INFO-H-301 : Programmation orientée objet
TP 4 - Interfaces, classes abstraites, héritage

Professeur : Hugues Bersini
http://cs.ulb.ac.be/public/teaching/infoh301

Année académique 2011-2012

Exercice 4.1 [Livre de cours exercice 12.4]

Que donne la compilation de ces fichiers (sur papier, sans l'exécuter) ? Si cela compile, que donne l'exécution du programme Java suivant ?

```
// fichier A.java
public class A {
   public A() {}
}

// fichier B.java
public class B extends A {
   public B() {
      super();
   }
   public String toString() {
      return(" Hello " + super.toString());
   }
}

// fichier testAB.java
public class TestAB {
   public TestAB() {
      A a = new B();
      System.out.println(a);
   }
   public static void main(String[] args) {
      TestAB tAB = new TestAB();
   }
}
```

Exercice 4.2 [Livre de cours exercice 12.5]

Sur papier, sans exécuter du code, supprimez dans le code qui suit les lignes qui provoquent une erreur et indiquez si l'erreur se produit à la compilation ou à l'exécution. Quel est le résultat de l'exécution qui s'affiche à l'écran après suppression des instructions à problème ?

```
class A {
   public void a() {
      System.out.println("a de A") ;
   }
   public void b() {
      System.out.println("b de A") ;
   }
}

class B extends A {
   public void b() {
      System.out.println("b de B") ;
   }
   public void c() {
      System.out.println("c de B") ;
   }
}

public class Correction2 {
   public static void main(String[] args) {
      A a1=new A();
      A b1=new B();
      B a2=new A();
      B b2=new B();
      a1.a() ;
      b1.a() ;
      a2.a() ;
      b2.a() ;
      a1.b() ;
      b1.b() ;
      a2.b() ;
      b2.b() ;
      a1.c() ;
      b1.c() ;
      a2.c() ;
      b2.c() ;
      ((B)a1).c() ;
      ((B)b1).c() ;
      ((B)a2).c() ;
      ((B)b2).c() ;
   }
}
```

Exercice 4.3

Implémenter les classes A, B et C telles que

- A est super classe de B qui est super classe de C.
- Le constructeur de la classe x imprime "constructeur de la classe x".
- Redéfinir la méthode `protected void finalize()` dans chacune des classes pour qu'elle imprime "destructeur de la classe x".
- Dans la fonction `main` construire un objet de la classe C, mettre sa référence à `null` et forcer le Garbage Collector avec la commande `System.gc()`.
- Que faire pour que le destructeur de chaque super classe soit également appelé ?

Exercice 4.4

Reprendre le code *Illustrator* du TP précédent et effectuer les modifications suivantes :

- Adapter la classe `Form` pour la rendre abstraite. Rendez abstraites les méthodes nécessaires.
- Ajouter une interface `Drawable` déclarant la méthode `isOn(Point p)` et `getColor()`, que va implémenter `Form`. Ajouter une classe `AsciiNumber` implémentant `Drawable` et représentant un chiffre de 0 à 9 se dessinant sur une aire de 3 sur 4. Faites toutes les modifications nécessaires permettant d'avoir dans vos Illustrations des `AsciiNumber` et des `Form`.

Exercice 4.5

- Ajouter une classe `Application` qui reçoit et parse des commandes (via `System.in`, la lecture au clavier en console), celles-ci permettent de créer des formes, de les supprimer et de les déplacer. Par exemple : `create circle 10 10 c 15` crée un cercle de centre $(10, 10)$, représenté par un `'c'` et de rayon 15.
- Ajouter un mécanisme pour que votre `Application` retienne toutes les commandes reçues et puisse les enregistrer dans un fichier.
- Ajouter un mécanisme permettant à votre `Application` d'appliquer les commandes lues dans un fichier. Ajouter éventuellement une fonction `wait(millisecond)` permettant ainsi de créer des *animations* console à partir d'un fichier.

Annexe : Lire dans la console

```
import java.util.Scanner;

...

Scanner in = new Scanner(System.in);
String command = in.nextLine();       //lit une ligne sur la console
String[] array = command.split(" ");  //découpe la ligne

for(String s : array)
{
System.out.println(s);
}
```