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

# Напредно програмирање

Аудиториски вежби 3

Верзија 1.0, 20 Септември, 2016

# Содржина

# 1. Читање од SI

## 1.1. Задача

Да се напише програма едноставен калкулатор. Калкулаторот чува еден број од тип `double` со име резултат и неговата почетна вредност е `0.0`. Во циклус му се дозволува на корисникот да додаде, одземе, помножи или подели со втор број. Резултатот од овие операции е новата вредност на резултатот. Пресметката завршува кога корисникот ќе внесе R за `result` (како мала или голема буква). Корисникот може да направи уште една пресметка од почеток или да ја заврши програмата `(Y/N)`. Ако корисникот внесе различен знак за оператор од `+`, `-`, `*` или `/`, тогаш се фрла исклучок `UnknownOperatorException` и се чека повторно на внес.

*Пример форматот на влезните податоци:*

```
Calculator is on.
result = 0.0
+5
result + 5.0 = 5.0
new result = 5.0
* 2.2
result * 2.2 = 11.0
updated result = 11.0
% 10
% is an unknown operation.
Reenter, your last line:
* 0.1
result * 0.1 = 1.1
updated result = 1.1
r
Final result = 1.1
Again? (y/n)
yes
result = 0.0
+10
result + 10.0 = 10.0
new result = 10.0
/2
result / 2.0 = 5.0
updated result = 5.0
r
Final result = 5.0
Again? (y/n)
N
End of Program
```

*Решение 1*

```
package mk.ukim.finki.np.av3;

public class Calculator {
    private double result;
    private static final char PLUS = '+';
    private static final char MINUS = '-';
    private static final char MULTIPLY = '*';
    private static final char DIVIDE = '/';

    public Calculator() {
        result = 0;
    }

    public String init() {
        return String.format("result = %f", result);
    }

    public double getResult() {
        return result;
    }

    public String execute(char operator, double value)
            throws UnknownOperatorException {

        if (operator == PLUS) {
            result += value;
        } else if (operator == MINUS) {
            result -= value;
        } else if (operator == MULTIPLY) {
            result *= value;
        } else if (operator == DIVIDE) {
            result /= value;
        } else {
            throw new UnknownOperatorException(operator);
        }
        return String.format("result %c %f = %f", operator, value, result);
    }

    class UnknownOperatorException extends Exception {
        public UnknownOperatorException(char operator) {
            super(String.format("%c is unknown operation", operator));
        }
    }

    @Override
    public String toString() {
        return String.format("updated result = %f", result);
    }

}
```

*Решение 1 (Тест програма)*

```
package mk.ukim.finki.np.av3;

import java.util.Scanner;

import edu.finki.np.av3.Calculator.UnknownOperatorException;

public class CalculatorTest {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (true) {
            Calculator calculator = new Calculator();
            System.out.println(calculator.init());
            while (true) {
                String line = scanner.nextLine();
                if (line.length() == 1) {
                    char r = line.charAt(0);
                    r = Character.toLowerCase(r);
                    if (r == 'r') {
                        System.out.println(String.format("final result = %f",
                                calculator.getResult()));
                        break;
                    }
                }
                String[] parts = line.split(" ");
                char operator = parts[0].charAt(0);
                double value = Double.parseDouble(parts[1]);
                try {
                    String result = calculator.execute(operator, value);
                    System.out.println(result);
                    System.out.println(calculator);
                } catch (UnknownOperatorException e) {
                    System.out.println(e.getMessage());
                }

            }

            System.out.println("(Y/N)");
            String s = scanner.nextLine();
            char c = s.charAt(0);
            c = Character.toLowerCase(c);
            if (c == 'n') {
                break;
            }
        }
    }
}
```

# 2. Текстуални датотеки

## 2.1. Задача

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.av3;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class WordCount {
    public static void main(String[] args) {
        StringBuilder result = new StringBuilder();
        for (String filename : args) {
            String wordCount = processFile(filename);
            result.append(wordCount);
        }
        System.out.println(result.toString());
    }

    private static String processFile(String filename) {
        int linesCount = 0;
        int wordsCount = 0;
        int charactersCount = 0;
        BufferedReader bufferedReader = null;
        try {
            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();
            }
            bufferedReader.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return String.format("%d %d %d\n", linesCount, wordsCount,
                charactersCount);
    }
}
```

## 2.2. Задача

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*

```
package mk.ukim.finki.np.av3;

import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class FindOldest {
    public static void main(String[] args) {
        Scanner fileScanner = null;
        try {
            fileScanner = new Scanner(new FileReader(args[0]));
            String oldestName = null;
            int maxAge = Integer.MIN_VALUE;
            while (fileScanner.hasNextLine()) {
                String line = fileScanner.nextLine();
                String[] parts = line.split(" ");
                String name = parts[0];
                int age = Integer.parseInt(parts[1]);
                if (age > maxAge) {
                    maxAge = age;
                    oldestName = name;
                }
            }
            fileScanner.close();
            System.out.println("Name: " + oldestName);
            System.out.println("Age: " + maxAge);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

## 2.3. Задача

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 испититите. 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:

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

Вашата програма треба да отпечати на стандардниот излез листа од студенти со оценката во следниот формат:

```
LastName FirstName LetterGrade
```

Исто така во датотека чие што име се внесува од стандарден влез се запишуваат резултатите во следнио 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 4*

```
package mk.ukim.finki.np.av3;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
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();
        }

    }
}

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 o) {
        if (this.total < o.total) {
            return 1;
        } else if (this.total > o.total) {
            return -1;
        } else {
            return 0;
        }
    }

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

}
```

# 3. Бинарни датотеки

## 3.1. Задача

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

*Решение 1*

```
package mk.ukim.finki.np.av3;

import java.io.EOFException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.Random;

public class BinaryNumbers {
    private static void generateFile(int n) {
        ObjectOutputStream oos = null;
        try {
            oos = new ObjectOutputStream(new FileOutputStream("numbers.dat"));
            Random random = new Random();
            for (int i = 0; i < n; i++) {
                int next = random.nextInt(1000);
                oos.writeInt(next);
            }
            oos.flush();
            oos.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

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

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

## 3.2. Задача

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