;
   PrimeMatrix(int m, int n){
      this.m=m;this.n=n;
      pmat=new int[m][n];
      currentPrime=1;
   }
   boolean isPrime(int x){
    int factors=0;
    for(int i=2;i<x;i++){
        if(x%i==0){ factors++;}
    }  return (factors==0);
}
int nextPrime(){
    currentPrime++; boolean b=false;
    while(!b){
        if(isPrime(currentPrime)){ b=true;}
        else{ currentPrime++;}
    } return currentPrime;
}
void fillPrime(){
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            pmat[i][j]=nextPrime();
        }
    }
}
void printMatrix(){
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            System.out.print(pmat[i][j]+" ");
        }
        System.out.println();
        }
    }

    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.print("Enter number of rows: ");
        int r=input.nextInt();
        System.out.print("Enter number of columns: ");
        int c=input.nextInt();
        PrimeMatrix pm = new PrimeMatrix(r,c);
        pm.fillPrime();pm.printMatrix();
    }
}
```

Output:

```
Enter number of rows: 3
Enter number of columns: 4
2   3 5   7
11   13 17 19
23 29 31 37
```

18. Tower of Hanoi

The Tower of Hanoi is a mathematical puzzle where we have three rods and n disks. The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1) Only one disk can be moved at a time.

2) Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.

3) No disk may be placed on top of a smaller disk.

This program, which is complex to solve using iteration is quite easy with recursion. It decodes the problem by computing the minimum required steps needed.

Code:

```
import java.util.Scanner;

public class TowerHanoi {

   static void towerHanoi(int nDisk, String from, String temp, String to){

      if(nDisk==1){

         System.out.println("Move disk 1 from "+from+" to "+to);

      }

      else{

         towerHanoi(nDisk-1,from,to,temp);

         System.out.println("Move disk "+nDisk+" from "+from+" to "+to);

         towerHanoi(nDisk-1,temp,from,to);

      }

   }

   public static void main(String[] args) {

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

```
        System.out.print("Enter number of disks: ");
        int n=input.nextInt();
        System.out.print("Enter name of initial tower: ");
        String from=input.next();
        System.out.print("Enter name of destination tower: ");
        String to=input.next();
        System.out.print("Enter name of intermediate tower: ");
        String temp=input.next();
        towerHanoi(n,from,temp,to);
    }
}
```

Output:

```
Enter number of disks: 3
Enter name of initial tower: A
Enter name of destination tower: C
Enter name of intermediate tower: B
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
```

19. Determinant

This program computes the determinant of a given square matrix using recursion.

Code:

```
import java.util.Scanner;
public class Determinant {
   Scanner input=new Scanner(System.in);
   double a[][];int n;
   Determinant(){}
   void getMat(){
      System.out.print("Enter the order of the Determinant: ");
      n=input.nextInt();
      a=new double[n][n];
      for(int i=0;i<a.length;i++){
         for(int j=0;j<a[i].length;j++){
            System.out.print("Enter element "+(j+1)+" of row "+(i+1)+": ");
            a[i][j] = input.nextDouble();
         }
      }
   }
   double[][] cofactor(double[][] mat, int r, int c){
      int l=mat.length;
      double[][] x=new double[l-1][l-1];
      int ii=0;
      for(int i=0;i<l;i++){
        if(i==r) continue;
        int jj=0;
        for(int j=0;j<l;j++){
            if(j==c) continue;
            x[ii][jj]=mat[i][j];
            jj++;
        }
        ii++;
    }
    return x;
}
double det(double M[][],int n){
    double result;
    if(n==2) result=M[0][0]*M[1][1]-M[1][0]*M[0][1];
    else{
        result=0;
        for(int j=0;j<n;j++){
            double[][] m=cofactor(M,0,j);
            result+=(Math.pow(-1.0, 2+j)*M[0][j]*det(m, n-1));
        }
    }
    return result;
}
public static void main(String[] args) {
    Determinant d=new Determinant();
```

```
        d.getMat();System.out.println("Determinant = "+d.det(d.a,d.n));
    }
}
```

Output:

```
Enter the order of the Determinant: 3
Enter element 1 of row 1: 2
Enter element 2 of row 1: -1
Enter element 3 of row 1: 4
Enter element 1 of row 2: 3
Enter element 2 of row 2: 0
Enter element 3 of row 2: 1
Enter element 1 of row 3: 2
Enter element 2 of row 3: 1
Enter element 3 of row 3: -1
Determinant = 5.0
```

20. Binary Search

This program searches for a number in an array of numbers in ascending order and returns its position in the array, if it exists in it.

Code:

```
import java.util.Scanner;
public class BinSearch {
    int[]a;
```

```
BinSearch(int n){
   a=new int[n];
}
void getArray(){
   Scanner input=new Scanner(System.in);
   for(int i=0;i<a.length;i++){
      System.out.print("Enter integer "+(i+1)+" of the array: ");
      a[i]=input.nextInt();
   }
}
void SelectionSort(){
   int i,j,smallt,t,position;
   for(i=0;i<a.length;i++){
      smallt = a[i]; position=i;
      for(j=i+1;j<a.length;j++){
         if(a[j]<smallt){
            smallt = a[j]; position = j;}}
      t=a[i]; a[i]=a[position]; a[position]=t;
   }
}
int Search(int l, int u, int val){
   int mid=(l+u)/2;
   if(l>u){ return -1;}
   if(val<a[mid]){
      return Search(l,mid-1,val);
```

```
        }
        if(val>a[mid]){
            return Search(mid+1,u,val);
        } return mid;
    }
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.print("Enter length of the array: ");
        int len=input.nextInt();
        BinSearch bs = new BinSearch(len);
        bs.getArray();bs.SelectionSort();
        System.out.print("Enter a number to search in the array: ");
        int num=input.nextInt();
        int x=bs.Search(0, len, num);
        if(x==-1){ System.out.println((num)+" wasn't found in the array");}
        else{ System.out.println(num+" lies in position "+(x+1)+" of the array");}
    }
}
```

Output:

```
Enter length of the array: 5
Enter integer 1 of the array: 12
Enter integer 2 of the array: 4
Enter integer 3 of the array: 7
Enter integer 4 of the array: 1
Enter integer 5 of the array: 9

Enter a number to search in the array: 12

12 lies in position 5 of the array
```

21. Shapes

The following classes demonstrate inheritance. Shapes is the superclass, having Rectangle and Circle as its subclasses. Rectangle further has a subclass named Square. Main() class Test demonstrates the inheritance between the classes.

Code:

```
public class Shape{
   String color="red"; boolean filled=true;
   Shape(){}
   Shape(String color, boolean filled){
      this.color=color;
      this.filled=filled;
   }
   String getColor(){return color;}
   void setColor(String color){this.color=color;}
   boolean isFilled(){return filled;}
   void setFilled(boolean filled){this.filled=filled;}
   public String toString(){
      return ("Shape[color="+color+",filled="+filled+"]");
   }
}
```

```
public class Circle extends Shape{
   double radius=1.0;
   Circle(){}
   Circle(double radius){this.radius=radius;}
   Circle(String color, boolean filled, double radius){
      super(color,filled);
      this.radius=radius;
   }
   double getRadius(){return radius;}
   double getArea(){return (3.14*radius*radius);}
   double getPerimeter(){return (6.28*radius);}
   void setRadius(double radius){this.radius=radius;}
   public String toString(){
      return ("Circle["+super.toString()+",radius="+radius+"]");
   }
}
public class Rectangle extends Shape{
   double width=1.0;double length=1.0;
   Rectangle(){}
   Rectangle(double width,double length){
      this.length=length;
      this.width=width;
   }
   Rectangle(String color, boolean filled, double width,double length){
      super(color,filled);
```

```
        this.length=length;
        this.width=width;
    }
    double getWidth(){return width;}
    double getLength(){return length;}
    double getArea(){return (width*length);}
    double getPerimeter(){return 2*(width+length);}
    void setWidth(double width){this.width=width;}
    void setLength(double length){this.length=length;}
    public String toString(){
        return
("Rectangle["+super.toString()+",width="+width+",length="+length+"]");
    }
}
public class Square extends Rectangle{
    Square(){}
    Square(double side){super(side,side);}
    Square(String color, boolean filled, double side){
        super(color,filled,side,side);
    }
    double getSide(){return width;}
    void setSide(double side){
        this.width=side;
        this.length=side;
    }
    void setWidth(double side){this.width=side;}
    void setLength(double side){this.length=side;}
    public String toString(){
        return ("Square["+super.toString()+"]");
    }
}
public class Test {
    public static void main(String[] args) {
        Circle c=new Circle();
        Square s=new Square();
        System.out.println(c);
        System.out.println(s);
    }
}
```

Output:

```
Circle[Shape[color=red,filled=true],radius=1.0]
Square[Rectangle[Shape[color=red,filled=true],width=1.0,length=1.0]]
```

22. Product Description Interface

This program demonstrates the usage of an Interface 'Taxable' to simplify the structure of a model which calculates the cost of standard transactions, including the tax charged.

Code:

```
class Goods {
   String description;
   double price;
   Goods(String d, double p) {
         description = d;
         price = p;
   }
   String display() {
         return ("Item: "+description+" costs "+price);
   }
}

interface Taxable {
   double taxRate = 0.08;
   double calculateTax();
}

class Book extends Goods implements Taxable {
   String author;
   double deliveryCharges;
   Book (String d, double p, String a, double dc) {
      super(d, p);
        this.author = a;
        this.deliveryCharges = dc;
   }
   String display() {
        return (super.display()+" plus delivery charges "+this.deliveryCharges+"
plus Tax "+calculateTax());
   }
   public double calculateTax() {
        return (price+this.deliveryCharges)*taxRate;
   }
}

class Test {
   public static void main(String[] args) {
        Book b = new Book("Pride and Prejudice", 275.0,"Jane Austin",25.0);
        System.out.println(b.display());
   }
}
```

Output:

```
Item: Pride and Prejudice costs 275.0 plus delivery charges 25.0 plus Tax 24.0
```

23. Employee Model

This program demonstrates the use of Abstract Classes in a simple inheritance model.

Code:

```
abstract public class Employee {
   String firstName, lastName;
   int joinDay, joinMonth, joinYear;
   Employee(String f,String l,int d,int m,int y){
      this.firstName = f;
      this.lastName = l;
      this.joinDay = d;
      this.joinMonth = m;
      this.joinYear = y;
   }
   abstract void display();
}
public class USEmployee extends Employee {
   USEmployee(String f, String l, int d, int m, int y){
      super(f,l,d,m,y);
   }
   void display(){
      System.out.println(this.firstName+" "+this.lastName+" joined on "+this.joinMonth+"/"+this.joinDay+"/"+this.joinYear);
   }
}
```

```
public class IndianEmployee extends Employee {
   IndianEmployee(String f, String l, int d, int m, int y){
      super(f,l,d,m,y);
   }
   void display(){
      System.out.println(this.firstName+" "+this.lastName+" joined on "+this.joinDay+"/"+this.joinMonth+"/"+this.joinYear);
   }
}
public class Test {
   public static void main(String[] args) {
      USEmployee usE  = new USEmployee("John","Doe", 12, 01, 2010);
      usE.display();
      IndianEmployee iE  = new IndianEmployee("Sarang","Galada", 12, 01, 2010);
      iE.display();
   }
}
```

Output:

```
John Doe joined on 1/12/2010
Sarang Galada joined on 12/1/2010
```

24. Matrix Subtraction

Code:

```
import java.util.Scanner;
public class Matrix_Subtract {
   int n, m, mat[][];
   Matrix_Subtract(int m, int n){
      this.n=n;this.m=m;
      mat=new int[m][n];
   }
   void getMatrix(){
      Scanner input=new Scanner(System.in);
      for(int i=0;i<m;i++){
         for(int j=0;j<n;j++){
            System.out.print("Enter element "+(j+1)+" of row "+(i+1)+": ");
            mat[i][j]=input.nextInt();
         }
      }
   }
   Matrix_Subtract subtract(Matrix_Subtract S){
      Matrix_Subtract M = new Matrix_Subtract(m,n);
      for(int i=0;i<m;i++){
         for(int j=0;j<n;j++){
            M.mat[i][j] = this.mat[i][j] - S.mat[i][j];
         }
```

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

    public static void main(String[] args) {
        Matrix_Subtract m1 = new Matrix_Subtract(3,4);m1.getMatrix();
        Matrix_Subtract m2 = new Matrix_Subtract(3,4);m2.getMatrix();
        Matrix_Subtract m = m1.subtract(m2);
        m1.display();System.out.println("  -  ");m2.display();System.out.println("  =  ");
        m.display();
    }
}
```

Output:

```
Enter element 1 of row1: 1

Enter element 2 of row1: 2

Enter element 3 of row1: 3
Enter element 4 of row 1: 4
Enter element 1 of row 2: 5
Enter element 2 of row 2: 6
Enter element 3 of row 2: 7
Enter element 4 of row 2: 8
Enter element 1 of row 3: 9
Enter element 2 of row 3: 1
Enter element 3 of row 3: 2
Enter element 4 of row 3: 3
Enter element 1 of row 1: 4
Enter element 2 of row 1: 5
Enter element 3 of row 1: 6
Enter element 4 of row 1: 7
Enter element 1 of row 2: 8
Enter element 2 of row 2: 9
Enter element 3 of row 2: 1
Enter element 4 of row 2: 2
Enter element 1 of row 3: 3
Enter element 2 of row 3: 4
Enter element 3 of row 3: 5
Enter element 4 of row 3: 6
```

```
1234
5678
9123
 -
4567
8912
3456
 =
-3-3-3-3
-3-366
6-3-3-3
```

25. Sorting Vowels in a Word

This program sorts and brings all the vowels in a word to the front. It also displays the number of vowels and consonants present.

Code:

```
import java.util.Scanner;
public class VowelSort {
   Scanner input=new Scanner(System.in);
   String wrd,newwrd;
   VowelSort(){}
   void readword(){
      System.out.print("Enter a word: ");
```

```
    wrd = (input.next()).toUpperCase();
}
void freq(){
    int v=0;char ch;
    for(int x=0;x<wrd.length();x++){
        ch=wrd.charAt(x);
        if("AEIOU".indexOf(ch)!=-1){
            v++;
        }
    }
    System.out.println("No. of vowels: "+v);
    System.out.println("No. of consonants: "+(wrd.length()-v));
}
void arrange(){
    String v="";String c="";char ch;
    for(int x=0;x<wrd.length();x++){
        ch=wrd.charAt(x);
        if("AEIOU".indexOf(ch)!=-1){
            v+=ch;
        }
        else{
            c+=ch;
        }
    }
    newwrd=v+c;
    }
    void display(){
        System.out.println("Original word: "+wrd);
        System.out.println("After sorting vowels & consonants: "+newwrd);
    }
    public static void main(String[] args) {
        VowelSort x = new VowelSort();
        x.readword();x.freq();x.arrange();x.display();
    }
}
```

Output:

```
Enter a word: miscellaneous
No. of vowels: 6
No. of consonants: 7
Original word: MISCELLANEOUS
After sorting vowels & consonants: IEAEOUMSCLLNS
```

26. Jumbling a Sentence

This program swaps alternate letters in a word and return it in uppercase.

Code:

```
import java.util.Scanner;import java.util.Arrays;
public class StringSwap {
    Scanner input = new Scanner(System.in);
    String Sent;
    StringSwap(){
        this.Sent="";
    }
    void getSentence(){
        System.out.print("Enter a sentence: ");
        Sent = input.nextLine();
    }

    String SwapAdj(String word){
        char t; int l = word.length();
        char[] a = new char[l];
        for(int x=0;x<l;x++){
            a[x]=word.charAt(x);
        }
        for(int y=0;y<2*(l/2);y+=2){
            t=a[y];
            a[y]=a[y+1];
            a[y+1]=t;
        }
        String swap = new String(a);
        return swap;
    }

    void printSwapSent(){
        String[] words = Sent.split(" ");
        for(int x=0;x<words.length;x++){
            words[x]=SwapAdj(words[x]);
        }String swapstring = Arrays.toString(words);
        swapstring=swapstring.substring(1, swapstring.length()-1);swapstring=swapstring.replaceAll(",", "");
        System.out.println(swapstring);
    }

    public static void main(String[] args) {
        StringSwap s = new StringSwap();
        s.getSentence();s.printSwapSent();
    }
}
```

Output:

```
Enter a sentence: I am now in a very good mood
I ma onw ni a evyr ogdo omdo
```