

# Conception et interfaces

## Programmation Orientée Objet

Jean-Christophe Routier  
Licence mention Informatique  
Université Lille 1



Université  
Lille1  
Sciences et Technologies

IEEA - FIL  
Informatique

# Le problème

On s'intéresse à la modélisation d'un bricoleur qui peut effectuer certaines tâches telles que visser, couper, casser. Chacune de ces tâches s'accomplit à l'aide d'un outil adapté.

Par exemple, un tournevis est un outil adapté pour visser, on pourrait donc avoir quelque chose ressemblant à :

```
public class Bricoleur {  
    public void visse(Tournevis t) {  
        t.visse();  
    }  
    ...  
}  
  
public class Tournevis {  
    public void visse() {  
        System.out.println("Tournevis visse");  
    }  
}
```

```
public class Tournevis {
    public void visse() {
        S.o.p("T visse");
    }
}
```

```
public class Marteau {
    public void casse() {
        S.o.p("Marteau casse");
    }
}
```

```
public class Scie {
    public void coupe() {
        S.o.p("Scie coupe");
    }
}
```

```
public class Bricoleur {
    public void visse(Tournevis t) {
        t.visse();
    }
    public void casse(Marteau m) {
        m.casse();
    }
    public void coupe(Scie s) {
        s.coupe();
    }
    ...
}
```

Prise en compte d'un cutter ? d'une masse ?

```
public class Masse {
    public void casse() {
        S.o.p("Masse casse");
    }
}
```

```
public class Cutter {
    public void coupe() {
        S.o.p("Cutter coupe");
    }
}
```

```
public class Bricoleur {
    public void visse(Tournevis t) {
        t.visse();
    }
    public void casse(Marteau m) {
        m.casse();
    }
    public void coupe(Scie s) {
        s.coupe();
    }
    public void coupe(Cutter c) {
        c.coupe();
    }
    public void casse(Masse m) {
        m.casse();
    }
    ...
}
```

# NON !

pas de généralisation possible,  
on est obligé de toucher au  
code de Bricoleur pour  
ajouter un nouvel outil

- ▶ Définir une **interface** pour les outils sachant couper, visser, casser

**abstraction** de ces notions

```
public interface PeutVisser {  
    public void visse();  
}
```

```
public interface PeutCasser {  
    public void casse();  
}
```

```
public interface PeutCouper {  
    public void coupe();  
}
```

# puis...

```
public class Tournevis implements PeutVisser {  
    public void visse() {  
        S.o.p("Tournevis visse");  
    }  
}
```

```
public class Marteau implements PeutCasser {  
    public void casse() {  
        S.o.p("Marteau casse");  
    }  
}
```

```
public class Scie implements PeutCouper {  
    public void coupe() {  
        S.o.p("Scie coupe");  
    }  
}
```

```
public class Masse implements PeutCasser {  
    public void casse() {  
        S.o.p("Masse casse");  
    }  
}
```

```
public class Cutter implements PeutCouper {  
    public void coupe() {  
        S.o.p("Cutter coupe");  
    }  
}
```

# et donc

```
public class Bricoleur {
    public void visse(PeutVisser visseur) {
        visseur.visse();
    }
    public void casse(PeutCasser cassant) {
        cassant.casse();
    }
    public void coupe(PeutCouper coupant) {
        coupant.coupe();
    }
    ...
}
```

```
Bricoleur bob = new Bricoleur();
bob.coupe(new Scie());
bob.casse(new Masse());
bob.coupe(new Cutter());
```

```
+--trace-----
+ Scie coupe
+ Masse casse
+ Cutter coupe
+-----
```

# Multi-Implémentation

```
public class CouteauSuisse implements PeutCouper, PeutVisser, PeutCasser {
    public void coupe() {
        S.o.p("CouteauSuisse coupe");
    }
    public void visse() {
        S.o.p("CouteauSuisse visse");
    }
    public void casse() {
        S.o.p("CouteauSuisse casse");
    }
}
```

```
Bricoleur bob = new Bricoleur();
CouteauSuisse mcGyver = new CouteauSuisse();
bob.coupe(mcGyver);
bob.casse(mcGyver);
bob.visse(mcGyver);
```

```
+--trace-----
+ CouteauSuisse coupe
+ CouteauSuisse casse
+ CouteauSuisse visse
+-----
```

```
CouteauSuisse mcGyver = new CouteauSuisse();
PeutCouper coupant = mcGyver; // Upcast de CouteauSuisse → PeutCouper
coupant.coupe(); // pas de pb
mcGyver.casse(); // pas de pb
coupant.casse(); // !!! INTERDIT !!! (détecté à la compilation)
((CouteauSuisse) coupant).casse(); // ok : Downcast de PeutCouper → CouteauSuisse
((Marteau) coupant).casse(); // compile mais Downcast illicite de Marteau → CouteauSuisse
```



# Interface de typage

- ▶ On veut pouvoir ranger les différents outils dans une boîte à outils représentée par un tableau.
- ▶ **Solution** : avoir une interface `Tool` qui sert uniquement à repérer les outils (typer)

```
public interface Tool { }

public class Scie implements PeutCouper, Tool { ...}
public class Marteau implements PeutCasser, Tool { ...}

Tool[] Toolbox = new Tool[5];
ToolBox[0] = new Scie();
ToolBox[1] = new Marteau();
...
```