Proiect – Partea a 3-a

Pentru partea a 3-a a proiectului va trebui să implementați în Java containerul/containerele folosite pentru partea a 2-a folosind o structură de date pe care o să o primiți după ce îmi trimiteți partea a 2-a. După ce implementați containerul va trebui să înlocuiți în codul de la partea a 2-a containerul gata implementat cu implementarea voastră (și proiectul va trebui să funcționeze în continuare).

Va trebui să implementați toate operațiile containerului chiar dacă nu aveți nevoie de ele în proiect. *Toate operațiile* înseamnă interfața containerului discutat la curs (lista de operații pe care o găsiți în cursul 4, 5 sau 6) nu toate operațiile care există în Java pentru containerul respectiv (Java în general are mult mai multe operații). Dacă ați folosit ceva operație care există în Java și noi nu am inclus-o în interfață, va trebui să implementați și operația respectivă.

În implementarea voastră nu aveți voie să vă folosiți de containere existente din Java (deci de exemplu, dacă aveți de implementat o mulțime pe vector dinamic, nu aveți voie să folosiți ArrayList sau Vector din Java, trebuie să lucrați cu vectorul simplu, fără a folosi câmpul *length)*. O singură excepție există, dacă aveți de implementat Dicționar sau MultiDicționar, acolo sunt anumite operații care returnează o Mulțime (operația mulțimea cheilor) sau o listă (căutarea la MultiDicționar). La acele operații puteți folosi Set/List din Java pentru a returna rezultatul, dar nu și pentru a reține elementele containerului.

Ideal ar fi să existe cât mai puține modificări în cod când treceți de la implementarea existentă pentru container la implementarea voastră. Un mod de a reduce numărul modificărilor este să denumiți operațiile voastre la fel cum se numesc în Java (de exemplu *add*, în loc de *adauga*) și să păstrați ordinea parametrilor.

Containerele din Java folosesc *generics* (adică parametrul de tip, adăugat între semne < >, de ex. Set<Integer>). Implementarea voastră nu trebuie să folosească generics, puteți implementa containerul direct cu tipul de care vor fi elementele când folosiți containerul. De exemplu, dacă veți avea nevoie de un vector cu elemente de tip *Coordonata*, puteți define direct *Coordonata elemente[]*). Dacă vreți să lucrați cu generics, adică să implementați un container pentru care puteți voi specifica tipul elementelor, trebuie să adăugați la definirea clasei tipul parametrului (parametrilor în cazul Dicționarului) și peste tot în cod când vă referiți la tipul datelor să folosiți acel parametru. De ex:

```
public class Multime<T> {
	private T elemente[];
	//…
}
```

Nota pentru partea a 3-a nu este influențată de faptul că folosiți sau nu generics. Alegeți varianta care vi se pare mai simplu.

Iteratorul din Java are doar 2 operații (3 cu constructor): *hasNext și next*. Varianta discutată la curs avea 3 operații: *valid, urmator, element*. Puteți implementa oricare dintre aceste variante.

Dacă aveți un container ordonat, trebuie să folosiți un Comparator (clasa Container trebuie să conțină un câmp/atribut de tip Comparator, care va fi folosit pentru a ordona elementele).

Nu trebuie să faceți citire din fișier. Dacă vreți, se poate, dar nu este necesar. Dacă nu vreți citire din fișier, dar nici nu vreți ca la fiecare rulare să trebuiască să adăugați o serie de elemente, doar să aveți informații cu care să lucrați, puteți face o funcție, care să adauge niște elemente în containerul vostru și să apelați funcția la începutul programului.