Exercice 1. Classes et visibilité

Q. Parmi les appels suivant indiquez ceux qui sont corrects et ceux qui ne le sont pas

```
public class A {
      public int i;
      private int a;
      protected int b;
      public void afficher() { System.out.println(i+a+b); }
}
public  class B extends A {
      public void afficher() { System.out.println(i+a+b); }
}
public class D {
      Public A a = new A();
      Public void afficher() { System.out.println(a.i + a.a + a.b); }
}
public class E {
      Public B b = new B();
      Public void afficher() { b.afficher();}
}
public class F {
      Public B b = new B();
      Public void afficher() { System.out.println(b.i + b.a + b.b) }
}
```

Exercice 2 - Classes abtraites

La classe *Vector* (package java.util) est couramment utilisée. Nous allons ici l'utiliser de deux façons pour créer une FIFO (First In / First Out) et une LIFO (Last In / First Out).

- Créer la classe abstraite *Stack* qui contient les classes abstraites:

  Object get();
  void put(Object);

- Implémenter cette classe de deux façons: par une classe *FIFOStack* puis par une classe *LIFOStack* selon le comportement de pile désiré. (Utiliser les méthodes *addElement()* et *removeElement()* de la classe *Vector*).

Solution

Créer la classe abstraite Stack qui contient les classes abstraites:
Object get();
void put(Object);
Classe Stack:

```
abstract class Stack{
  Vector v=new Vector();
  abstract Object get();
  abstract void put(Object o);
}
```

Implémenter cette classe de deux façons: par une classe FIFOStack puis par une classe LIFOStack selon le comportement de pile désiré

```
class FIFOStack extends Stack{
  Object get(){
    if ( v.size()>0 ){
      Object o=v.elementAt(0);
      v.removeElementAt(0);
      return o;
    }
    else{
      return null;
    }
  }
  void put(Object o){
    v.addElement(o);
  }
}
```

```
class LIFOStack extends Stack{
  Object get(){
    if ( v.size()>0 ){
      Object o=v.elementAt(v.size()-1);
      v.removeElementAt(v.size()-1);
      return o;
    }
    else{
      return null;
    }
  }
  void put(Object o){
    v.addElement(o);
  }
}
```

Exercice 3 - Interfaces

- Transformez la classe *Stack* pour qu'elle devienne une interface qui est implémentée par les classes *FIFOStack* et *LIFOStack*.
- De quels type(s) sont les instances de *FIFOStack* et *LIFOStack* ?

Solution

Interface Stack:

```
interface Stack{
  static final Vector v=new Vector();
  Object get();
  void put(Object o);
}
```

```
class FIFOStack implements Stack{
  public Object get(){
    if ( v.size()>0 ){
      Object o=v.elementAt(0);
      v.removeElementAt(0);
      return o;
    }
    else{
      return null;
    }
  }
  public void put(Object o){
    v.addElement(o);
  }
}
```

```
class LIFOStack implements Stack{
  public Object get(){
    if ( v.size()>0 ){
      Object o=v.elementAt(v.size()-1);
      v.removeElementAt(v.size()-1);
      return o;
    }
    else{
      return null;
    }
  }
  public void put(Object o){
    v.addElement(o);
  }
}
```

Remarque: Les méthodes *get()* et *put()* doivent être déclarées *public* pour éviter que leurs privilèges soit moindres que ceux de l'interface *Stack* dont les méthodes sont *public* par défaut.

- De quels type(s) sont les instances de FIFOStack et LIFOStack ?

Une instance de *FIFOStack* est de type: *FIFOStack*, *Object* et *Stack*.