1 ArrayList

1. Во спортот скокови во вода, седум судии го оценуваат скокот со оценка од 0 до 10, со што оценката може да биде децимален број. Најдобрата и најлошата оценка се исфрлаат, а останатите се собираат. Сумата потоа се множи со степенот на тежина на тој скок. Степенот на тежина е број од 1.2 до 3.8 поени. Вкупниот збир потоа се множи со 0.6 и се добива крајниот резултат на скокачот. Да се напише компјутерска програма која за дадени степен на тежина и оценките од седумте судии, ќе го пресмета резултатот од скокот. Програмата треба да користи `ArrayList` со тип `Double` за оценките од судиите.

```java
package edu.finki.np.av4;

import java.util.ArrayList;
import java.util.Scanner;

public class Dive {
    private static final double COEFICIENT = 0.6;
    private ArrayList<Double> scores;
    private double difficulty;

    public Dive(double difficulty) {
        this.difficulty = difficulty;
        this.scores = new ArrayList<Double>();
    }

    public void addScore(double score) {
        scores.add(score);
    }

    public double getFinalScore() {
        double min = Double.MAX_VALUE;
        double max = Double.MIN_VALUE;
        int maxIndex = -1;
        int minIndex = -1;
        for (int i = 0; i < scores.size(); i++) {
            double d = scores.get(i);
            if (d < min) {
                min = d;
                minIndex = i;
            }
            if (d > max) {
                max = d;
                maxIndex = i;
            }
        }
        double sum = 0;
        for (int i = 0; i < scores.size(); i++) {
            double d = scores.get(i);
            if (i != minIndex && i != maxIndex) {
                sum += d;
            }
        }
        return sum * difficulty * COEFICIENT;
    }

    public static void main(String[] args) {
```

```
        double difficulty;
        System.out.println("Difficulty: ");
        Scanner scanner = new Scanner(System.in);
        difficulty = scanner.nextDouble();
        Dive dive = new Dive(difficulty);
        for (int i = 0; i < 7; i++) {
            System.out.println(String.format("Judge %d: ", i +
                1));
            double score = scanner.nextDouble();
            dive.addScore(score);
        }
        System.out.println(String.format("Result: %.2f", dive.
            getFinalScore()));
    }
}
```

2. Со помош на Глобалниот Систем за Позиционирање (GPS) може да добиваме патеки на движење. Патека на движење се состои од кооридните на локации на мапа заедно со соодветни времиња (timestamp). Нашиот GPS систем запишува точки на движење кои се состојат од (X, Y) координати на мапа заедно со времиња (timestamp t) во кои се запишани бројот на секунди кои поминале откако е вклучен уредот. Да се напише програма во која од многу точки кои се читаат и се сместуваат во `ArrayList`, со помош на класа за репрезетнација на точка. Секоја точка е последователна локација од движењето по некоја планинарска рута. Координатите се од тип `double`, а времето од тип `integer`. Вашата програма треба да го пресметува вкупното поминато растојание и просечната брзина во километри на час. Мапата е скалирана со фактор 1 = 0.1 километар. На пример, ако две точки се на координати (X = 1, Y = 1, T = 0) и (X = 2, Y = 1, T = 3600), тогаш вкупното поминато растојание е 0.1 километар за 3600 секунди, односно со брзина 0.1 km/h.

```
package edu.finki.np.av4;

public class Waypoint {
    private double x;
    private double y;
    private int timestamp;

    public Waypoint(double x, double y, int timestamp) {
        this.x = x;
        this.y = y;
        this.timestamp = timestamp;
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

    public int getTimestamp() {
        return timestamp;
    }
```

```
    public int getTime(Waypoint point) {
        return timestamp - point.timestamp;
    }

    public double getDistance(Waypoint point) {
        return Math.sqrt((x - point.x) * (x - point.x) + (y -
            point.y)
                * (y - point.y));
    }

}
```

```
package edu.finki.np.av4;

import java.util.ArrayList;

public class HikerTrack {
    private static final double SCALE_FACTOR = 0.1;
    private ArrayList<Waypoint> waypoints;
    private double distance;
    private int time;

    public HikerTrack() {
        waypoints = new ArrayList<Waypoint>();
        distance = 0;
        time = 0;
    }

    public void addPoint(double x, double y, int timestamp) {
        Waypoint w = new Waypoint(x, y, timestamp);
        waypoints.add(w);
        int size = waypoints.size();
        if(size >= 2) {
            distance += waypoints.get(size - 2).getDistance(
                waypoints.get(size - 1));
        }
    }

    public double getDistance() {
        /*double d = 0;
        for (int i = 1; i < waypoints.size(); i++) {
            d += waypoints.get(i).getDistance(waypoints.get(i -
                1));
        }
*/      return distance * SCALE_FACTOR;
    }

    public int getTime() {
        return waypoints.get(waypoints.size() - 1).getTimestamp
            ()
                - waypoints.get(0).getTimestamp();
    }

    public double getSpeed() {
        double distance = getDistance();
        int time = getTime();
        return distance * 3600 / time;
```

```
    }

    public static void main(String[] args) {
        HikerTrack track = new HikerTrack();
        double[] x = new double[] { 0.1, 3.4, 4, 5.3, 8.9, 12.4,
             13 };
        double[] y = new double[] { 1, 2.4, 5, 6.3, 8.2, 11.4,
            12 };
        int[] times = new int[] { 0, 500, 1000, 1500, 2000,
            2500, 3000 };
        for (int i = 0; i < x.length; i++) {
            track.addPoint(x[i], y[i], times[i]);
        }
        System.out.println(String.format("Distance: %.2f km",
                track.getDistance()));
        System.out.println(String.format("Time: %d seconds",
            track.getTime()));
        System.out.println(String.format("Speed: %.2f km/h",
            track.getSpeed()));
    }
}
```

2 Генеричко програмирање

3. Да се напише генеричка класа кој симулира исцртување на случаен предмет од некоја кутија. Оваа класа треба да се користи за случајно исцртување. На пример, класата може да содржи листа со имиња и избира едно случајно име, или пак листа со броеви за лотарија и избира случајно број. Креирајте метод `add` за додавање објект од соодветниот тип и метод `isEmpty` кој проверува дали кутијата е празна. На крај, имплементирајте метод `drawItem` кој случајно избира објект од кутијата и го враќа назад. Ако се обидеме да цртаме со празна кутија се враќа `null`. Да се напише `main` метод кој ја тестира класата.

```
package edu.finki.np.av4;

import java.util.ArrayList;
import java.util.Random;

public class Box<T> {
    private ArrayList<T> items;

    public Box() {
        items = new ArrayList<T>();
    }

    public void add(T item) {
        items.add(item);
    }

    public boolean isEmpty() {
        return items.size() == 0;
    }

    public T drawItem() {
```

```
        if (isEmpty()) {
            return null;
        }
        Random random = new Random();
        return items.get(random.nextInt(items.size()));
    }

    public static void main(String[] args) {
        Box<String> stringBox = new Box<String>();
        stringBox.add("Dexter");
        stringBox.add("Seinfeld");
        stringBox.add("Barney");
        stringBox.add("Sheldon");
        stringBox.add("Costanza");
        stringBox.add("Hank");
        System.out.println(stringBox.drawItem());
        Box<Integer> intBox = new Box<Integer>();
        intBox.add(23);
        intBox.add(15);
        intBox.add(19);
        intBox.add(3);
        intBox.add(92);
        System.out.println(intBox.drawItem());
    }
}
```

4. Да се имплементира класа за податочна структура `PriorityQueue` со помош на `ArrayList`. `PriorityQueue` е податочна структура во која секоја елемент се додава заедно со неговиот приоритет (цел број). Приоритет да се дефинира така што оние елементи со најголема вредност на приоритет имаат повисок приоритет. Класата треба да ги имплементира следните методи:

   - `add(item, priority)` - Додава нов елемент со асоциран приоритет.
   - `remove()` - Го враќа елементот со најголем приоритет и го брише од редот. Ако редот е празен се враќа `null`.

   Пример за `priority queue` со стрингови:

```
q.add("X", 10);
q.add("Y", 1);
q.add("Z", 3);
System.out.println(q.remove()); // Returns X
System.out.println(q.remove()); // Returns Z
System.out.println(q.remove()); // Returns Y
```

   Тестирајте го редот со податоци со приоритет во различен редослед (пр., растечки, опаѓачки, мешан). Редот може да се имплементира со линеарно пребарување низ `ArrayList`.

```
package edu.finki.np.av4;

import java.util.ArrayList;

public class PriorityQueue<T> {
    private ArrayList<T> queue;
```

```
    private ArrayList<Integer> priorities;

    public PriorityQueue() {
        queue = new ArrayList<T>();
        priorities = new ArrayList<Integer>();

    }

    public void add(T item, int priority) {
        int i;
        for (i = 0; i < priorities.size(); i++) {
            if (priorities.get(i) < priority) {
                break;
            }
        }
        queue.add(i, item);
        priorities.add(i, priority);
    }

    public T remove() {
        if (queue.isEmpty()) {
            return null;
        }
        T item = queue.get(0);
        queue.remove(0);
        priorities.remove(0);
        return item;
    }

    public static void main(String[] args) {
        PriorityQueue<String> pq = new PriorityQueue<String>();
        pq.add("X", 0);
        pq.add("Y", 1);
        pq.add("Z", 3);
        System.out.println(pq.remove()); // Returns X
        System.out.println(pq.remove()); // Returns Z
        System.out.println(pq.remove()); // Returns Y
    }
}
```

5. Да се напише класа `MyMathClass`, во која ќе се имплементира статички метод `standardDeviation` кој како аргумент прима `ArrayList` од тип T, каде што T е нумерички тип (пр. Integer, Double, или било која класа која наследува од `java.lang.Number`) и враќа резултат `double` кој претставува стандардната девијација на вредностите во листата. Стандардна девијација се пресметува со следната формула: $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}, \mu = \frac{1}{N}\sum_{i=1}^{N} x_i$

   Вашата програма треба да генерира грешка при компајлирање ако методот за пресметување стандардна девијација се повика со `ArrayList` која е дефинирана со ненумерички тип (пр. `String`).

```
package edu.finki.np.av4;

import java.util.ArrayList;

public class MyMathClass {
```

```
    public static ArrayList<? super Double> result(double d) {
        return new ArrayList<Number>();
    }

    public static double standardDeviation(ArrayList<? extends
        Number> array) {
        double sum = 0;
        for (Number n : array) {
            sum += n.doubleValue();
        }
        double avg = sum / array.size();
        sum = 0;
        for (Number n : array) {
            sum += (avg - n.doubleValue()) * (avg - n.
                doubleValue());
        }
        return Math.sqrt(sum / array.size());
    }

    public static void main(String[] args) {
        ArrayList<Integer> ints = new ArrayList<Integer>();
        ints.add(1);
        ints.add(10);
        ints.add(20);
        ints.add(30);
        ints.add(40);
        ints.add(50);
        System.out.println(String.format("STD: %.2f",
                MyMathClass.standardDeviation(ints)));
        ArrayList<Double> doubles = new ArrayList<Double>();
        doubles.add(3.4);
        System.out.println(String.format("STD: %.2f",
                MyMathClass.standardDeviation(doubles)));
    }
}
```