Examen Partiel 2019

1 Classes

Considérer le programme suivant :

```
1  class A {
2          private int x=1;
3          static int y;
4          public A() { this(2); y++; }
5          public A(int x) { this.x = x; }
6          public A(int x, int y) { this.x = x; A.y = y; }
7          public void f() { System.out.println("A:f:"+x+","+y); inc(); }
8          private void inc() { x++; y++; }
9  }
10
11 class B{
12         int x=7;
13         public void f(A a){ a.f(); }
14         public void g(B b){ b.f(); }
15         public static void main(String [] args){
16                 System.out.println(x++);
17                 A a1 = new A();
18                 a1.f();
19                 A a2 = new A(2,3);
20                 a1.f();
21                 a2.f();
22                 B b1 = new A(2);
23                 B b2 = new B();
24                 f(a1);
25                 b2.f(a2);
26         }
27 }
```

1. Trouver toutes les erreurs de compilation eventuelles de ce programme. Proposer des corrections.
2. Si on supprime toutes les lignes qui contiennent une erreur de compilation, que va le programme afficher ?

2 Héritage

Considérer le programme suivant :

```
1  class A {
2          private int x;
3          static int y;
4          public A() { this(2); y++; System.out.println("A()"); }
5          public A(int x) { this.x = x; }
6          A copy() { return new A(x); }
7          public String toString() { return "A:"+x+","+y; }
8          void f(B b){ System.out.println("AfB:"+ b); }
9  }
10 class B extends A{
11         int z;
12         public B() { z=7; }
13         public B(int z) { super(); this.z = z; }
14         public B copy() { return new B(z); }
15         public String toString() { return "B:"+x+","+y+","+z; }
16         void f(A a){ System.out.println("BfA:"+ a); }
17 }
18 class C extends B{
19         int x;
20         public C(int z, int x) { super(z); this.x = x; }
21         C copy() { return new C(z,x); }
22         public String toString() { return "C:"+x+","+y+","+z; }
23         void f(A a){ System.out.println("CfA:"+ a); }
24         void f(B b){ System.out.println("CfB:"+ b); }
25
26         public static void main(String [] args){
27                 A ac = new C(2,3);
28                 B bb = new B();
29                 A ab = new B(7);
30                 C cc = new C(4,5);
31                 B ba = new A();
32                 System.out.println(ac);
33                 System.out.println(bb);
34                 System.out.println(ab.copy());
35                 ac.f(ac);
36                 ac.f(bb); bb.f(ac); cc.f(ab);
37                 (new C(7,8)).f(cc);
38         }
39 }
```

1. Trouver toutes les erreurs de compilation eventuelles de ce programme. Proposer des corrections.
2. Si on supprime toutes les lignes qui contiennent une erreur de compilation, que va le programme afficher ?

3 Interfaces

Rappel : on a défini les classes : `Rational`, qui représente des nombres rationnels en utilisant deux entiers (numérateur, dénominateur) ; `Time`, qui représente l'heure actuelle en utilisant deux entiers (heure, minutes) ; et `Money` qui représente une somme en euros, centimes.

```
1  class Rational {
2          int p,q=1;
3          /*..*/
4  }
5  class Time {
6          int hours, minutes;
7          /*..*/
8  }
9  class Money {
10         int euros, centimes;
11         /*..*/
12 }
```

On définit maintenant l'interface `Testable` comme suit :

```
1  interface Testable {
2          boolean isOK();
3          void reset();
4  }
```

L'idée est que cette interface permet à l'utilisateur d'un objet de vérifier si l'état interne de l'objet est incohérent. Par exemple, un rationnel dont le dénominateur est zéro, une heure dont le nombre de minutes est > 59, ou une somme dont le nombre de centimes est > 99 ou < 0, représentent des objets avec un état erroné. La fonction `isOK` retourne `true` si l'état de l'objet paraît cohérent (aucun problème évident n'est constaté), sinon elle retourne `false`. La fonction `reset` ré-initialise l'objet avec des valeurs par défaut.

Modifier les définitions des trois classes pour que le programme suivant compile. Vous pouvez supposer qu'une implémentation des trois classes existe déjà et vous devez simplement ajouter ce qu'il manque pour que les classes soient compatibles avec l'interface `Testable`. Le but est que la fonction `isOK` détecte au moins toutes les erreurs décrites ci-dessus.

```
1  class Test {
2          public static void main(String [] args){
3                  Testable [] s = new Testable[3];
4                  s[0] = new Rational();
5                  s[1] = new Time();
6                  s[2] = new Money();
7                  for(Testable i:s){ System.out.println(i.isOK()); i.reset(); }
8          }
9  }
```

4 Files de priorité

Considérer le programme partiel suivant :

```
1  abstract class PriorityQ {
2          abstract void push(int x);
3          abstract int popLargest();
4          abstract int [] toArray();
5  }
6
7  class LinkedList {
8          int data;
9          LinkedList next;
10 }
11
12 class ListPQ extends PriorityQ {
13         LinkedList head;
14         /* .. */
15 }
```

La classe abstraite `PriorityQ` définit une file de priorité : La méthode `push` permet d'ajouter un entier. La méthode `popLargest` retire l'élément de valeur maximum (attn : cette méthode modifie l'état de la structure). La méthode `toArray` retourne un tableau avec tous les éléments de la file, trié en ordre décroissant.

Donner une implémentation concréte de cette classe, en utilisant des listes chaînées. Pour vous aider, on a déjà défini une classe auxillaire (`LinkedList`) représentant un noeud d'une telle liste. On a aussi commencé la définition d'une classe `ListPQ`.

1. Ajouter des implémentations pour les méthodes abstraites.
2. Ajouter une redéfinition de la méthode `toString`, et si necessaire, des constructeurs.

Pour tester votre implémentation, considérez le programme suivant :

```
1          PriorityQ f = new ListPQ();
2          for(int i=0;i<4;i++) { f.push(i); System.out.println(f); }
3          for(int i=0;i<4;i++) { f.push(2*i); System.out.println(f); }
4          for(int i=0;i<2;i++) { f.popLargest(); System.out.println(f); }
```

Ce programme doit fonctionner correctement et afficher :

```
0,
1, 0,
2, 1, 0,
3, 2, 1, 0,
3, 2, 1, 0, 0,
3, 2, 2, 1, 0, 0,
4, 3, 2, 2, 1, 0, 0,
6, 4, 3, 2, 2, 1, 0, 0,
4, 3, 2, 2, 1, 0, 0,
3, 2, 2, 1, 0, 0,
```