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](https://github.com/sankalpmukim/java-lab-fall-sem-2021)

# Question 10:

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



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

Table

Description automatically generated

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.

1. 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
2. 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:

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:

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:

Text, chat or text message

Description automatically generated

# 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;

    public Node left;

    public Node right;

    public Node(int n) {

        value = n;

        left = null;

        right = null;

    }

}

#### LinkedListNode:

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:

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:

Text

Description automatically generated

# 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());

    }

}

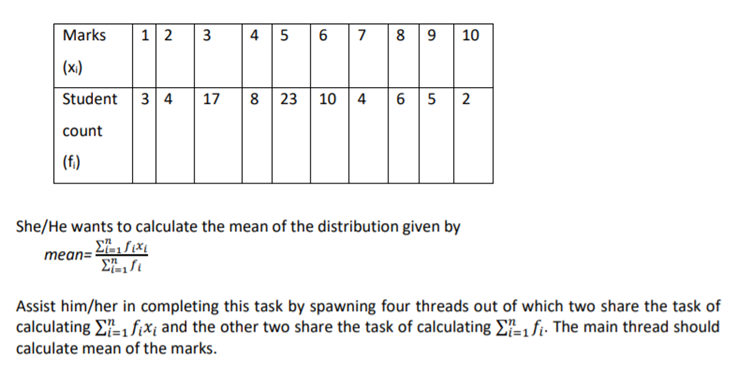
## Output:

Text

Description automatically generated

# Question 13:

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



## Code:

### Numerator:

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:

package Q13;

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:

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:

Text

Description automatically generated with medium confidence

# 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;

    }

    public int total() {

        return (numDish1 \* 50) + (numDish2 \* 75) + (numDish3 \* 100) + (numDish4 \* 150);

    }

}

### Female:

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:

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: