Универзитет „Св. Кирил и Методиј“ - Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Напредно програмирање

Аудиториски вежби 6

Верзија 1.0, 20 Септември, 2016

Содржина

1. Генеричко програмирање

1.1. Кутија

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 како резултат. Ако се обидеме да цртаме со празна кутија се враќа `null`.

Да се напише main метод кој ја тестира класата.

Решение (`Box.java`)

```
package mk.ukim.finki.np.av6;

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

public class Box<T> {

    private ArrayList<T> items;

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

    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<>();
        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<>();
        intBox.add(23);
        intBox.add(15);
        intBox.add(19);
        intBox.add(3);
        intBox.add(92);
        System.out.println(intBox.drawItem());
    }
}
```

1.2. Priority Queue (Ред на чекање со приоритет)

Да се имплементира класа за податочна структура `PriorityQueue` со помош на `ArrayList`. `PriorityQueue` е податочна структура во која секој елемент е придружен заедно со неговиот приоритет (цел број). Приоритет да се дефинира така што оние елементи со најголема вредност 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`.

Решение (`PriorityQueue.java`)

```
package mk.ukim.finki.np.av6;

import java.util.ArrayList;

public class PriorityQueue<T> {
    private ArrayList<T> queue;
    private ArrayList<Integer> priorities;

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

    }

    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
    }
}
```

1.3. Стандардна девијација (MyMathClass)

Да се напише класа `MyMathClass`, во која ќе се имплементира статички метод `standardDeviation`. Методот 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

Решение (`MyMathClass.java`)

```
package mk.ukim.finki.np.av6;

import java.util.ArrayList;

public class MyMathClass {

    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<>();
        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<>();
        doubles.add(3.4);
        System.out.println(String.format("STD: %.2f",
                MyMathClass.standardDeviation(doubles)));
    }
}
```

2. Изворен код од примери и задачи

https://github.com/finki-mk/NP/

Source Code ZIP