Interfaces

Une interface ne contient que des constantes de classe et des méthodes publiques abstraites (mot-clé **interface**)

Une interface est un **modèle d'implémentation**. Une classe non abstraite qui implémente une interface (mot-clé **implements**) définit toutes les méthodes déclarées dans celle-ci.

Un nom d''interface peut être utilisé comme un nom de classe et est donc un **type** à part entière

Ainsi on utilise souvent les interfaces dans les types de paramètres. En effet, dans certains cas, peu importe la classe d'un objet donné, car cela ne nous intéresse pas. Nous avons seulement besoin de savoir que telle ou telle classe possède ces méthodes et se comporte de telle manière.

Une variable dont le type est une interface peut avoir comme valeur n'importe quel objet implémentant l'interface en question. Quelque soit l'objet référencé par la variable, on pourra lui appliquer une méthode déclarée dans l'interface.

L'héritage multiple d'interfaces est possible :

```
interface Dessinable extends Coloriable, Tracable
{
        ...
}
```

Exemple Broadcast (fichier TestBroadcast.java)

Cet exemple illustre la capacité de Java à « broadcaster » de manière concise des messages à des objets de différentes classes. La réalisation passe par une interface que chaque classe voulant être broadcastée doit implémenter. Les interfaces remplacent agréablement les pointeurs de fonctions du langage C.

```
package broadcast;
public class TestBroadcast {
        public static void main(String args[]) {
           Broadcaster b = new Broadcaster();
           b.addItem(new ItemBox());
           b.addItem(new ItemCircle());
           b.drawAllItems();
        }
}
```

```
package broadcast;
import java.util.List;
import java.util.ArrayList;

public class Broadcaster {
   private List<VisualItem> visualItems = new ArrayList<VisualItem> ();

   public void addItem(VisualItem visualItem) {
     visualItems add(visualItem);
   }

   public void drawAllItems() {
      for (VisualItem visualItem : visualItems) {
                        visualItem.draw();
        }
   }
}
```

```
package broadcast;
interface VisualItem
{
        abstract public void Draw();
}
```

```
package broadcast;
public class ItemBox implements VisualItem
{
        public void Draw() {        System.out.println("Draw a box");    }
}
```

```
package broadcast;
public class ItemCircle implements VisualItem
{
        public void Draw() {        System.out.println("Draw a circle");    }
}
```
