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

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

Année académique 2011-2012

Correction 4.1 [Livre de cours exercice 12.4]

```java
// 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();
   }
}
```

Ce programme affiche

```
Hello B@199a0c7c
```

- La première instruction qui s'exécute est `TestAB tAB = new TestAB();` (23) qui crée un objet de type `TestAB` en utilisant le constructeur par défaut.
- Ce constructeur par défaut (18) crée un nouvel objet de type `B` référencé par la variable `a` de type `A` (19). Cette opération est permise car `B` hérite de `A` et donc "`B` est un `A`".
- Ensuite, l'instruction `System.out.println(a);` (20) est appelée. Quand `println` est appelé sur un objet, il affiche le résultat de l'envoi du message `toString` sur cet objet. La méthode `toString` de la classe `B` est donc appelée (11). Celle ci renvoie la chaîne `" Hello "` concaténée au résultat de la méthode `toString` supérieure la plus proche dans la hiérarchie. Ici, la méthode appelée est celle de la classe `Object` qui affiche le type de l'objet (`B`) ainsi que son adresse (`199a0c7c`).

Correction 4.2 [Livre de cours exercice 12.5]

```java
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();    // OK
      A b1=new B();    // OK car B est un A
      B a2=new A();    // NOK (compilation) car A n'est pas un B !
      B b2=new B();    // OK
      a1.a() ;         // affiche ``a de A''
      b1.a() ;         // affiche ``a de A'' puisque B n'a pas de methode a()
      a2.a() ;         // NOK (ligne 23)
      b2.a() ;         // affiche ``a de A'' puisque B n'a pas de methode a()
      a1.b() ;         // affiche ``b de A''
      b1.b() ;         // affiche ``b de B'' (un message est toujours recu par l'objet instancie)
      a2.b() ;         // NOK (ligne 23)
      b2.b() ;         // affiche ``b de B''
      a1.c() ;         // NOK (compilation) car la classe A n'a pas de methode c()
      b1.c() ;         // NOK (compilation) car la classe A n'a pas de methode c()
      a2.c() ;         // NOK (ligne 23)
      b2.c() ;         // affiche ``c de B''
      ((B)a1).c() ;    // NOK (execution) : un objet de type A ne peut pas repondre au message c()
      ((B)b1).c() ;    // OK : l'objet de type B recoit le message c() et affiche ``c de B''
      ((B)a2).c() ;    // NOK (ligne 23)
      ((B)b2).c() ;    // OK : affiche ``c de B''
   }
}
```

Correction 4.3

```
1  class A {
2     public A() {
3        System.out.println("Constructeur de la classe A");
4     }
5     protected void finalize() throws Throwable{
6        System.out.println("Finalisation de la classe A");
7        super.finalize();
8     }
9  }
10
11 class B extends A {
12    public B() {
13       System.out.println("Constructeur de la classe B");
14    }
15    protected void finalize() throws Throwable{
16       System.out.println("Finalisation de la classe B");
17       super.finalize();
18    }
19 }
20
21 class C extends B {
22    public C() {
23       System.out.println("Constructeur de la classe C");
24    }
25    protected void finalize() throws Throwable{
26       System.out.println("Finalisation de la classe C");
27       super.finalize();
28    }
29 }
30
31 public class Program {
32    public static void main(String[] args) {
33       C c = new C();
34       c = null;
35       System.gc();   //appel explicite du Garbage Collector
36    }
37 }
```

Correction 4.4

Voir code téléchargeable sur la page web des TPs.