



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

## Напредно програмирање

Аудиториски вежби 2

Верзија 1.0, 23 Септември, 2016

# Содржина

1. String prefix . . . . .	1
2. Матрица (сума и просек) . . . . .	1
3. Катанец (CombinationLock) . . . . .	2
4. BigComplex . . . . .	3
5. Изворен код од примери и задачи . . . . .	5

## 1. String prefix

Да се напише метод кој враќа точно ако String str1 е префикс на String str2. Не е дозволено користење на готови методи за пребарување, единствено дозволено е користење на charAt.

### Решение (isPrefix)

```
public static boolean isPrefix(String str1, String str2) {
    if (str1.length() > str2.length()) {
        return false;
    }
    for (int i = 0; i < str1.length(); i++) {
        if (str1.charAt(i) != str2.charAt(i)) {
            return false;
        }
    }
    return true;
}
```

### Решение со Java 8 Streams (isPrefixStream)

```
public static boolean isPrefixStream(String str1, String str2) {
    if (str1.length() > str2.length()) {
        return false;
    }
    return IntStream.range(0, str1.length())
        .allMatch(i -> str1.charAt(i) == str2.charAt(i));
}
```

## 2. Матрица (сума и просек)

Да се имплементираат следните методи кои примаат дво-димензионални низи од double и враќаат како резултат сума и просек.

```
public static double sum(double[][] a)
public static double average(double[][] a)
```

### Решение (sum)

```
public static double sum(double[][] a) {
    double s = 0;
    for (int i = 0; i < a.length; i++) {
        for (int j = 0; j < a[i].length; j++) {
            s += a[i][j];
        }
    }
    return s;
}
```

## Решение со Java 8 Streams (sumStream)

```
public static double sumStream(double[][] a) {  
    return Arrays.stream(a)  
        .mapToDouble(row -> Arrays.stream(row).sum())  
        .sum();  
}
```

## Решение 2 (average)

```
public static double average(double[][] a) {  
    double s = 0;  
    for (int i = 0; i < a.length; i++) {  
        for (int j = 0; j < a[i].length; j++) {  
            s += a[i][j];  
        }  
    }  
    return s / (a.length * a[0].length);  
}
```

## 3. Катанец (CombinationLock)

Катанец со комбинации Combination Lock ги има следните својства:

- комбинацијата (секвенца од 3 цифри) е скриена
- катанецот може да се отвори само ако се внесе точната комбинација
- комбинацијата може да се промени само ако се знае претходната комбинација.

Да се дизајнира класа со јавни методи open и changeCombo и приватни податоци кои ја чуваат комбинацијата. Комбинацијата се поставува преку конструкторот.

## Решение (CombinationLock.java)

```
package mk.ukim.finki.np.av2;

public class CombinationLock {

    private int combination;
    private boolean isOpen;

    public CombinationLock(int combination) {
        this.combination = combination;
        this.isOpen = false;
    }

    public boolean open(int combination) {
        this.isOpen = (this.combination == combination);
        return this.isOpen;
    }

    public boolean changeCombo(int oldCombination, int newCombination) {
        boolean isCorrect = (this.combination == oldCombination);
        if (isCorrect) {
            this.combination = newCombination;
        }
        return isCorrect;
    }

    public boolean isOpen() {
        return isOpen;
    }
}
```

## 4. BigComplex

Комплексен број се состои од реален дел и имагинарен дел. Да се имплементира класа `BigComplex`, во која реалниот и имагинарниот дел ќе се чуваат во објекти од класата `BigDecimal`.

## Решение (BigComplex.java)

```
package mk.ukim.finki.np.av2;

import java.math.BigDecimal;

public final class BigComplex {

    private final BigDecimal real;
    private final BigDecimal imag;

    public BigComplex(BigDecimal real, BigDecimal imag) {
        this.real = real;
        this.imag = imag;
    }

    public BigComplex add(BigComplex complex) {
        BigDecimal real = this.real.add(complex.real);
        BigDecimal imag = this.real.add(complex.imag);
        return new BigComplex(real, imag);
    }
}
```

### *За дома*



Имплементирајте ги останатите методи за одземање `subtract`, множење `multiply` и делење `divide`.

## 5. Изворен код од примери и задачи

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

Source Code ZIP