Poo Laboratoare[1]

Contents

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

Laborator4

Programare Orientata pe Obiecte: Laborator 4

1 Clase noi si clase derivate

```
class ListAsVector
{
   Vector v;
   public ListAsVector()
   {
      v = new Vector();
   }
   public boolean add(Object o)
   {
      return v.add(o);
   }
   ....
}

class ListAsVectorDerivat extends Vector
{
   public boolean add(Object o)
   {
      return super.add(o);
   }
...
}
```

1.1 Principii POO

1.1.1 Moştenirea

Permine definirea unor noi clase care păstrează caracteristicile, datele, funcţiile unei alte clase (denumită clasă de bază). Pornind de la această proprietate putem defini clase noi care extind comportamentul unei clase de baze, făcând dintr-o clasă generală, una particulară pentru ceea ce avem nevoie.

1.1.2 Supraîncărcarea

Oferă posibilitatea de a redefini metode din clasele de bază. Se observă astfel, cum am redefinit în clasa SetAsVectorDerivat metoda add, în schimb am putut utiliza metoda add din clasa de bază *Vector* datorită lui super (super.add(o) apelează metoda add din *Vector*).

1.1.3 Observaţii

Clasele Vector si Hashtable se găsesc în pachetul java.util. Pentru a vedea metodele acestor clase accesaţi documentaţia!

2 Probleme de laborator

2.1 Problema 1

(2 puncte) Să se definească o clasă *SetAsVector* pentru o mulțime de obiecte realizată ca vector neordonat cu elemente distincte, în două variante:

- clasa *SetAsVector* derivată din clasa *Vector* (pachetul java.util)
- clasa *SetAsVector* conține un obiect de tip *Vector*

Metode din clasa *Vector* care vor fi implementate în clasa *SetAsVector*:

- boolean add(Object)
- boolean remove(Object)
- boolean contains(Object)
- String toString()

Rezultatul metodelor *add* și *remove* este *true* dacă operația cerută a reusit.

Să se scrie un program pentru crearea unei mulțimi de șiruri prin adăugări succesive, eliminarea unor elemente și afișare dupa fiecare operație, folosind clasa *SetAsVector*.

2.2 Problema 2

(2 puncte) Să se definească o clasă *SortedVector* derivată din clasa *Vector* pentru vectori ordonați de obiecte, cu următoarele metode redefinite:

- addElement(Object) : adaugă un element la sfârșitul vectorului și apoi ordonează cu metoda Collections.sort
- insertElementAt(Object, int) : adaugă într-o poziție dată și reordonează
- removeElement(Object) : șterge elementul și reordonează

Să se scrie un program pentru operații cu un vector ordonat de șiruri: adăugări succesive de elemente (obiecte *Integer*) și ștergeri de elemente (removeElement) cu afișare dupa fiecare modificare.

2.3 Problema 3

(2 puncte) Să se definească o clasă *HSet* pentru o mulțime realizată ca tabel de dispersie, cu metodele: add, contains, toString, size. Clasa *HSet* va fi derivată din clasa *Hashtable* (pachetul java.util), în care cheia și valoarea asociată vor fi egale (cheile sunt elementele mulțimii).

Să se scrie un program pentru adăugarea unor numere la mulțime și afișarea mulțimii.

2.4 Problema 4

(2 puncte) Să se definească o clasă *SList* pentru liste simplu înlănțuite de obiecte de tip *Object*, cu adăugare la început de listă. Se va declara și o clasă *Node* ce conține un nod de listă (informație de tip Object și legătura la un alt nod).

Clasa *SList* va conține ca date: începutul listei (variabila de tip *Node*). Metode *SList*: add, del, contains, empty, toString.

Să se definească apoi o clasă derivată *SetAsList* pentru mulțimi de liste (se redefinește metoda de adaugare). Program pentru crearea și afișarea unei mulțimi a cuvintelor distincte dintr-un vector de șiruri.

2.5 Problema 5

(2 puncte) Să se definească o clasă *Graph* pentru grafuri reprezentate prin liste de adiacențe, ca o clasă derivată din clasa *Vector*. Listele de adiacențe sunt tot vectori (obiecte *Vector*). Nodurile grafului se numeroteaza de la 1. Se va defini un constructor cu argument număr de noduri din graf.

Metode:

- *size* : numărul de noduri din graf)
- *addArc* : adaugă un arc la graf
- *isArc* : verifică daca există arc intre doua noduri date
- *toString* : lista de arce din graf

Să se scrie un program pentru creare de graf prin adăugări succesive de arce (pe baza unor perechi de numere date in program), afișare arce si afișare grad interior și exterior pentru fiecare nod (număr de arce în și din nod).