

## Contents

<b>Laborator6</b>	<b>2</b>
<b>1 Clase abstracte și interfețe</b>	<b>2</b>
1.1 Ce este o interfață ?	2
1.1.1 Definirea unei interfețe	2
1.1.2 Componente posibile	2
1.1.3 Exemplu	2
1.2 Ce este o clasă abstractă ?	3
<b>2 Probleme de laborator</b>	<b>3</b>
2.1 Problema 1	3
2.2 Problema 2	3
2.3 Problema 3	4
2.4 Problema 4	4
2.5 Problema 5	4
2.6 Problema 6 (bonus)	4

---

<sup>1</sup><http://www.google.ro/>

# Laborator6

## Programare Orientata pe Obiecte: Laborator 6

### 1 Clase abstracte și interfețe

#### 1.1 Ce este o interfață ?

- Colecție de metode abstracte și declarații de constante.
- Definește un set de metode dar nu specifică nici o implementare pentru ele.
- O clasă care implementează o interfață trebuie obligatoriu să specifice implementări pentru toate metodele interfeței.

##### 1.1.1 Definirea unei interfețe

```
[public] interface NumeInterfata extends [SuperInterfata1, Superinterfata2, ...]
{
    // Declaratii de constante
    // Declaratii de metode
}
```

##### 1.1.2 Componente posibile

- **constante** : acestea pot fi sau nu declarate cu modificatorii *public*, *static* și *final*. Constantele unei interfețe trebuie obligatoriu inițializate.
- **metode fără implementare** : acestea pot fi sau nu declarate cu modificatorul *public*, care este implicit; nici un alt modificator nu poate apărea în declarația unei metode a unei interfețe

##### 1.1.3 Exemplu

```
public interface Stack {
    void push(Object item) throws StackException;
    void pop() throws StackException;
    Object peek() throws StackException;
    boolean empty();
    String toString();
}
// Definim o exceptie proprie
public class StackException extends Exception {
    public StackException() {
        super();
    }
    public StackException(String msg) {
        super(msg);
    }
}
public class Stiva implements Stack {
    private Object items[]; // Lista de obiecte din stiva
    private int n = 0; // numarul de elemente din stiva

    public Stiva(int max) {
        items = new Object[max];
    }
}
```

```

    }
    public void push(Object item) throws StackException
        if (n == items.length)
            throw new StackException("Stiva plina");
            items[n++] = item;
    }
    public void pop() throws StackException {
        if (empty())
            throw new StackException("Stiva vida");
            items[--n] = null;
    }
    ...
}

```

## 1.2 Ce este o clasă abstractă ?

- O clasă declarată cu modificatorul *abstract*
- Nu poate fi instanțiată, dar poate fi extinsă
- Dacă o clasă conține o metodă declarată cu modificatorul *abstract*, automat clasa trebuie să conțină acest modificator
- Poate să conțină declarații de metode fără implementări (asemeni interfețelor), dar și metode cu implementări.

## 2 Probleme de laborator

### 2.1 Problema 1

(2 puncte) Să se definească o clasă *Name* care implementează interfața *Comparable* și conține ca variabile separate: numele și prenumele unei persoane, cu următoarele metode redefinite:

- `public boolean equals (Object);`
- `public int compareTo (Object);`
- `public String toString();`

Să se scrie un program pentru crearea unui vector (intrinsec - *Name []* - sau obiect din clasa *Vector*) ce conține obiecte *Name*, ordonarea lui (cu metoda "Arrays.sort" sau "Collections.sort") și afișarea lui.

Exemplu de date:

```

Name tab[] = { new Name("Popa", "Mihai"),
               new Name("Pop", "Vasile"),
               new Name("Popa", "Mihaela") };

```

### 2.2 Problema 2

(2 puncte) Să se definească o clasă *SortedVector* derivată din *Vector*, care să permită ordonarea după orice criteriu, specificat de utilizator la construirea unui obiect *SortedVector*. Clasa va conține o variabilă de tip *Comparator*, inițializată de un constructor cu argument de tip *Comparator* și folosită de metoda *Collections.sort*.

Să se definească o clasă *Pair* care conține două variabile de tip *Object*, cu metodele *equals* și *toString* redefinite. Să se scrie un program pentru crearea a doi vectori *SortedVector* de obiecte *Pair*, unul ordonat după primul obiect din pereche și altul ordonat după al doilea obiect.

Clasa *Pair* conține o variabilă *String* și o variabilă *Integer*.

### 2.3 Problema 3

(2 puncte) Să se definească o clasă filtru pentru selecție de fișiere după o listă de extensii (tipuri de fișiere). Clasa implementează una din interfețele *FileFilter* sau *FilenameFilter* și conține un vector cu tipurile de fișiere acceptate. Program pentru afișarea fișierelor legate de Java, cu oricare din extensiile *java*, *class* sau *jar*.

### 2.4 Problema 4

(2 puncte) Un fișier text conține linii formate din mai multe cuvinte separate prin spații albe. Să se scrie un program pentru ordonarea unui astfel de fișier după oricare din cuvintele din linie (ex. după primul cuvânt din fiecare linie, al doilea, etc.). Programul primește în linia de comandă numele fișierului și numărul cuvântului după care se ordonează. Liniile ordonate se afișează pe ecran. Se va folosi un vector de linii din fișier și funcția *Arrays.sort* cu parametru *Comparator*.

Se va defini și folosi o clasă care implementează interfața *Comparator* având un constructor cu parametru întreg (număr cuvânt din linie).

### 2.5 Problema 5

(2 puncte) Să se scrie o funcție statică pentru ordonarea descrescătoare, în ordine naturală, a unui vector de obiecte *Object*, folosind funcția *Arrays.sort* și o clasă comparator anonimă definită chiar în apelul funcției *sort*. Să se scrie un program pentru verificarea funcției prin ordonare vector de șiruri.

### 2.6 Problema 6 (bonus)

(1 punct) Funcție statică pentru ordonarea unei matrice de obiecte *a* după valorile dintr-o coloană dată *c* (ordonare naturală crescătoare):

```
public static void tabsort (final Object[] [] a, final int c) {...}
```

Clasa comparator este o clasă inclusă anonimă definită la apelul funcției *Arrays.sort*, în cadrul funcției *tabsort*. Exemplu de program pentru verificarea funcției *tabsort*:

```
public static void main (String argv[]) {  
    Object a[] [] =  
    for ( int i = 0; i < 3; i++ ){  
        tabsort( a, i ); // ordonare matrice a dupa coloana i  
        print( a ); // afisare matrice a  
    }  
}
```