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

Напредно програмирање

Аудиториски вежби 5

Верзија 1.0, 20 Септември, 2016

Содржина

1. ArrayList

`ArrayList` е класа од стандарадните библиотеки во Java која го имплементира интерфејсот `List`. За разлика од низите, кои имаат фиксна должина откако ќе се креираат, `ArrayList` е објект кој може да се проширува додека програмата се извршува. Генерално, `ArrayList` ја има истата улога како и низите, со тоа што може да ја менува својата должина за време на извршувањето на програмата. Имплементирана е со помош на низа како приватна класна променлива.

1.1. Зошто секогаш да не користиме ArrayList наместо низи?

1. `ArrayList` е по неефикасна од низа
2. Не подржува нотација на големи загради `[]`
3. Основниот тип на ArrayList мора да биде класа (или друг тип референца): не може да биде примитивен тип `int`, `float`, `double`,...

1.2. Користење на `ArrayList`

- За да се користи ArrayList, треба да се вклучи од пакетот `java.util.ArrayList`
- Се креира и именува на ист начин како и објект од било која класа, освен што мора да се специфицира најзиниот тип на следниов начин:

```
ArrayList<BaseType> list = new ArrayList<>();
```

- Почетниот капацитет може да се проследи како аргумент на конструкторот
- Следниот код креира `ArrayList` кој чува референци од тип `String` и има почетен капацитет од 20 членови

```
ArrayList<String> strings = new ArrayList<>(20);
```

Специфицирање на почетниот капацитет не ја ограничува големината до која `ArrayList` може да расте

1.3. Најчесто користени методи

- add се користи за дадовање на елемент
- Постои и преоптоварена верзија со два аргументи кој овозможува да се додаде елемент на одредена позиција зададена преку индекс
- Методот `size` се користи да се добие бројот на елементи

```
int howMany = list.size();
```

- Методот `set` се користи за менување на постоечки елемент
- Методот get се користи за пристапување до одреден постоечки елемент

И двата методи го примаат за аргумент индексот (0 базиран) на елементот кој сакаме да го пристапиме.

```
list.set(index, "something else");
String thing = list.get(index);
```

2. Текстуални датотеки

2.1. Word Count

Да се напише програма која го прикажува бројот на знаци, бројот на зборови и бројот на редови во датотеките чии што имиња се задаваат како аргументи на командна линија.

Решение 2

```
package mk.ukim.finki.np.av5;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Arrays;
import java.util.function.Consumer;
import java.util.regex.Pattern;
import java.util.stream.Collectors;

/**
 * Argument: examples/data/quotes.txt
 */
public class WordCount {

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

```
        StringBuilder result = new StringBuilder();
        for (String fileName : args) {
            try {
                String wordCount = processFile(fileName);
                //wordCount = processWithMapReduce(fileName);
                result.append(String.format("%s -> %s\n", fileName, wordCount));
            } catch (IOException e) {
                System.err.println(e.getMessage());
            }
        }
        System.out.println(result.toString());
    }

    /**
     * Solution using {@link BufferedReader} reading line by line
     */
    private static String processFile(String fileName) throws IOException {
        int linesCount = 0;
        int wordsCount = 0;
        int charactersCount = 0;
        try (BufferedReader bufferedReader = new BufferedReader(new FileReader(fileName)))
{
            String line;
            while ((line = bufferedReader.readLine()) != null) {
                linesCount++;
                String[] words = line.split("\\s+");
                wordsCount += words.length;
                charactersCount += line.length() + 1;
            }
        }
        return String.format("%d %d %d", linesCount, wordsCount, charactersCount);
    }

    /**
     * Solution using {@link Consumer< FileCounts >} function
     */
    private static String processWithConsumer(String fileName) throws IOException {
        FileCounts fileCounts = new FileCounts();
        Files.lines(Paths.get(fileName))
                .forEach(fileCounts);
        return fileCounts.toString();
    }

    /**
     * Solution using map-reduce and long[] array as data holder
     */
    private static String processWithMapReduce(String fileName) throws IOException {
        Pattern word = Pattern.compile("\\s+");
        long[] result = Files.lines(Paths.get(fileName))
                .map(line -> {
                    long words = word.split(line).length;
                    return new long[]{1, words, line.length() + 1};
                })
                .reduce(new long[]{0, 0, 0},
                        (left, right) -> {
                            long[] sum = new long[3];
                            Arrays.setAll(sum, i -> left[i] + right[i]);
                            return sum;
                        });
        return Arrays.stream(result)
                .mapToObj(l -> String.format("%d", l))
                .collect(Collectors.joining(" "));

    }

    static class FileCounts implements Consumer<String> {
        long lines;
        long chars;
        long words;

        public FileCounts() {
            this.lines = 0;
            this.chars = 0;
            this.words = 0;
        }
```

```
        @Override
        public void accept(String line) {
            ++lines;
            this.chars += line.length() + 1;
            this.words += line.split("\\s+").length;
        }

        @Override
        public String toString() {
            return String.format("%d %d %d", lines, words, chars);
        }
    }

}
```

2.2. Oldest Person

Во секој ред од една датотека се запишани име (`String`) и возраст (`int`). Да се напише програма која ќе го отпечати името и возраста на највозрасното лице.

```
Кристијан 25
Дритон 39
Ристе 17
Лусијана 28
Бобан 7
Оливера 71
Ана 14
Димитар 56
Диме 11
Билјана 12
```

Решение 3

```java
package mk.ukim.finki.np.av5;

import java.io.*;
import java.util.Scanner;

public class FindOldest {
    public static void main(String[] args) {
        try {
            //findWithScanner(new FileInputStream(args[0]));
            findWithStream(new FileInputStream(args[0]));
        } catch (FileNotFoundException e) {
            System.err.println(e.getMessage());
        }
    }

    /**
     * Imperative solution using {@link Scanner}
     */
    static void findWithScanner(InputStream inputStream) {
        try (Scanner scanner = new Scanner(inputStream)) {
            String oldestName = null;
            int maxAge = Integer.MIN_VALUE;
            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                String[] parts = line.split("\\s+");
                String name = parts[0];
                int age = Integer.parseInt(parts[1]);
                if (age > maxAge) {
                    maxAge = age;
                    oldestName = name;
                }
            }
            System.out.println("Name: " + oldestName);
            System.out.println("Age: " + maxAge);
        }
    }

    /**
     * Functional solution using {@link BufferedReader}
     */
    static void findWithStream(InputStream inputStream) {
        BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));
        String max = reader.lines()
                .reduce("", (left, right) -> {
                    if (left.length() == 0) return right;
                    if (right.length() == 0) return left;
                    String[] leftParts = left.split("\\s+");
                    String[] rightParts = right.split("\\s+");
                    int leftAge = Integer.parseInt(leftParts[1]);
                    int rightAge = Integer.parseInt(rightParts[1]);
                    return leftAge > rightAge ? left : right;
                });
        System.out.println(max);
    }
}
```

2.3. Оценки за курс

Да се напише програма која пресметува оценки за одреден курс. Во програмата прво се вчитува името на датотеката која ги содржи информациите за резултатите од испитите. Секој ред од датотеката е во следниот формат:

`LastName:FirstName:Exam1:Exam2:Exam3`

Испитите се вреднуваат тежински и тоа 25% за првиот, 30% за вториот и 45% за третиот испит. Врз основа на ова, конечната оценка се добива според следната скала:

- A [90 - 100)
- B [80 - 90)
- C [70 - 80)
- D [60 - 70)
- F [0 - 60)

Вашата програма треба да отпечати на стандардниот излез листа од студенти подредени според оценката во опаѓачки редослед, во следниот формат:

```
LastName FirstName LetterGrade
```

Исто така во датотека чие што име се внесува од стандарден влез се запишуваат резултатите во следниот формат:

```
LastName FirstName Exam1 Exam2 Exam3 TotalPoints LetterGrade
```

Откако ќе се запише оваа содржина во датотека, се печати на стандарден излез дистрибуцијата на оценките.

Пример ако содржината на датотеката е:

```
Doe:John:100:100:100
Pantz:Smartee:80:90:80
```

Излезот е:

```
Doe John A
Pantz Smartee B
```

Излезот во датотеката ќе биде:

```
Doe John 100 100 100 100 A
Pantz Smartee 80 90 80 83 B
A 1
B 1
C 0
D 0
F 0
```

Решение (`Student.java`)

```
package mk.ukim.finki.np.av5;

/**
 * Student class
 */
class Student implements Comparable<Student> {
    private String firstName;
    private String lastName;
    private int exam1;
    private int exam2;
    private int exam3;
    private char grade;
    private double total;

    public Student(String firstName, String lastName, int exam1, int exam2,
                   int exam3) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.exam1 = exam1;
        this.exam2 = exam2;
        this.exam3 = exam3;
        this.total = exam1 * .25 + exam2 * .3 + exam3 * .45;
        if (this.total >= 91) {
            this.grade = 'A';
        } else if (this.total >= 81) {
            this.grade = 'B';
        } else if (this.total >= 71) {
            this.grade = 'C';
        } else if (this.total >= 61) {
            this.grade = 'D';
        } else {
            this.grade = 'F';
        }
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public int getExam1() {
        return exam1;
    }

    public int getExam2() {
        return exam2;
    }

    public int getExam3() {
        return exam3;
    }

    public char getGrade() {
        return grade;
    }

    public double getTotal() {
        return total;
    }

    @Override
    public int compareTo(Student other) {
        return Double.compare(this.total, other.total);
    }

    public static Student fromString(String line) {
        String[] parts = line.split(":");
        int exam1 = Integer.parseInt(parts[2]);
        int exam2 = Integer.parseInt(parts[3]);
        int exam3 = Integer.parseInt(parts[4]);
```

```
            return new Student(parts[0], parts[1], exam1, exam2, exam3);
        }

        @Override
        public String toString() {
            return String.format("%s %s %d %d %d %.0f %c", firstName, lastName,
                    exam1, exam2, exam3, total, grade);
        }

    }
```

Решение (`CalculateGrades.java`*)*

```
package mk.ukim.finki.np.av5;

import java.io.*;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

public class CalculateGrades {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String filename = scanner.nextLine();
        BufferedReader fileReader = null;
        ArrayList<Student> students = new ArrayList<Student>();
        int[] gradesDistribution = new int[5];
        try {
            fileReader = new BufferedReader(new FileReader(filename));
            String line = null;
            while ((line = fileReader.readLine()) != null) {
                String[] parts = line.split(":");
                int exam1 = Integer.parseInt(parts[2]);
                int exam2 = Integer.parseInt(parts[3]);
                int exam3 = Integer.parseInt(parts[4]);
                Student s = new Student(parts[0], parts[1], exam1, exam2, exam3);
                students.add(s);
                if (s.getGrade() == 'A') {
                    gradesDistribution[0]++;
                } else if (s.getGrade() == 'B') {
                    gradesDistribution[1]++;
                } else if (s.getGrade() == 'C') {
                    gradesDistribution[2]++;
                } else if (s.getGrade() == 'D') {
                    gradesDistribution[3]++;
                } else {
                    gradesDistribution[4]++;
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                fileReader.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        Collections.sort(students);
        for (Student s : students) {
            System.out.println(String.format("%s %s %c", s.getFirstName(),
                    s.getLastName(), s.getGrade()));
        }
        String outputFile = scanner.nextLine();
        PrintWriter printWriter = null;
        try {
            printWriter = new PrintWriter(new FileWriter(outputFile));
            for (Student s : students) {
                printWriter.println(s);
            }
            printWriter.println(String.format("A : %d", gradesDistribution[0]));
            printWriter.println(String.format("B : %d", gradesDistribution[1]));
            printWriter.println(String.format("C : %d", gradesDistribution[2]));
            printWriter.println(String.format("D : %d", gradesDistribution[3]));
            printWriter.println(String.format("F : %d", gradesDistribution[4]));
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            printWriter.close();
        }

    }
}
```

Решение со Java 8 Streams (`CalculateGradesImproved.java`*)*

```
package mk.ukim.finki.np.av5;

import java.io.BufferedWriter;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

/**
 * File path for test: examples/data/grades.txt
 */
public class CalculateGradesImproved {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String filename = scanner.nextLine();
        List<Student> students = loadStudents(filename);
        students.stream()
                .sorted()
                .forEach(each -> System.out.println(String.format("%s %s %c",
                        each.getFirstName(), each.getLastName(), each.getGrade())));
        int[] distribution = findDistribution(students);
        printDistribution(distribution);
        filename = scanner.nextLine();
        writeToFile(students, filename);
    }

    static List<Student> loadStudents(String fileName) {
        try {
            return Files.lines(Paths.get(fileName))
                    .map(Student::fromString)
                    .collect(Collectors.toList());
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }
        System.exit(-1);
        return null;
    }

    static int[] findDistribution(List<Student> students) {
        return students.stream()
                .map(Student::getGrade)
                .collect(() -> new int[6],
                        (ints, character) -> ints[character - 'A']++,
                        (left, right) -> Arrays.setAll(left, i -> left[i] + right[i])
                );
    }

    static void printDistribution(int[] distribution) {
        IntStream.range(0, distribution.length)
                .mapToObj(i -> String.format("%c : %d", 'A' + i, distribution[i]))
                .forEach(System.out::println);
    }

    static void writeToFile(List<Student> students, String fileName) {
        try (BufferedWriter writer = Files.newBufferedWriter(Paths.get(fileName),
StandardOpenOption.CREATE)) {
            students.forEach(each -> {
                try {
                    writer.write(each.toString());
                } catch (IOException e) {
                    System.err.println(e.getMessage());
                }
            });
        } catch (IOException e) {
            System.exit(-1);
        }
    }
```

```
}
```

3. Бинарни датотеки

3.1. Просек на случајни броеви

Да се напише програма која запишува n случајни цели броеви во бинарна датотека, потоа ги вчитува и го пресметува нивниот просек.

Решение 1

```
package mk.ukim.finki.np.av5;

import java.io.*;
import java.util.Random;

public class BinaryNumbers {
    static final String FILE_NAME = "numbers.dat";

    private static void generateFile(int n) {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream
(FILE_NAME))) {
            Random random = new Random();
            for (int i = 0; i < n; i++) {
                int next = random.nextInt(1000);
                oos.writeInt(next);
            }
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }
    }

    private static double findNumbersAverage() {
        int total = 0;
        double sum = 0;
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME)
)) {
            try {
                while (true) {
                    int num = ois.readInt();
                    sum += num;
                    total++;
                }
            } catch (EOFException e) {
                System.out.println("All numbers are read");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return sum / total;
    }

    public static void main(String[] args) {
        generateFile(1000);
        double avg = findNumbersAverage();
        System.out.println("Average: " + avg);
    }
}
```

3.2. Листа со резултати

Да се напише класа за резултат (`Score`) од некоја видео игра. Секој резултат се состои од името и бројот на поени. Да се напише нова класа (`Scoreboard`) која ќе ги чува најдобрите N резултати. Оваа класа треба да има можност за додавање нов резултат и прикажување на тековните резултати.

За дома

4. Изворен код од примери и задачи

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

Source Code ZIP