Poo Laboratoare[1]

Contents

[1] http://www.google.ro/

Laborator6

Programare Orientata pe Obiecte: Laborator 6

1 Clase abstracte si interfete

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 dupa 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 dupa primul obiect din pereche și altul ordonat dupa al doilea obiect.

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

2.3 Problema 3

(2 punctee) Să se definească o clasă filtru pentru selecție de fișiere dupa o listă de extensii (tipuri de fisiere). Clasa implementează una din interfațele *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 puncte) Functie statica pentru ordonarea unei matrice de obiecte "a" dupa valorile dintr-o coloana data "c" (ordonare naturala crescatoare):

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

Clasa comparator este o clasa inclusa anonima definita la apelul functiei "Arrays.sort", in cadrul functiei "tabsort". Exemplu de program pentru verificarea functiei tabsort:

public static void main (String av[]) {

```
Object a[][] =
for (int i=0;i<;i++){
   tabsort(a,i); // ordonare matrice a dupa coloana i
   print (a); // afisare matrice a
}
```

}