Année 2018-2019

Feuille 2 - Objets

1 Classes

Considérez le programme suivant :

```
1  class Test{
2          private int a;
3          int b;
4          static int c;
5          Test() { this(3); }
6          Test(int a, int b){ c=a=b; }
7          Test(int a){ this.a=a; c++;}
8          int test(){ b++; c++; return a+b+c; }
9          static int test2() { return a+c; }
10         public static void main(String [] args){
11                 Test t1 = new Test();
12                 t1.test();
13                 System.out.println(test());
14                 System.out.println(t1.a);
15                 System.out.println(t1.b);
16                 System.out.println(t1.c);
17                 Test t2 = new Test(2,3);
18                 System.out.println(t2.test());
19         }
20 }
```

1. Trouver et corriger toutes les erreurs de compilation.
2. Une fois corrigé, qu'affiche ce programme ?

2 La classe Complex

Pour cet exercice on va programmer une classe pour représenter les nombres complexes. Rappel : un nombre complexe est un nombre de la forme $z = a + bi$, où a, b sont des réels, et i est la racine de -1. Votre classe doit contenir au moins :

1. Deux champs `double a,b`, qui sont privés et immuables. Pourquoi ?
2. Deux méthodes "getters" qui donnent accès à `a,b`.
3. Une méthode `add` qui fait l'addition de deux complexes.
4. Une méthode `mult` qui fait la multiplication de deux complexes.
5. Une méthode `mult` qui fait la multiplication d'un complexe avec un double.

6. Une méthode `toString`.

Attn : il faut respecter les principes de la programmation par objet : minimiser la duplication de code, donner le minimum possible d'accès dans la classe.

3 La classe Money

Pour cet exercice vous devez définir une classe qui peut représenter une somme en euros et centimes. Donner la signature de la classe qui doit contenir au moins :

1. Deux constructeurs, un par défaut, et un qui prend comme paramètres le nombre d'euros et centimes. Ex : pour représenter la somme $3,50$ on appelle `new Money(3,50)`.
2. Une méthode `toString()` qui retourne un `String` qui représente la somme. Attn : cette méthode doit marcher correctement dans tous les cas.
3. Une méthode `isZero()` qui vérifie si la somme est égale à 0.
4. Une méthode `Money add(Money)` qui ajoute une autre somme à cette somme et retourne un nouveau objet qui représente le résultat.
5. Une méthode `Money add(int)` qui prend comme paramètre un `int x`, et ajoute à la somme actuelle `x` euros.
6. Une méthode `boolean isMoreThan(Money)` qui fait la comparaison entre deux sommes.

Attn : il faut respecter les principes de la programmation par objet : minimiser la duplication de code, donner le minimum possible d'accès dans la classe.