Examen L3 -P00 19/12/2012

Lisez bien l'énonce jusqu'à la fin. Vous pouvez répondre aux questions dans n'importe quel ordre en précisant bien leur numéro. Les barèmes sont indicatifs

Question 1- (5.5pts)

1. Définissez la classe *Inter* pour représenter un intervalle ouvert d'entiers avec des variables (attributs) privés: 1- *bInf*: borne inférieure, 2-*bSup* borne supérieure. (Les bornes n'appartiennent pas à l'intervalle). Donnez les méthodes suivantes pour cette classe

1. un constructeur qui prend en paramètre deux entiers $bInf, bSup$ et lance une exception si $bInf > bSup$.
2. une méthode qui retourne un booléen (*boolean contient(Inter i)*): si l'intervalle qui invoque cette méthode contient l'intervalle i passé en paramètre le résultat retourné est vrai (true) , sinon c'est faux (false)
3. une méthode statique qui retourne l'intervalle qui est l'intersection de deux intervalles passées en paramètre (*Inter intersection(Inter i1, Inter i2)*). La méthode retourne nulle si l'intersection est vide.

Question 2- (1.5pts)

Ecrivez la classe *InterF* qui est une classe dérivée de *Inter* pour représenter un intervalle d'entiers fermé (les bornes inférieure et supérieure sont dans l'intervalle). Dans cette classe, on ajoute une variable (attribut) statique booléenne *inclus* initialisée à vrai (true) pour indiquer que les bornes sont incluses dans l'intervalle.

1. Donnez un constructeur pour cette classe .
2. Modifiez la méthode *contient (InterF i)* pour un intervalle fermé

Question 3- (5pts)

Soit *InterL* une classe qui a un attribut privé *listeInter* : une liste où chaque élément est soit un objet *Inter* ou un objet *InterF* ($List < Inter > listeInter$). Donnez les méthodes suivantes pour cette classe:

1. une méthode qui retourne l'ensemble des intervalles qui apparaissent au moins 2 fois dans *listeInter*.

   $Set < Inter > repetitions()$.
2. une méthode qui retourne l'ensemble des intervalles de l'attribut *listeInter*

   $Set < Inter > sansrepetitions()$.
3. une méthode qui retourne un tableau associatif construit à partir de *listeInter* . Les clés sont les distances ($bSup - bInf$). La valeur pour une clé d est le nombre d'intervalles dans *listeInter* dont la distance est d.

   $Map < Integer, Integer > \ distances()$.

Question 4-(2 pt)

Ecrivez dans une classe générique ($< T >$) une méthode qui a deux paramètres : une liste d'objets $< T >$ et un objet de classe $<?\ extends\ T >$. La méthode retourne le nombre de répétions de l'objet dans la liste. Nous supposons que les méthodes *equals* sont définies dans les classes qui sont susceptibles d'appeler cette méthode.

$int\ nmbrepet\ (List < T >\ liste,\ <?\ extends\ T > objet)$

Question 5. (1.5pt)

Ecrivez le programme principal pour définir 2 intervalles ouverts: $i1 = (1, 7)$, $i2 = (2, 8)$ et un intervalle fermé $i3 = (0, 10)$. Appelez la méthode *intersection* avec paramètres $i1$ et $i2$. Soit g un objet de classe *interL*. Peut-on appeler la méthode *nmbrepet* avec paramètres: l'attribut *listeInter* de g et $i1$? Si oui, donnez l'instruction.

Question 6-(3pt)

Quels sont les résultats de la compilation et l'exécution de chaque instruction, si si

1-A a = new A(); a.f2();

2- A a =new B(); a.f1();

3- A a =new B(); a.f2();

4- A a =new B(); D.m(a); C.m(a);

5- B a =new B(); a.f1();

6- B a =new B(); a.f2();

7- B a =new B(); D.m(a); C.m(a);

```
 class A{
   public void f2() {System.out.println("appel de f2 de A");}
   public void f3() {System.out.println("appel de f3 de A");}
}
class B extends A {
public void f1() {System.out.println("appel de f1 de B");}
public void f2() {System.out.println("appel de f2 de B");}   }

class C{   public static void m(A a) {a.f1(); a.f2(); a.f3();}   }

class D{  public static void m(B a){a.f1(); a.f2(); a.f3();} }
```

Question 7-(1.5pt)

On considère les classes suivantes:

```
class C {
public int x;
public C (int i) {x=i;}
}
class A{
public void h(int i, C o1, C o2)   {i=5; o1.x=5; o2=o1; }
}
```

Qu'affiche le fragment de programme suivant.

```
A a= new A();
C objet1=new C(1);   C objet2=new C(3);   int j=2;
a.h(j,objet1,objet2);      System.out.println(j+ " "+objet1.x+" "+objet2.x) ;
```

MAP< K,V >: int size(); boolean isEmpty(); put(K key, V value) ; V get (K key); boolean containsKey(K key); boolean containsValue(V value)

Collection< E > : int size(); boolean isEmpty(); boolean contains(Object element); boolean add(E element); boolean remove(Object element);