Final Test Codes

Employee Implementation

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
import java.util.stream.*;
interface Company {
void assignSalaries(int[] salaries);
void averageSalary();
void maxSalary();
void minSalary();
}
/*
* model output for cut and paste
* Incomes of __ credited
* Average salary of __ is __
* Maximum salary amongst __ is __
* Minimum salary amongst __ is __
*/
class AccountantFirm extends EngineerFirm {
public AccountantFirm(int n) {
super(n);
}
public void assignSalaries(int[] salaries) {
super.assignIncome(salaries);
printMessages(0, "", "accountants");
}
public void maxSalary() {
printMessages(super.MaxSalary(), "max", "accountants");
}
public void minSalary() {
printMessages(super.MinSalary(), "min", "accountants");
}
public void averageSalary() {
printMessages(super.AveSalary(), "ave", "accountants");
}
}
class EngineerFirm {
private final int[] income;
public EngineerFirm(int n) {
income = new int[n];
for (int i = 0; i < n; i++) {
income[i] = 0;
}
}
public static void printMessages(double salaryAmount, String salarySpecification, String profession) {
switch (salarySpecification) {
case "max":
System.out.print("Maximum salary amongst " + profession);
```

```
System.out.printf(" is %d", (int) salaryAmount);
System.out.println("");
break;
case "min":
System.out.print("Minimum salary amongst " + profession);
System.out.printf(" is %d", (int) salaryAmount);
System.out.println("");
break;
case "ave":
System.out.print("Average salary of " + profession);
System.out.printf(" is %.2f", salaryAmount);
System.out.println("");
break;
default:
System.out.println("Incomes of " + profession + " credited");
break;
}
}
public void assignSalaries(int[] salaries) {
if (salaries != null) {
assignIncome(salaries);
printMessages(0, "", "engineers");
}
}
public void maxSalary() {
printMessages(MaxSalary(), "max", "engineers");
}
public void minSalary() {
printMessages(MinSalary(), "min", "engineers");
}
public void averageSalary() {
printMessages(AveSalary(), "ave", "engineers");
}
public Integer MaxSalary() {
List<Integer> list = Arrays.stream(income).boxed().collect(Collectors.toList());
return list.stream().max(Integer::compare).get();
}
public Integer MinSalary() {
List<Integer> list = Arrays.stream(income).boxed().collect(Collectors.toList());
return list.stream().min(Integer::compare).get();
}
public double AveSalary() {
List<Integer> list = Arrays.stream(income).boxed().collect(Collectors.toList());
IntSummaryStatistics stats = list.stream().mapToInt((x) -> x).summaryStatistics();
return stats.getAverage();
}
public void assignIncome(int[] salaries) {
System.arraycopy(salaries, 0, income, 0, Math.min(income.length, salaries.length));
}
}
public class Solution {
public static void main(String args[]) throws Exception {
Scanner sc = new Scanner(System.in);
String[] count = sc.nextLine().split(" ");
```

```
EngineerFirm e = new EngineerFirm(Integer.parseInt(count[0]));
AccountantFirm a = new AccountantFirm(Integer.parseInt(count[1]));
count = sc.nextLine().split(" ");
int[] incomeEngineers = new int[count.length];
for (int i = 0; i < count.length; i++) {
incomeEngineers[i] = Integer.parseInt(count[i]);
}
e.assignSalaries(incomeEngineers);
count = sc.nextLine().split(" ");
int[] incomeAccountants = new int[count.length];
for (int i = 0; i < count.length; i++) {
incomeAccountants[i] = Integer.parseInt(count[i]);
}
a.assignSalaries(incomeAccountants);
e.averageSalary();
e.maxSalary();
e.minSalary();
a.averageSalary();
a.maxSalary();
a.minSalary();
}
}
```

Student Enrollement

```
import java.util.*;
class Student
{
   String name;
   int studentClass;
   float result;
   Student (String s, int sClass)
   {
      name = s;
      studentClass = sClass;
      System.out.println ("Added student: " + s + " to the roll of class: " +sClass);
   }
   String getName ()
   {
      return name;
   }
   String Publish ()
   {
      if (result >= 33.33)
      {
          return (name + " has been promoted to class: " + (studentClass + 1));
      }
      else
      {
          return (name + " has been retained in class: " + studentClass);
      }
   }
}
class Result extends Student
{
```

```
   int subject1, subject2, subject3;
      Result (int a, int b, int c, String s, int sClass)
   {
      super (s, sClass);
      subject1 = a;
      subject2 = b;
      subject3 = c;
      System.out.println (s + " obtained " + a + " marks in subject1");
      System.out.println (s + " obtained " + b + " marks in subject2");
      System.out.println (s + " obtained " + c + " marks in subject3");
   }
   String calculateResult ()
   {
      super.result = ((subject1 + subject2 + subject3) * (100) / 300);
      String str = super.Publish ();
      return str;
   }
   String changeMarks (int newMarks, String subject)
   {
      System.out.println (super.name + " has ordered recheck in " + subject);
      switch(subject) {
         case "subject1":
                     subject1 = newMarks;
                     break;
         case "subject2":
                    subject2 = newMarks;
                     break;
             case "subject3":
                  subject3 = newMarks;
                  break;
         }
         return("Following is the new result: "+ calculateResult());
   }
}
public class Main{
      public static void main(String[] args){
            Scanner sc=new Scanner(System.in);
            String[] names=sc.nextLine().split(" ");
            int[][] marks= new int[names.length][3];
            for(int i=0;i<names.length;i++){
                  String[] temp=sc.nextLine().split(" ");
                  marks[i][0]=Integer.parseInt(temp[0]);
                  marks[i][1]=Integer.parseInt(temp[1]);
                  marks[i][2]=Integer.parseInt(temp[2]);
            }
            String[] cla =sc.nextLine().split(" ");
            Result r1=new Result(marks[0][0],marks[0][1],marks[0][2],names[0], Integer.parseInt(cla[0]));
            Result r2=new Result(marks[1][0],marks[1][1],marks[1][2],names[1], Integer.parseInt(cla[1]));
            Result r3=new Result(marks[2][0],marks[2][1],marks[2][2],names[2], Integer.parseInt(cla[2]));
            Result r4=new Result(marks[3][0],marks[3][1],marks[3][2],names[3], Integer.parseInt(cla[3]));
            Result r5=new Result(marks[4][0],marks[4][1],marks[4][2],names[4], Integer.parseInt(cla[4]));
            String sub=sc.nextLine();
            int newMarks=Integer.parseInt(sc.nextLine());
            System.out.println(r1.calculateResult());
```

```
        System.out.println(r2.calculateResult());
        System.out.println(r3.calculateResult());
        System.out.println(r4.calculateResult());
        System.out.println(r5.calculateResult());

        System.out.println(r1.changeMarks(newMarks,sub));
        System.out.println(r3.changeMarks(newMarks,sub));
        System.out.println(r5.changeMarks(newMarks,sub));
        }
}
```

Sport inheritance

```
import java.util.*;
interface Sport {
        void calculateAvgAge(int []age);
        void retiredPlayer(int id);
}
class Cricket implements Sport {
        int []playerId=new int[11];
        Cricket() {
                Arrays.fill(playerId, 1);
                System.out.println("A new cricket team has been formed");
        }
        public void calculateAvgAge(int []age)
        {
                double sum=0,length=age.length;
                for(int i=0;i<length;i++) {
                        sum+=age[i];
                }
                double avg=sum/length;
                System.out.println("The average age of the team is"+String.format("%.2f,avg"));
        }
        public void retiredPlayer(int id) {
                        if(playerId[id-1]!=-1) {
                                playerId[id-1] = -1;
                                System.out.println("Player with id:"+id+"has retired");
                        }
                        else
                                System.out.println("Player has already retired");
        }
}
class Football implements Sport {
        int []playerId=new int [11];
        Football() {
                Arrays.fill(playerId,1);
                System.out.println("A new football team has been formed");
        }
        public void calculateAvgAge(int []age)
        {
                double sum=0,length=age.length;
                for(int i=0;i<length;i++) {
                        sum+=age[i];
```

```
                }
                double avg=sum/length;
                System.out.print("The average age of the team is"+String.format("%2f",avg));
        }
        public void retiredPlayer(int id) {
                if(playerId[id-1]!=-1) {
                        playerId[id-1]=-1;
                        System.out.print
                        ("Player with id:"+id+"has retired");
                }
                else
                        System.out.println("Player has already retired");
        }
        public void playerTransfer(int fee,int id) {
                if(playerId[id-1]!=-1) {
                        playerId[id-1]=-1;
                        System.out.println("Player with id:"+id+"has been transferred with a fee of"+fee);
                }
                else
                        System.out.println("Player has already retired");
        }
}
public class Main {
        public static void main(String[] args) {
                Scanner sc=new Scanner(System.in);
        }
}
```

Building Implementation

```
public interface building{
    public int[] floors;
}
public class school implements building{

      school(int n)
      {
            floors=new int[n];
            for(int i=0;i<n;i++)
            {
                  floors[i]=0;
            }
            System.out.println("A school is being constructed");
      }
      void floorCompleted(int floorNum){
            if(floorNum<=n){
                  floors[floorNum-1]=1;
                  System.out.println("Construction for floor number "+floorNum+" completed in school");
            }
            else{
                  System.out.println("Floor number "+floorNum+" does not exist in school");
            }
      }
```

```
	void printStatus(int floorNum){
		if(floorNum>floors.length){
			System.out.println("Floor number "+floorNum+" does not exist in school");
		}
		else if(floors[floorNum-1]==1){
			System.out.println("Construction for floor number "+floorNum+" completed in school");
		}
		else{
			if(floors[floorNum-1]==0){
				System.out.println("Construction for floor number "+floorNum+" not completed in school");
			}
		}
	}
	void printNumberOfFloors(){
		System.out.println(floors.length);
	}

}
public class hospital implements building{
	hospital(int n)
	{
		floors=new int[n];
		for(int i=0;i<n;i++)
		{
			floors[i]=0;
		}
		System.out.println("A hospital is being constructed");
	}
	void floorCompleted(int floorNum){
		if(floorNum<=n){
			floors[floorNum-1]=1;
			System.out.println("Construction for floor number "+floorNum+" completed in hospital");
		}
		else{
			System.out.println("Floor number "+floorNum+" does not exist in hospital");
		}
	}
	void printStatus(int floorNum){
		if(floorNum>floors.length){
			System.out.println("Floor number "+floorNum+" does not exist in hospital");
		}
		else if(floors[floorNum-1]==1){
			System.out.println("Construction for floor number "+floorNum+" completed in hospital");
		}
		else{
			if(floors[floorNum-1]==0){
				System.out.println("Construction for floor number "+floorNum+" not completed in hospital");
			}
		}
	}
	void printNumberOfFloors(){
		System.out.println(floors.length);
	}
}
```

Intermediate test Codes

Animal code

```
interface Animal{
  void eat();
  void makeSound();
}
interface Bird{
  static int legs = 2;
  void fly();
}
class Parrot implements Bird, Animal{
  public void eat(){
  System.out.println("Parrots can eat up to 100 gms in a day");
  }
  public void makeSound(){
  System.out.println("Parrots make sound of screech");
  }
  public void fly(){
  System.out.println("Parrots can fly up to 50 miles in a day");
  }
}
```

Nutrition code

```
import java.util.*;
abstract class Food{
  double proteins;
  double fats;
  double carbs;
  double tastyScore;
  void getMacroNutrients(){}
}
class Bread extends Food{
  String type;
  public Bread(double proteins,double fats,double carbs) {
  this.proteins=proteins;
  this.fats=fats;
  this.carbs=carbs;
  this.tastyScore=8;
  this.type = "vegeterian";
  }
  void getMacroNutrients()
  {
  System.out.println("A slice of bread has "+String.valueOf(this.proteins)+" gms of protein,
"+String.valueOf(this.fats)+
  " gms of fats and "+String.valueOf(this.carbs)+" gms of carbohydrates.");
  }
}
class Egg extends Food{
  String type;
  public Egg(double proteins,double fats,double carbs) {
  this.proteins=proteins;
  this.fats=fats;
```

```
  this.carbs=carbs;
  this.tastyScore=8;
  this.type = "non-vegeterian";
  }
  void getMacroNutrients()
  {
  System.out.println("An egg has "+String.valueOf(this.proteins)+" gms of protein,
"+String.valueOf(this.fats)+
  " gms of fats and "+String.valueOf(this.carbs)+" gms of carbohydrates.");
  }
}
public class Solution {
  public static void main(String args[]) {
  Scanner sc = new Scanner(System.in);
  int cnt = Integer.parseInt(sc.nextLine());
  for(int i=0;i< cnt;i++)
  {
  String name = sc.nextLine();
  if(name.equals("Bread")) {
  Bread breadobj = new Bread(4, 1.1, 13.8);
  for (int j = 0; j < 3; j++) {
  String command = sc.nextLine();
  if (command.equals("getMacros"))
  breadobj.getMacroNutrients();
  else if (command.equals("getTaste"))
  System.out.println("Taste: " + breadobj.tastyScore);
  else if (command.equals("getType"))
  System.out.println("Bread is " + breadobj.type);
  }
  }
  if(name.equals("Egg")) {
  Egg eggobj = new Egg(6.3, 5.3, 0.6);
  for (int j = 0; j < 3; j++) {
  String command = sc.nextLine();
  if (command.equals("getMacros"))
  eggobj.getMacroNutrients();
  else if (command.equals("getTaste"))
  System.out.println("Taste: " + eggobj.tastyScore);
  else if (command.equals("getType"))
  System.out.println("Bread is " + eggobj.type);
  }
  }
  }
  }
}
```

Car fueling

```
class Car
{
  public void topSpeed()
  {
  System.out.println("Top speed of the vehicle is 100 kmph");
  }
```

```
public void fuelType()
  {
  System.out.println("Car fuel type is Petrol");
  }
  }
class SUV extends Car{
  public void fuelType()
  {
  System.out.println("SUV fuel type is Diesel");
  }
}
```

Serial multiplier CODE

```
import java.util.*;
public class SerialMultiplier
{
  static int first=1, second=1, third=1, fourth=1, fifth=1;
  static int result=0;
  public SerialMultiplier(int first)
  {
  this.first=first;
  result=first;
  }
  public SerialMultiplier(int first, int second)
  {
  this.first=first;
  this.second=second;
  result=first*second;
  }
  public SerialMultiplier(int first, int second, int third)
  {
  this.first=first;
  this.second=second;
  this.third=third;
  result=first*second*third;
  }
  public SerialMultiplier(int first, int second, int third, int fourth)
  {
  this.first=first;
  this.second=second;
  this.third=third;
  this.fourth=fourth;
  result=first*second*third*fourth;
  }
  public SerialMultiplier(int first, int second, int third, int fourth, int fifth)
  {
  this.first=first;
  this.second=second;
  this.third=third;
  this.fourth=fourth;
  this.fifth=fifth;
  result=first*second*third*fourth*fifth;
```

```
  }

  public static void main( String args[])
  {
  Scanner sc= new Scanner(System.in);
  int n=sc.nextInt();
  int a[]=new int[n];
  for(int i=0;i<n;i++)
  {
  a[i]=sc.nextInt();
  }

  if(n==1)
  {
  SerialMultiplier obj=new SerialMultiplier(a[0]);
  }
  if(n==2)
  {
  SerialMultiplier obj=new SerialMultiplier(a[0],a[1]);
  }
  if(n==3)
  {
  SerialMultiplier obj=new SerialMultiplier(a[0],a[1],a[2]);
  }
  if(n==4)
  {
  SerialMultiplier obj=new SerialMultiplier(a[0],a[1],a[2],a[3]);
  }
  if(n==5)
  {
  SerialMultiplier obj=new SerialMultiplier(a[0],a[1],a[2],a[3],a[4]);
  }
  System.out.println(result);
  }
}
```

Count Binary Substrings CODE

```
class Solution {
  public int countBinarySubstrings(String s) {
  int count=0;
  for(int i=0;i<s.length()-1;i++){

  count+=doCount(s,i,i+1,0);

  }
  return count;
  }
  private int doCount(String s,int start,int end,int count){
  if(s.charAt(start)=='0'&&s.charAt(end)=='1'){
  while(start>=0&&end<s.length()&&s.charAt(start)=='0'&&s.charAt(end)=='1'){
  count++;
  start--;
  end++;
```

```
  }
  }else if(s.charAt(start)=='1'&&s.charAt(end)=='0'){
  while(start>=0&&end<s.length()&&s.charAt(start)=='1'&&s.charAt(end)=='0'){
  count++;
  start--;
  end++;
  }
  }
  return count;
  }
}
```

Student class

```
class Student
{
  private String myName;
  private static int myRegNum = 0;
  Student(String name)
  {
  myName = name;
  myRegNum += 1;
  }

  @Override
  public String toString() {
  return this.myRegNum + ": " + this.myName;
  }
}
```

Library structure code

```
class Library{
  private int number_of_books;
  private String name;
  private Map<String,Integer> bookGeneres = new HashMap<>();
  public int getNumber_of_books() {
  return number_of_books;
  }
  public void setNumber_of_books(int number_of_books) {
  this.number_of_books = number_of_books;
  }
  public String getName() {
  return name;
  }
  public void setName(String name) {
  this.name = name;
  }
  public Map<String, Integer> getBookGeneres() {
  return bookGeneres;
  }
  public void setBookGeneres(Map<String, Integer> bookGeneres) {
  this.bookGeneres = bookGeneres;
```

```
  }
}
```

Braces code

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {

public static boolean isBalanced(String s) {
     int len=s.length();
       if(len==0 || s==null) return true;
          Stack<Character> stack = new Stack<Character>();
       for(int i=0;i<s.length();i++)
       {
             if(s.charAt(i)=='(' || s.charAt(i)=='[' || s.charAt(i)=='{')     stack.push(s.charAt(i));
             else if(s.charAt(i)==')' && !stack.empty() && stack.peek()=='(') stack.pop();
             else if(s.charAt(i)==']' && !stack.empty() && stack.peek()=='[') stack.pop();

             else if(s.charAt(i)=='}' && !stack.empty() && stack.peek()=='{') stack.pop();
             else return false;


       }
       return stack.empty();
       }

       public static void main(String[] args) {
       Scanner in = new Scanner(System.in);
       int t = in.nextInt();
       for (int a0 = 0; a0 < t; a0++) {
             String expression = in.next();
             System.out.println( (isBalanced(expression)) ? "YES" : "NO" );
       }
}
}
```

Employee profile

```
public abstract class AbstractEmployee {

       public abstract void setSalary(int salary);

       public abstract int    getSalary();

       public abstract void setGrade(String grade);

       public abstract String    getGrade();

         void label(int salary, String grade)
```

```
	{
		System.out.println("Employee's data :"+ "Salary:"+salary +"Grade:" +grade);
	}
}
public class Engineer extends AbstractEmployee {
	private int salary;
	private String grade;
	public void setSalary(int salary) {
		salary = this.salary;
	}

	public int    getSalary() {
		return    salary;
	}

	public void setGrade(String grade) {
	grade = this.grade;
	}

	public String getGrade() {
	return grade;
	}

	public static void main(String args[])
	{
		Engineer e1 = new Engineer();
		e1.label(10000,"Grade-A");

	}
}
public class Manager extends AbstractEmployee {

	private int salary;
	private String grade;

	public void setSalary(int salary) {
		salary = this.salary;
	}

	public int getSalary() {
		return salary;
	}

	public void setGrade(String grade) {
		grade = this.grade;
	}

	public String getGrade() {
		return grade;
	}

	public static void main(String args[]) {
		Manager m1 = new Manager();
		m1.label(20000,"Grade-B");
```

```
    }
}
```

Car Engine

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
class Car
{
  public void printTopSpeed()
  {
  System.out.println("Top speed of the vehicle is 100 kmph");
  }
  public void printTopSpeed(int topSpeed)
  {
  System.out.println("Top speed of the vehicle is " +topSpeed+ " kmph ");
  }
  public void printTopSpeed(String vehicleName,int topSpeed)
  {
  System.out.println("Top speed of the vehicle " +vehicleName+ " is " +topSpeed+ " kmph ");
  }
}
```

Addition Magic

```
public class AdditionMagic{
    public String Add(double a, String b) {
        String one01=String.valueOf(a);
        String second22=one01.concat(b);
        return second22;
    }
    public String add(double a, double b){
        double c01=a+b;
        double d01=Math.round(c01*100.00)/100.00;
        String one1=String.valueOf(d01);
        return one1;
    }
    public String Add(String a, String b) {
        String third11=a.concat(b);
        return third11;
    }
}
```