Cyclesheet- 2 Java Lab

Name: Sankalp Mukim

Registration Number: 20BDS0128

Subject: Java (Lab)

Slot: L21+L22

Link to all the code: https://github.com/sankalpmukim/jafva-lab-fall-sem-2021

Contents

Question 10:

An interface stores an array of 84-2-1 codes of the decimal digits from 0 to 9 stored in random order.

0101	0100	0000	1111	1000	1011	1001	0111	1010	0110

Define a nested class inside the interface with a method to create the following 2-D array. The decimal equivalent should be found by adding the weights of all the 1's in the corresponding 84-2-1code. Eg., For the code 1011, decimal equivalent = 1*8 + 0*4 + 1*(-2) + 1*(-1) = 5

84-2-1 Code	Decimal Digit
0101	
0100	
0000	
1111	
1000	
1011	
1001	
0111	
1010	
0110	

Derive a class “Class8421” from the nested class that uses the above table to generate the 84-2-1 code of a given number, n. Eg., If the number is 395, its 84-2-1 code is 0101 1111 1011 Define another class “Complement9” that inherits “Class8421” with two methods.

i) First method to find the 9’s complement of the number by changing 1’s to 0’s and 0’s to 1’s. Eg., 0101 1111 1011 should change to 1010 0000 0100

ii) Second method to find the decimal equivalent of the complemented 84-2-1 code using the table. Eg., Decimal equivalent of 1010 0000 0100 is 604

Code:

Class8421:

```java
package Q10;

public class Class8421 implements Decode8421 {
    String code;

    String numToCode(int n) {
        NestedClass obj = new NestedClass();
        for (String[] arr : obj.twoDArray) {
            if (Integer.parseInt(arr[1]) == n) {
                return arr[0];
            }
        }
        return "error";
    }

    public Class8421(int n) {
        // number of digits in n
        int temp = n, count = 0;
        while (temp > 0) {
            count++;
            temp /= 10;
        }

        temp = n;
        String[] codeArr = new String[count];
        for (int i = codeArr.length - 1; i >= 0; i--) {
            codeArr[i] = numToCode(temp % 10);
            temp /= 10;
        }
        code = codeArr[0];
        for (int i = 1; i < codeArr.length; i++) {
            code += " ";
            code += codeArr[i];
        }
    }
}
```

Complement9:

```java
package Q10;

public class Complement9 extends Class8421 {
    public Complement9(int n) {
        super(n);
    }

    public void ninesComplement() {
        char[] charArr = code.toCharArray();
        for (int i = 0; i < charArr.length; i++) {
            if (charArr[i] == ' ') {
                continue;
            } else {
                if (charArr[i] == '1') {
                    charArr[i] = '0';
                } else {
                    charArr[i] = '1';
                }
            }
        }
        code = String.valueOf(charArr);
    }

    public int findDecimal() {
        String[] indiviDigits = code.split(" ");
        int num = 0;

        for (String string : indiviDigits) {
            num *= 10;
            num += NestedClass.CodeToDecimal(string);
        }
        return num;
    }
}
```

Decode8421

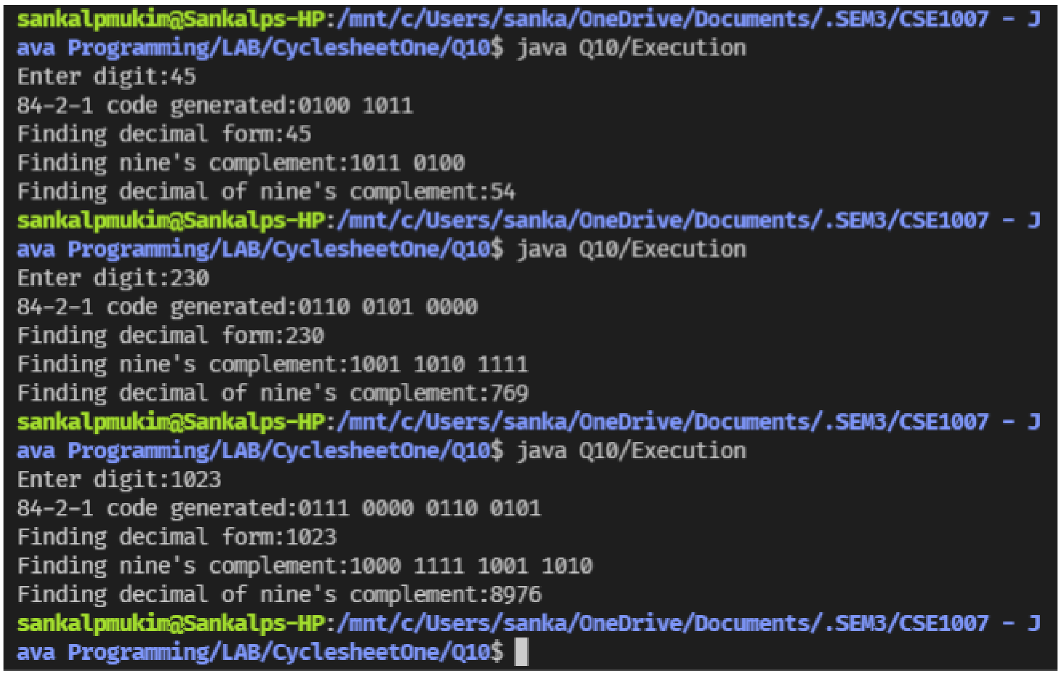
```
package Q10;

public interface Decode8421 {
    String[] arr = { "0101", "0100", "0000", "1111", "1000", "1011",
"1001", "0111", "1010", "0110" };

    public static class NestedClass implements Decode8421 {
        String[][] twoDArray;

        public static int CodeToDecimal(String num) {
            int dig0, dig1, dig2, dig3;
            dig0 = Character.getNumericValue(num.charAt(0)) * 8;
            dig1 = Character.getNumericValue(num.charAt(1)) * 4;
            dig2 = Character.getNumericValue(num.charAt(2)) * -2;
            dig3 = Character.getNumericValue(num.charAt(3)) * -1;
            return dig0 + dig1 + dig2 + dig3;
        }

        public NestedClass() {
            this.twoDArray = new String[10][2];
            int i = 0;
            for (String string : arr) {
                twoDArray[i] = new String[2];
                twoDArray[i][0] = string;
                twoDArray[i][1] =
Integer.toString(CodeToDecimal(string));
                i++;
            }
        }
    }
}
```

Execution:

```
package Q10;

import java.util.Scanner;

public class Execution {
    public static void main(String[] args) {
        System.out.print("Enter digit:");
        Scanner in = new Scanner(System.in);
        int num = in.nextInt();
        in.close();
        Complement9 object = new Complement9(num);
        System.out.println("84-2-1 code generated:" + object.code);
        System.out.println("Finding decimal form:" +
object.findDecimal());
```

```
        object.ninesComplement();
        System.out.println("Finding nine's complement:" +
object.code);
        System.out.println("Finding decimal of nine's complement:" +
object.findDecimal());

    }
}
```

Output:



```
sankalpmukim@Sankalps-HP:/mnt/c/Users/sanka/OneDrive/Documents/.SEM3/CSE1007 - J
ava Programming/LAB/CyclesheetOne/Q10$ java Q10/Execution
Enter digit:45
84-2-1 code generated:0100 1011
Finding decimal form:45
Finding nine's complement:1011 0100
Finding decimal of nine's complement:54
sankalpmukim@Sankalps-HP:/mnt/c/Users/sanka/OneDrive/Documents/.SEM3/CSE1007 - J
ava Programming/LAB/CyclesheetOne/Q10$ java Q10/Execution
Enter digit:230
84-2-1 code generated:0110 0101 0000
Finding decimal form:230
Finding nine's complement:1001 1010 1111
Finding decimal of nine's complement:769
sankalpmukim@Sankalps-HP:/mnt/c/Users/sanka/OneDrive/Documents/.SEM3/CSE1007 - J
ava Programming/LAB/CyclesheetOne/Q10$ java Q10/Execution
Enter digit:1023
84-2-1 code generated:0111 0000 0110 0101
Finding decimal form:1023
Finding nine's complement:1000 1111 1001 1010
Finding decimal of nine's complement:8976
sankalpmukim@Sankalps-HP:/mnt/c/Users/sanka/OneDrive/Documents/.SEM3/CSE1007 - J
ava Programming/LAB/CyclesheetOne/Q10$
```

Question 11:

Define an interface BinaryTree that includes two methods to perform insertion and level order traversal (ie., breadth-first traversal). Define a class ArrayBinaryTree that implements the interface so as to perform the binary tree operations in an array. Define another class LinkedBinaryTree that implements the above interface so as to perform the binary tree operations with dynamically created nodes. Invoke these methods from the main class. The main class should be defined in the package "mainpackage" in a directory and the interface should be defined in the package "parentpackage" in a different directory and the two classes should be defined in the subpackage "childpackage" created inside "parentpackage".

Code:

parentpackage:

BinaryTree:

```
package parentpackage;

public interface BinaryTree {
    void insertion(int n, int pos, boolean set_left);

    void levelOrderTraversal();
}
```

parentpackage.childpackage:

ArrayBinaryTree:

```
package parentpackage.childpackage;

import parentpackage.BinaryTree;

public class ArrayBinaryTree implements BinaryTree {
    public int[] arr;

    public ArrayBinaryTree(int n) {
        arr = new int[n];
    }

    public void insertion(int n, int parent, boolean set_left) {
        if (parent != -1 && arr[parent] == 0) {
            System.out.println("Parent not found at " + parent + ",
therefore cannot set " + n);
        } else if (parent == -1) {
            arr[0] = n;
        } else if (set_left) {
            System.out.println("Element succesfully added left of " +
parent + ":" + n);
            arr[(parent * 2) + 1] = n;
        } else {
            System.out.println("Element succesfully added right of "
+ parent + ":" + n);
            arr[(parent * 2) + 2] = n;
        }
    }

    public void levelOrderTraversal() {
        System.out.println("Level order traversal of linked binary
tree!:");
        System.out.print("[\n\t");
        for (int i : arr) {
            System.out.print("" + i + ", ");
        }
        System.out.println("\n]");
    }
}
```

Node:

```
package parentpackage.childpackage;

public class Node {
    public int value;
```

```java
    public Node left;
    public Node right;

    public Node(int n) {
        value = n;
        left = null;
        right = null;
    }
}
```

LinkedListNode:

```java
package parentpackage.childpackage;

public class LinkedListNode {
    public Node node;
    public LinkedListNode next;
    public static LinkedListNode front = null;

    public LinkedListNode(Node x) {
        node = x;
        next = null;
    }

    public static void add(Node x) {
        LinkedListNode temp = front;
        if (temp == null) {
            front = new LinkedListNode(x);
            return;
        }
        while (temp.next != null) {
            temp = temp.next;
        }
        temp.next = new LinkedListNode(x);
    }

    public static Node remove() {
        Node temp = front.node;
        front = front.next;
        return temp;
    }
}
```

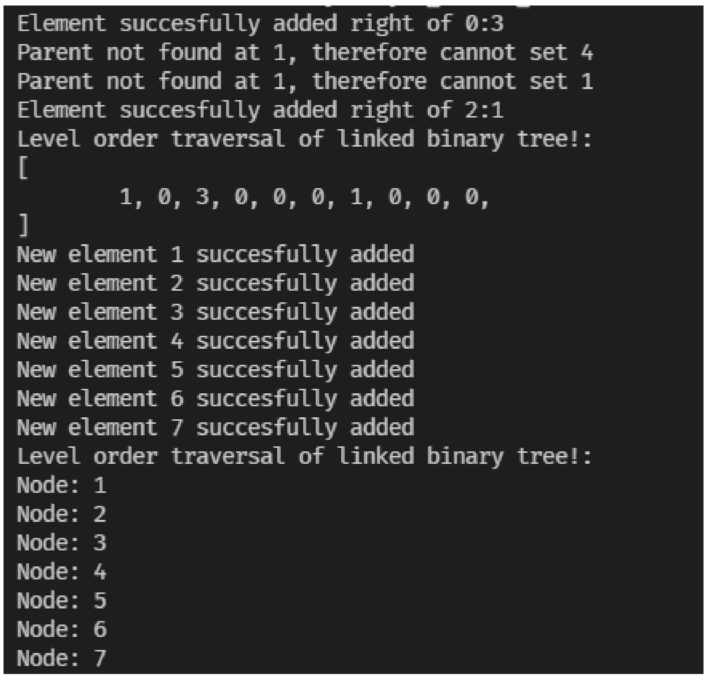
LinkedBinaryTree:

```java
package parentpackage.childpackage;

import parentpackage.BinaryTree;

public class LinkedBinaryTree implements BinaryTree {
    public Node root;

    public LinkedBinaryTree(int n) {
        System.out.println("New element " + n + " succesfully
added");
        root = new Node(n);
    }

    public static Node findElementNode(Node root, int n) {
        if (root == null) {
            return null;
        }
        if (root.value == n) {
            return root;
        }
        Node leftSide = findElementNode(root.left, n);
        if (leftSide != null) {
            return leftSide;
        }
        Node rightSide = findElementNode(root.right, n);
        return rightSide;
    }

    public void insertion(int n, int parent, boolean set_left) {
        Node parentNode = findElementNode(root, parent);
        if (parentNode == null) {
            System.out.println("Parent element not found");
            return;
        }
        if (set_left) {
            parentNode.left = new Node(n);
        } else {
            parentNode.right = new Node(n);
        }
        System.out.println("New element " + n + " succesfully
added");
    }

    public void levelOrderTraversal() {
        System.out.println("Level order traversal of linked binary
tree!:");
        LinkedListNode.front = new LinkedListNode(this.root);
        while (LinkedListNode.front != null &&
LinkedListNode.front.node != null) {
            Node curr = LinkedListNode.remove();
            System.out.println("Node: " + curr.value);
            LinkedListNode.add(curr.left);
            LinkedListNode.add(curr.right);
        }
    }
}
```

mainpackage:

```
package mainpackage;

import parentpackage.BinaryTree;
import parentpackage.childpackage.ArrayBinaryTree;
import parentpackage.childpackage.LinkedBinaryTree;

public class Q11 {
    public static void main(String[] args) {
        BinaryTree arrTree = new ArrayBinaryTree(10);
        arrTree.insertion(1, -1, true);
        arrTree.insertion(3, 0, false);
        arrTree.insertion(4, 1, true);
        arrTree.insertion(1, 1, false);
        arrTree.insertion(1, 2, false);

        arrTree.levelOrderTraversal();

        BinaryTree linkedTree = new LinkedBinaryTree(1);
        linkedTree.insertion(2, 1, true);
        linkedTree.insertion(3, 1, false);
        linkedTree.insertion(4, 2, true);
        linkedTree.insertion(5, 2, false);
        linkedTree.insertion(6, 3, true);
        linkedTree.insertion(7, 3, false);

        linkedTree.levelOrderTraversal();
    }
}
```

Output:



```
Element succesfully added right of 0:3
Parent not found at 1, therefore cannot set 4
Parent not found at 1, therefore cannot set 1
Element succesfully added right of 2:1
Level order traversal of linked binary tree!:
[
        1, 0, 3, 0, 0, 0, 1, 0, 0, 0,
]
New element 1 succesfully added
New element 2 succesfully added
New element 3 succesfully added
New element 4 succesfully added
New element 5 succesfully added
New element 6 succesfully added
New element 7 succesfully added
Level order traversal of linked binary tree!:
Node: 1
Node: 2
Node: 3
Node: 4
Node: 5
Node: 6
Node: 7
```

Question 12:

Define a class Line with 4 instance variables x1, y1 (cartesian coordinates of point 1), x2 and y2 (cartesian coordinates of point2). The constructor of the class should throw NumberFormatException if the two points are same. Derive the class Triangle from Line with two Line objects as its instance variables. Trigger TriangleNotPossibleException if the end point of one line is not the starting point of the other. In either case, the user should be prompted again to enter the input. At last, print the number of additional attempts made until a valid triangle is formed.

Code:

Exceptions:

TriangleNotFoundException:

```
package Q12.Exceptions;

public class TriangleNotPossibleException extends Exception {
    public TriangleNotPossibleException(String msg) {
        super(msg);
    }
}
```

Line:

```
package Q12;

public class Line {
    int x1, y1, x2, y2;

    public Line(int X1, int Y1, int X2, int Y2) throws
NumberFormatException {
        x1 = X1;
        y1 = Y1;
        x2 = X2;
        y2 = Y2;
        if (x1 == x2 && y1 == y2) {
            throw new NumberFormatException("Both points are same");
        }
    }

}
```

Triangle:

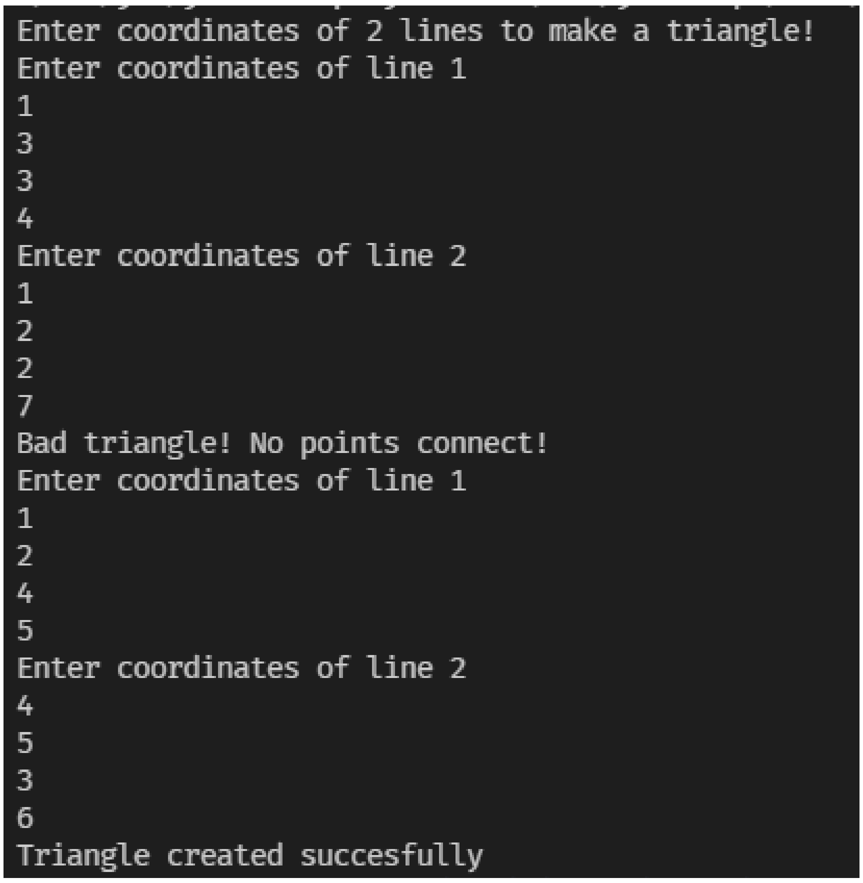
```
package Q12;

import Q12.Exceptions.TriangleNotPossibleException;

public class Triangle {
    Line l1, l2;

    public boolean pointsEqual(int x1, int y1, int x2, int y2) {
        return x1 == x2 && y1 == y2;
    }

    public Triangle(Line line1, Line line2) throws
TriangleNotPossibleException {
        if ((pointsEqual(line1.x1, line1.y1, line2.x1, line2.y1) ||
pointsEqual(line1.x1, line1.y1, line2.x2, line2.y2)
                || pointsEqual(line1.x2, line1.y2, line2.x1,
line2.y1)
                || pointsEqual(line1.x2, line1.y2, line2.x2,
line2.y2)) == false) {
            throw new TriangleNotPossibleException("Points not
connected");
        }
        l1 = line1;
        l2 = line2;
    }

    String success() {
        return ("Triangle created succesfully");
    }
}
```

Execution:

```
package Q12;

import java.util.Scanner;

import Q12.Exceptions.TriangleNotPossibleException;
```

```
public class Execution {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter coordinates of 2 lines to make a
triangle!");

        boolean line1 = true, line2 = true, triangle = true;
        Line l1 = null, l2 = null;
        Triangle t = null;
        while (triangle) {
            triangle = false;
            line1 = true;
            line2 = true;
            while (line1) {
                line1 = false;
                System.out.println("Enter coordinates of line 1");
                int x1 = in.nextInt(), y1 = in.nextInt(), x2 =
in.nextInt(), y2 = in.nextInt();
                try {
                    l1 = new Line(x1, y1, x2, y2);
                } catch (NumberFormatException e) {
                    System.out.println("Bad line! Points at the same
place! Try again!");
                    line1 = true;
                }
            }
            while (line2) {
                line2 = false;
                System.out.println("Enter coordinates of line 2");
                int x1 = in.nextInt(), y1 = in.nextInt(), x2 =
in.nextInt(), y2 = in.nextInt();
                try {
                    l2 = new Line(x1, y1, x2, y2);
                } catch (NumberFormatException e) {
                    System.out.println("Bad line! Points at the same
place! Try again!");
                    line2 = true;
                }
            }
            try {
                t = new Triangle(l1, l2);
            } catch (TriangleNotPossibleException e) {
                triangle = true;
                System.out.println("Bad triangle! No points
connect!");
            }
        }
        in.close();
        System.out.println(t.success());
    }
}
```

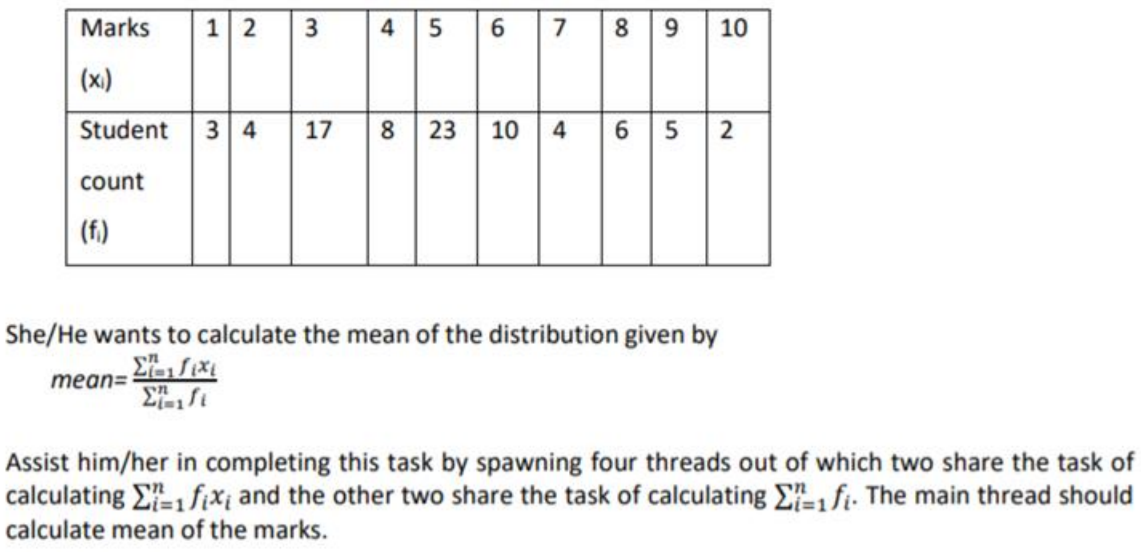
Output:



```
Enter coordinates of 2 lines to make a triangle!
Enter coordinates of line 1
1
3
3
4
Enter coordinates of line 2
1
2
2
7
Bad triangle! No points connect!
Enter coordinates of line 1
1
2
4
5
Enter coordinates of line 2
4
5
3
6
Triangle created succesfully
```

Question 13:

A faculty evaluates quiz and stores the count of students who have scored 1 mark, 2marks and so on, as follows.



Marks (x_i)	1	2	3	4	5	6	7	8	9	10
Student count (f_i)	3	4	17	8	23	10	4	6	5	2

She/He wants to calculate the mean of the distribution given by

$$mean=\frac{\sum_{i=1}^{n} f_i x_i}{\sum_{i=1}^{n} f_i}$$

Assist him/her in completing this task by spawning four threads out of which two share the task of calculating $\sum_{i=1}^{n} f_i x_i$ and the other two share the task of calculating $\sum_{i=1}^{n} f_i$. The main thread should calculate mean of the marks.

Code:

Numerator:

```java
package Q13;

public class Numerator extends Thread {
    public int sum;
    public boolean odds;
    public boolean notDone;
    public int[] marks;
    public int[] studentCount;

    public Numerator(boolean odds, int[] marks, int[] studentCount) {
        this.sum = 0;
        this.odds = odds;
        this.notDone = true;
        this.marks = marks;
        this.studentCount = studentCount;
    }

    public void run() {
        int startVal = this.odds ? 1 : 0;
        for (int i = startVal; i < marks.length; i += 2) {
            sum += (marks[i] * studentCount[i]);
        }
        notDone = false;
    }
}
```

Denominator:

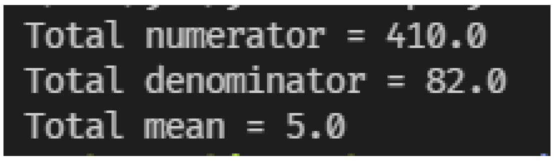
```java
package Q13;

```

```java
public class Denominator extends Thread {
    public int sum;
    public boolean odds;
    public boolean notDone;
    public int[] marks;
    public int[] studentCount;

    public Denominator(boolean odds, int[] studentCount) {
        this.sum = 0;
        this.odds = odds;
        this.notDone = true;
        this.studentCount = studentCount;
    }

    public void run() {
        int startVal = this.odds ? 1 : 0;
        for (int i = startVal; i < studentCount.length; i += 2) {
            sum += (studentCount[i]);
        }
        notDone = false;
    }
}
```

Execution:

```java
package Q13;

public class Execution {
    public static void main(String[] args) {
        int[] studentCount = { 3, 4, 17, 8, 23, 10, 4, 6, 5, 2 };
        int[] marks = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

        Numerator n1 = new Numerator(true, marks, studentCount);
        Numerator n2 = new Numerator(false, marks, studentCount);
        n1.start();
        n2.start();

        Denominator d1 = new Denominator(true, studentCount);
        Denominator d2 = new Denominator(false, studentCount);
        d1.start();
        d2.start();

        while (n1.notDone || n2.notDone || d1.notDone || d2.notDone)
{
            try {
                n1.join();
                n2.join();
                d1.join();
                d2.join();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        double numeratorTotal = n1.sum + n2.sum;
        double denominatorTotal = d1.sum + d2.sum;

        System.out.println("Total numerator = " + numeratorTotal);
        System.out.println("Total denominator = " +
denominatorTotal);
        System.out.println("Total mean = " + (numeratorTotal /
denominatorTotal));
    }
}
```

Output:



```
Total numerator = 410.0
Total denominator = 82.0
Total mean = 5.0
```

Question 14:

Three queues are maintained at a college canteen – one for faculty, another for male students and the last queue for female students. Assume there are only 4 different dishes prepared in the canteen costing Rs. 50, Rs. 75, Rs. 100 and Rs. 150 respectively. Bill processing should be done in order as follows - 3 faculty first, followed by one male student and then one female student and this pattern should continue until all the customer requests are processed. For each customer, read the number of items in each dish and calculate the bill and print. Print the total number of faculty and students (including both boys and girls) who visited the canteen on that day.

Code:

Male:

```
package Q14;

public class Male {
    public int numDish1;
    public int numDish2;
    public int numDish3;
    public int numDish4;

    public Male(int dish1, int dish2, int dish3, int dish4) {
        this.numDish1 = dish1;
        this.numDish2 = dish2;
        this.numDish3 = dish3;
        this.numDish4 = dish4;
    }
```

```java
    public int total() {
        return (numDish1 * 50) + (numDish2 * 75) + (numDish3 * 100) +
(numDish4 * 150);
    }
}
```

Female:

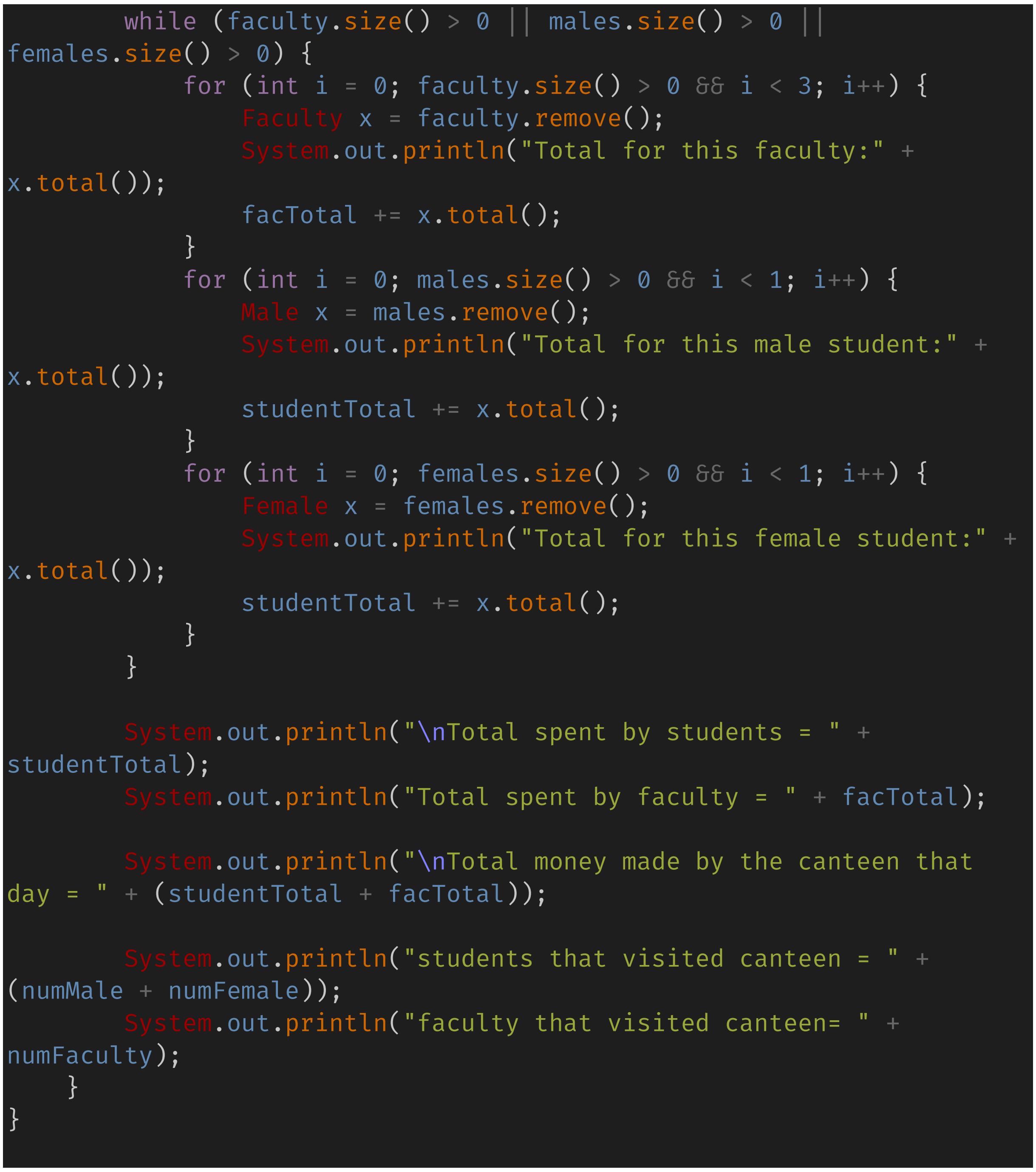
```java
package Q14;

public class Female {
    public int numDish1;
    public int numDish2;
    public int numDish3;
    public int numDish4;

    public Female(int dish1, int dish2, int dish3, int dish4) {
        this.numDish1 = dish1;
        this.numDish2 = dish2;
        this.numDish3 = dish3;
        this.numDish4 = dish4;
    }

    public int total() {
        return (numDish1 * 50) + (numDish2 * 75) + (numDish3 * 100) +
(numDish4 * 150);
    }
}
```

Faculty:

```java
package Q14;

public class Faculty {
    public int numDish1;
    public int numDish2;
    public int numDish3;
    public int numDish4;

    public Faculty(int dish1, int dish2, int dish3, int dish4) {
        this.numDish1 = dish1;
        this.numDish2 = dish2;
        this.numDish3 = dish3;
        this.numDish4 = dish4;
    }

    public int total() {
        return (numDish1 * 50) + (numDish2 * 75) + (numDish3 * 100) +
(numDish4 * 150);
    }
}
```

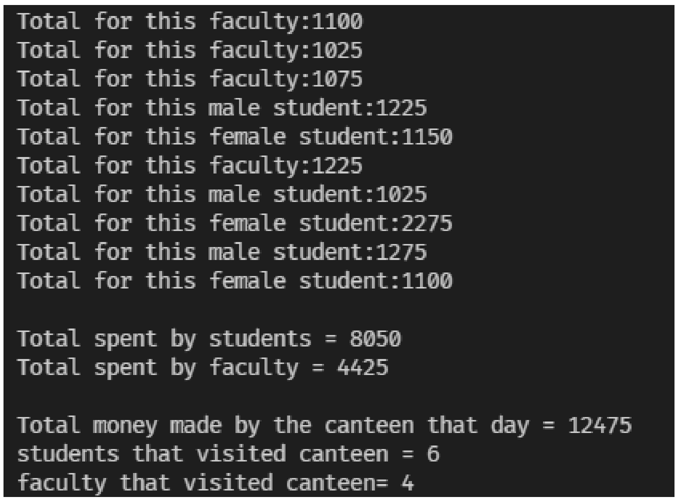
Execution:

```
package Q14;

import java.util.LinkedList;
import java.util.Queue;

public class Execution {
    public static void main(String[] args) {
        Queue<Faculty> faculty = new LinkedList<>();
        Queue<Male> males = new LinkedList<>();
        Queue<Female> females = new LinkedList<>();

        // Input process
        Faculty a = new Faculty(1, 2, 3, 4);
        Faculty b = new Faculty(2, 3, 4, 2);
        Faculty c = new Faculty(4, 3, 5, 1);
        Faculty d = new Faculty(8, 5, 3, 1);

        Male aM = new Male(2, 3, 6, 2);
        Male bM = new Male(4, 5, 3, 1);
        Male cM = new Male(4, 3, 4, 3);

        Female aF = new Female(5, 4, 3, 2);
        Female bF = new Female(4, 5, 5, 8);
        Female cF = new Female(1, 2, 3, 4);

        faculty.add(a);
        faculty.add(b);
        faculty.add(c);
        faculty.add(d);

        males.add(aM);
        males.add(bM);
        males.add(cM);

        females.add(aF);
        females.add(bF);
        females.add(cF);

        int numMale = males.size();
        int numFemale = females.size();
        int numFaculty = faculty.size();

        int studentTotal = 0, facTotal = 0;
```



```
        while (faculty.size() > 0 || males.size() > 0 ||
females.size() > 0) {
            for (int i = 0; faculty.size() > 0 && i < 3; i++) {
                Faculty x = faculty.remove();
                System.out.println("Total for this faculty:" +
x.total());
                facTotal += x.total();
            }
            for (int i = 0; males.size() > 0 && i < 1; i++) {
                Male x = males.remove();
                System.out.println("Total for this male student:" +
x.total());
                studentTotal += x.total();
            }
            for (int i = 0; females.size() > 0 && i < 1; i++) {
                Female x = females.remove();
                System.out.println("Total for this female student:" +
x.total());
                studentTotal += x.total();
            }
        }

        System.out.println("\nTotal spent by students = " +
studentTotal);
        System.out.println("Total spent by faculty = " + facTotal);

        System.out.println("\nTotal money made by the canteen that
day = " + (studentTotal + facTotal));

        System.out.println("students that visited canteen = " +
(numMale + numFemale));
        System.out.println("faculty that visited canteen= " +
numFaculty);
    }
}
```

Output:



```
Total for this faculty:1100
Total for this faculty:1025
Total for this faculty:1075
Total for this male student:1225
Total for this female student:1150
Total for this faculty:1225
Total for this male student:1025
Total for this female student:2275
Total for this male student:1275
Total for this female student:1100

Total spent by students = 8050
Total spent by faculty = 4425

Total money made by the canteen that day = 12475
students that visited canteen = 6
faculty that visited canteen= 4
```

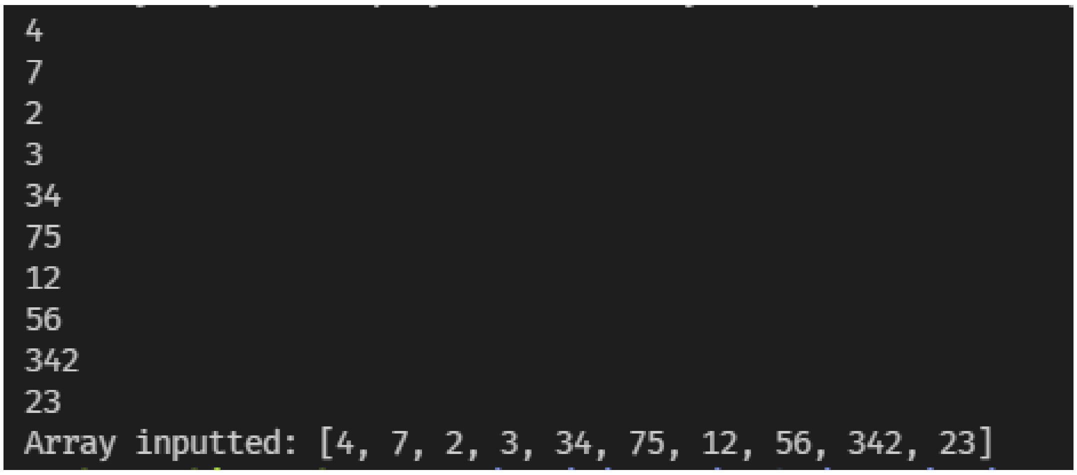
Question 15:

Use Scanner class methods to read exactly 10 numeric inputs from the user. Calculate the individual count of float and integer values out of those 10 inputs, without using any hasNext() methods. Use exceptions instead.

Code:

```
import java.util.Arrays;
import java.util.NoSuchElementException;
import java.util.Scanner;

public class Q15 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        final int LIMIT = 10;
        int[] arr = new int[10];

        int i = 0;

        while (i < LIMIT) {
            try {
                arr[i] = sc.nextInt();
                i++;
            } catch (NoSuchElementException e) {
                e.printStackTrace();
            }
        }

        System.out.println("Array inputted: " +
Arrays.toString(arr));

        sc.close();
    }
}
```

Output:



```
4
7
2
3
34
75
12
56
342
23
Array inputted: [4, 7, 2, 3, 34, 75, 12, 56, 342, 23]
```

Question 16:

Define a class Prime with a method to determine if the given number is a prime number or not. If not, throw NotPrimeException. Derive TwinPrime class from Prime with a method that takes two numbers as arguments. It should throw NotPrimeException, even if any one of the two numbers is not prime and should throw NotTwinException if the difference between the numbers is not two. Instantiate the TwinPrime class from the main method to process ‘n’ such pairs.

Code:

NotPrimeException:

```java
package Q16;

public class NotPrimeException extends Exception {
    public NotPrimeException(String msg) {
        super(msg);
    }
}
```

NotTwinException:

```java
package Q16;

public class NotTwinException extends Exception {
    public NotTwinException(String msg) {
        super(msg);
    }
}
```

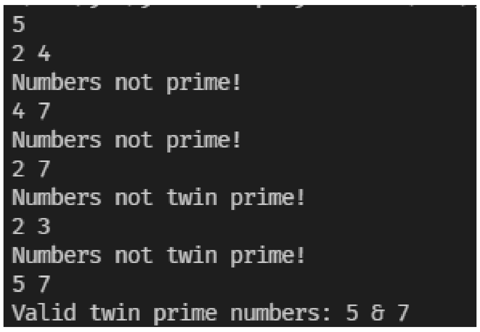
Prime:

```java
package Q16;

public class Prime {
    public static void checkPrime(int n) throws NotPrimeException {
        for (int i = 2; i <= n / 2; i++) {
            if ((n % i) == 0) {
                throw new NotPrimeException("" + n + " is not a prime
number");
            }
        }
    }
}
```

TwinPrime:

```
package Q16;

import java.util.Scanner;

public class TwinPrime extends Prime {
    public static void checkTwinPrime(int x, int y) throws
NotTwinException, NotPrimeException {
        try {
            checkPrime(x);
        } catch (NotPrimeException e) {
            throw e;
        }
        try {
            checkPrime(y);
        } catch (NotPrimeException e) {
            throw e;
        }
        if (x - y != -2 && x - y != 2) {
            throw new NotTwinException("Difference not = 2");
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        for (int i = 0; i < n; i++) {
            int x = sc.nextInt();
            int y = sc.nextInt();

            try {
                checkTwinPrime(x, y);
                System.out.println("Valid twin prime numbers: " + x +
" & " + y);
            } catch (NotPrimeException e) {
                System.out.println("Numbers not prime!");
            } catch (NotTwinException e) {
                System.out.println("Numbers not twin prime!");
            }
        }
        sc.close();
    }
}
```

Output:



```
5
2 4
Numbers not prime!
4 7
Numbers not prime!
2 7
Numbers not twin prime!
2 3
Numbers not twin prime!
5 7
Valid twin prime numbers: 5 & 7
```

Question 17:

A text file contains several lines of sentences as follows. Each line contains a sequence of words and their corresponding parts-of-speech within a pair of parentheses. Here, AUX, N,V,ADJ,DET refer to auxiliary verb, noun, verb, adjective and determiner respectively. Read them and create a TreeMap to store each parts-of-speech as the key and the list of words which have been tagged with that parts-of-speech as the value. Eg., N={Brown, coffee, Swift, car, drive}

Did Brown drink coffee? (AUX N V N) Coffee is brown (N AUX ADJ) Swift is a car (N AUX DET N) Car is brown (N AUX ADJ) Brown had a swift drive (N AUX DET ADJ N) Did Brown drive? (AUX N V) Read an input phrase as the one given below and store all possible parts-of-speech for each word in a 2-D ragged array. Read the user-given tag sequence for the phrase and check if it is an acceptable tagging or not, based on the 2-D ragged array. Print the map, array and the acceptance status. “a brown swift”

DET

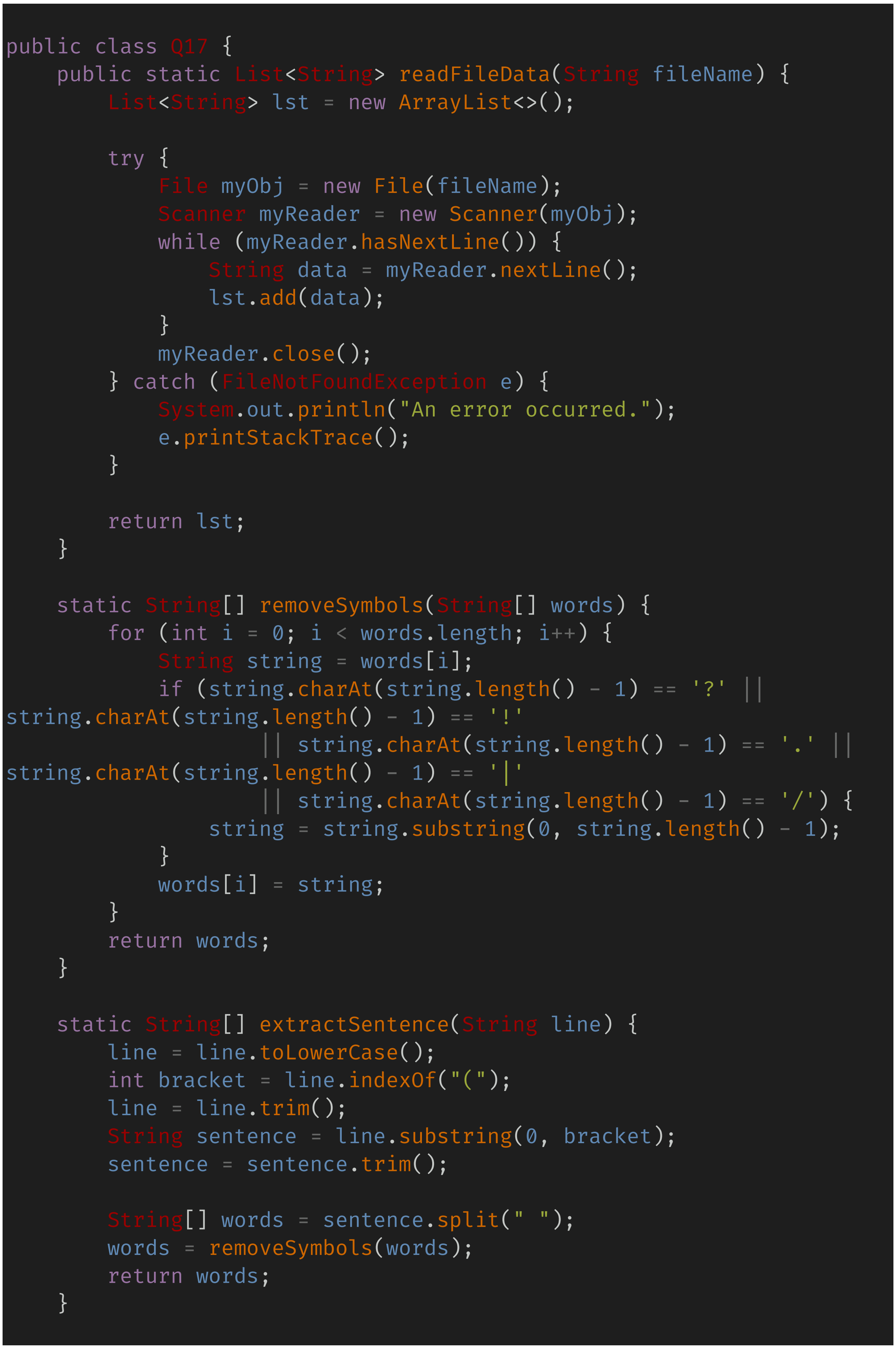
N ADJ

N ADJ

Eg., if the given tag sequence is “DET N V”, it is not an acceptable tagging. If the given tag sequence is “DET ADJ ADJ”, it is acceptable.

Code:

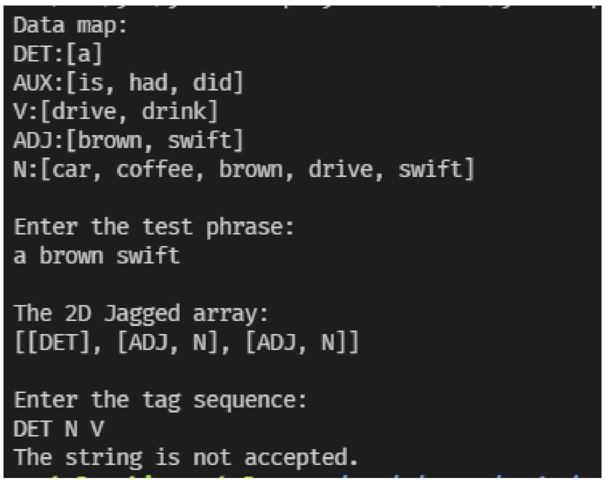
```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.Set;
```



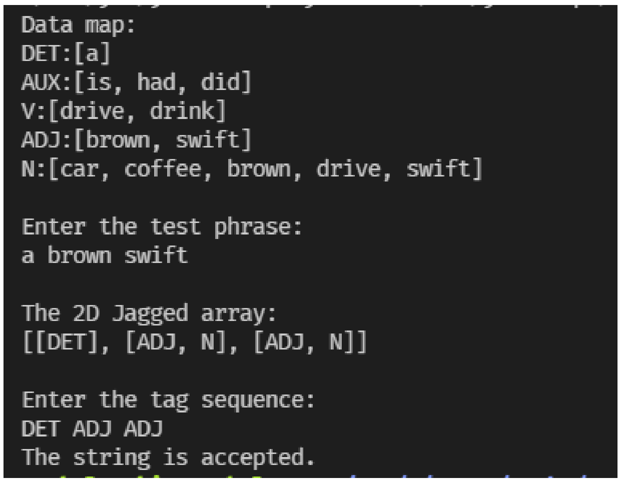
```java
public class Q17 {
    public static List<String> readFileData(String fileName) {
        List<String> lst = new ArrayList<>();

        try {
            File myObj = new File(fileName);
            Scanner myReader = new Scanner(myObj);
            while (myReader.hasNextLine()) {
                String data = myReader.nextLine();
                lst.add(data);
            }
            myReader.close();
        } catch (FileNotFoundException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }

        return lst;
    }

    static String[] removeSymbols(String[] words) {
        for (int i = 0; i < words.length; i++) {
            String string = words[i];
            if (string.charAt(string.length() - 1) == '?' ||
string.charAt(string.length() - 1) == '!'
                    || string.charAt(string.length() - 1) == '.' ||
string.charAt(string.length() - 1) == '|'
                    || string.charAt(string.length() - 1) == '/') {
                string = string.substring(0, string.length() - 1);
            }
            words[i] = string;
        }
        return words;
    }

    static String[] extractSentence(String line) {
        line = line.toLowerCase();
        int bracket = line.indexOf("(");
        line = line.trim();
        String sentence = line.substring(0, bracket);
        sentence = sentence.trim();

        String[] words = sentence.split(" ");
        words = removeSymbols(words);
        return words;
    }
    static String[] extractTerms(String line) {
        line = line.toUpperCase();
        int bracket = line.indexOf("(");
        line = line.trim();
        String sentence = line.substring(bracket + 1, line.length() - 1);
        sentence = sentence.trim();

        String[] words = sentence.split(" ");
        words = removeSymbols(words);
        return words;
    }

    static Map<String, Set<String>> mapAdder(String[] words, String[] terms, Map<String, Set<String>> dataMap) {

        for (int i = 0; i < terms.length; i++) {
            for (int j = 0; j < words.length; j++) {
                if (dataMap.containsKey(terms[i])) {
                    Set<String> s = dataMap.get(terms[i]);
                    s.add(words[i]);
                    dataMap.put(terms[i], s);
                } else {
                    Set<String> s = new HashSet<>();
                    s.add(words[i]);
                    dataMap.put(terms[i], s);
                }
            }
        }
        return dataMap;
    }

    public static void main(String[] args) {
        List<String> fileData = readFileData("./q17text.txt");

        Map<String, Set<String>> dataMap = new HashMap<>();
        for (String line : fileData) {

            String[] words = removeSymbols(extractSentence(line));
            String[] terms = removeSymbols(extractTerms(line));

            dataMap = mapAdder(words, terms, dataMap);
        }

        System.out.println("Data map:");
        dataMap.forEach((k, v) -> System.out.println("" + k + ":" + v));
```

```java
        System.out.println("");
        System.out.println("Enter the test phrase:");
        Scanner sc = new Scanner(System.in);
        String phrase = sc.nextLine().trim();
        List<Set<String>> phrases = new ArrayList<>();
        String[] phraseWords = phrase.split(" ");

        for (String phrWord : phraseWords) {
            Set<String> str = new HashSet<>();
            for (Map.Entry<String, Set<String>> entry :
dataMap.entrySet()) {
                if (entry.getValue().contains(phrWord)) {
                    str.add(entry.getKey());
                }
            }
            phrases.add(str);
        }

        System.out.println("");
        System.out.println("The 2D Jagged array:");
        System.out.println(phrases);

        System.out.println("");
        System.out.println("Enter the tag sequence:");
        String tagSequnce = sc.nextLine().trim();
        String[] tagSeqWords = tagSequnce.split(" ");
        sc.close();

        boolean accepted = true;

        int i = 0;
        for (String tag : tagSeqWords) {
            if (phrases.get(i).contains(tag) == false) {
                accepted = false;
                break;
            }
            i++;
        }

        if (accepted) {
            System.out.println("The string is accepted.");
        } else {
            System.out.println("The string is not accepted.");
        }
    }
}
```

Output:



```
Data map:
DET:[a]
AUX:[is, had, did]
V:[drive, drink]
ADJ:[brown, swift]
N:[car, coffee, brown, drive, swift]

Enter the test phrase:
a brown swift

The 2D Jagged array:
[[DET], [ADJ, N], [ADJ, N]]

Enter the tag sequence:
DET N V
The string is not accepted.
```



```
Data map:
DET:[a]
AUX:[is, had, did]
V:[drive, drink]
ADJ:[brown, swift]
N:[car, coffee, brown, drive, swift]

Enter the test phrase:
a brown swift

The 2D Jagged array:
[[DET], [ADJ, N], [ADJ, N]]

Enter the tag sequence:
DET ADJ ADJ
The string is accepted.
```

****