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

Напредно програмирање

Аудиториски вежби 7

Верзија 1.0, 15 Ноември, 2016

Содржина

1. Рандомизација и случајни броеви

1.1. Финалисти

Ваша задача е да распределите три еднакви награди на 30 финалисти. За секој финалист има назначено број од 1 до 30. Напишете програма која случајно ги избира броевите на 3-те финалисти кои треба да добијат награда. Треба да внимавате на некои ограничувања при избирањето на наградените. На пример, одбирање на финалисти со броеви 3, 15 и 29 е валидно, но избирањето на 3, 3 и 31 како добитници не е валидно, затоа што финалистот со број 3 е избран два пати, а 31 не е валиден број на финалист.

Решение (`RandomPicker.java`)

```
package mk.ukim.finki.np.av7;

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

/**
 * Random picker
 */
public class RandomPicker {
    private final int n;

    public RandomPicker(int n) {
        this.n = n;
    }

    public List<Integer> pick(int x) {
        Random random = new Random();
        List<Integer> picked = new ArrayList<>();
        while (picked.size() != x) {
            int pick = random.nextInt(n) + 1;
            if (!picked.contains(pick)) {
                picked.add(pick);
            }
        }
        return picked;
    }
}
```

Решение (`Finalists.java`)

```
package mk.ukim.finki.np.av7;

import java.util.List;

public class Finalists {
    public static void main(String[] args) {
        RandomPicker picker = new RandomPicker(30);
        List<Integer> picked = picker.pick(3);
        System.out.println(picked);
    }
}
```

1.2. Benford’s Law

Задачата е дел од "Niffty Assignment" од авторот Steve Wolfman (http://nifty.stanford.edu/2006/wolfman-pretid). Дадена ни е листа на броеви од податочни извори од реалниот живот, на пример, листа со број на:

- студенти запишани на различни курсеви
- коментари на различни Facebook статуси
- книги во различни библиотеки
- бројот на гласови по избирачко место, итн.

Логични би било почетната цифра на секој број во листата да биде 1-9 со приближно еднаква веројатност. Меѓутоа, законот на Бенфорд (Benford’s Law) тврди дека почетната цифра 1 е се појавува околу 30% од времето и оваа вредност опаѓа со големината на цифрата. Почетна цифра 9 се појавува само околу 5% од времето. Да се напише програма која го тестира законот на Бенфорд. Соберете листа од најмалку 10 броеви од извори од реланиот живот и ставете ги во текстуална датотека. Вашата програма треба ги измине сите броеви и треба да изброи колку броеви се со прва цифра 1, колку со прва цифра 2, итн. За секоја цифра да се отпечати процентот на застапеност како прва цифра.

Решение (`BenfordLawTest.java`*)*

```
package mk.ukim.finki.np.av7;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.Arrays;
import java.util.List;

public class BenfordLawTest {
    static final String INPUT_FILE = "examples/data/librarybooks.txt";
    /*
     * librarybooks.txt
     * (* Library holdings (# of books in each library), *)
       (* collected by Christian Ayotte.                 *)
       (* Labels not available.                          *)

       livejournal.txt
       (* LiveJournal data collected by Shirley Man from *)
       (* http://www.livejournal.com/stats/stats.txt     *)
       (* Number of new accounts on LiveJournal,         *)
       (* day by day from 2000/1/1 to 2005/2/28          *)
       (* Individual data are NOT labelled.              *)

       sunspots.txt
       (* Sunspot data collected by Robin McQuinn from *)
       (* http://sidc.oma.be/html/sunspot.html         *)

     */

    public static void main(String[] args) throws FileNotFoundException {
        NumbersReader numbersReader = new LineNumbersReader();
        List<Integer> numbers = numbersReader.read(new FileInputStream(INPUT_FILE));
        BenfordLawTest benfordLawTest = new BenfordLawTest();
        int[] count = benfordLawTest.counts(numbers);
        CountVisualizer visualizer = new CountVisualizer(100);
        visualizer.visualize(System.out, count);
    }

    public int[] counts(List<Integer> numbers) {
        int[] result = new int[10];
        for (Integer number : numbers) {
            int digit = firstDigit(number);
            result[digit]++;
        }
        return result;
    }

    public int[] countsFunc(List<Integer> numbers) {
        return numbers.stream()
                .map(BenfordLawTest::firstDigit)
                .map(x -> {
                    int[] res = new int[10];
                    res[x]++;
                    return res;
                })
                .reduce(new int[10], (left, right) -> {
                    Arrays.setAll(left, i -> left[i] + right[i]);
                    return left;
                });
    }

    static int firstDigit(int num) {
        while (num >= 10) {
            num /= 10;
        }
        return num;
    }
}
```

*Решение (*NumbersReader.java*)*

```
package mk.ukim.finki.np.av7;

import java.io.InputStream;
import java.util.List;

/**
 * Interface for reading numbers from InputStream
 */
public interface NumbersReader {

    List<Integer> read(InputStream inputStream);

}
```

*Решение (*LineNumbersReader.java*)*

```
package mk.ukim.finki.np.av7;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.Collections;
import java.util.List;
import java.util.stream.Collectors;

/**
 * Implementation for reading single number per line
 */
public class LineNumbersReader implements NumbersReader {
    @Override
    public List<Integer> read(InputStream inputStream) {
        try (BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream
))) {
            return reader.lines()
                    .filter(line -> !line.isEmpty())
                    .map(line -> Integer.parseInt(line.trim()))
                    .collect(Collectors.toList());
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }
        return Collections.emptyList();
    }
}
```

*Решение (*SunspotNumbersReader.java*)*

```
package mk.ukim.finki.np.av7;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.Collections;
import java.util.List;
import java.util.stream.Collectors;

/**
 * Implementation for reading numbers from sunspots.txt
 */
public class SunspotNumbersReader implements NumbersReader {
    @Override
    public List<Integer> read(InputStream inputStream) {
        try (BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream
))) {
            return reader.lines()
                    .filter(line -> !line.isEmpty())
                    .map(line -> {
                        String[] parts = line.split("\\s+");
                        return Integer.parseInt(parts[parts.length - 1]);
                    })
                    .collect(Collectors.toList());
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }
        return Collections.emptyList();
    }
}
```

*Решение (*CountVisualizer.java*)*

```
package mk.ukim.finki.np.av7;

import java.io.OutputStream;
import java.io.PrintWriter;

/**
 * Count visualizer using one * per n units
 */
public class CountVisualizer {
    private final int n;

    public CountVisualizer(int n) {
        this.n = n;
    }

    public void visualize(OutputStream outputStream, int[] counts) {
        PrintWriter writer = new PrintWriter(outputStream);
        for (Integer count : counts) {
            while (count > 0) {
                writer.print("*");
                count -= n;
            }
            writer.println();
        }
        writer.flush();
    }
}
```

1.3. Arrange Letters (from codefu.com.mk)

You are given a string and your task is to arrange the words in the string as follows:

- each word should begin with a capital letter
- the lowercase letters should be arranged alphabetically

Afterwards, the words in the sentence should be arranged alphabetically.

Example:

"kO pSk sO" should return: "Ok Os Skp"

Input parameters:

- sentence - String

Constraints:

- sentence will have between 1 and 1000 characters inclusive,
- every character will be a letter (’a’-’z’ ’A’-’Z’) or ’ ’ - space.
- each word in sentence will have exactly one capital letter.
- there will be no two consecutive spaces, and no spaces at the beginning or end of the string.
- a word may consist of only one capital letter

Решение (`ArrangeLetters.java`)

```
package mk.ukim.finki.np.av7;

import java.util.Arrays;
import java.util.stream.Collectors;


public class ArrangeLetters {
    public String arrange(String input) {
        String[] parts = input.split("\\s+");
/*        IntStream.range(0, parts.length)
                .mapToObj(i -> {
                    char[] part = parts[i].toCharArray();
                    Arrays.sort(part);
                    return new String(part);
                }).sorted();*/

        for (int i = 0; i < parts.length; ++i) {
            char[] w = parts[i].toCharArray();
            Arrays.sort(w);
            parts[i] = new String(w);
        }
        return Arrays.stream(parts)
                .sorted()
                .collect(Collectors.joining(" "));
    }
}
```

2. Изворен код од примери и задачи

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

Source Code ZIP