Cours 8 : Héritage

On peut définir une classe héritant d'une autre classe

- La classe héritante :
  - Dérivation : récupère tous les comportements (accessibles) de la classe dont elle hérite
  - Extension : Peut ajouter de nouveaux comportements qui lui sont propres
  - Spécialisation : Peut modifier certains comportements hérités

Les instances de la classe héritante sont également du type de la classe héritée : Polymorphisme

On parle de sous-classe (sous-types)

En Java, pour indiquier qu'une classe hérite d'une autre, on utilise le mot clé `extends`

```java
public class Parcel {}
public class LightParcel extends Parcel {}
public class ExpressParcel extends Parcel {}
public class InsuredParcel extends Parcel {}
public class FragileParce extends InsuredParcel {}
```

On n'hérite de qu'une seule classe en java :(

> **Example**
>
> Petit exemple de code licite grâce à l'héritage

```java
Parcel p;
LightParcel lightP = new LightParcel( ... );

p = lightP;

P.deliver();
lightP.deliver();
```

Quand une classe hérite d'une autre en java, les attributs privés hérités ne sont pas accessibles :(, mais il possède ces attributs tout de même. Possible de changer cela…

> **Note**
>
> La relation d'héritage est transitive

Modification de comportement

On ne peut pas hériter de certaines méthodes uniquement. Pour redéfinir le comportement de certaines méthodes, il faut utiliser le décorateur `Override`

> **Warning**
>
> Pour modifier une méthode, il faut scrupuleusement respecter la signature, sinon on fait une extension

---

Toutes les classes héritent de Object (implicitement ou par transitivité)

Constructeur

Pour gérer les attributs hérités, il faut utiliser le super-constructeur, comme suit :

```java
public class Parcel {
    private int weight;
    private Person recipient
    private boolean delivered;

    public Parcel(int weight, Person p) {
        this.weight = weight;
        this.recipient = p;
        this.delivered = false;
    }
}

public class InsuredParcel extends Parcel {
    private float insuredValue;

    public InsuredParcel(int weight, Person recipient, float value) {
        super(weight, recipient);
        this.insuredValue = value;
    }
}
```

Exceptions

On peut créer des exceptions comme suit :

```java
public class CapacityExceededException extends Exception {
    public CapacityExceededException(String msg) {
        super(msg);
    }
}
```

> ⓘ Note
>
> Les exceptions qui héritent de `RuntimeException` n'ont pas besoin d'être capturées

Lookup

Lorsque l'on appelle une méthode héritée, que se passe-t-il ?

La recherche de la méthode est dynamique :

- La recherche commence dans la classe de l'objet invoquant late-binding
- Si aucune définition de la méthode est trouvée, on refait cela dans la super-classe