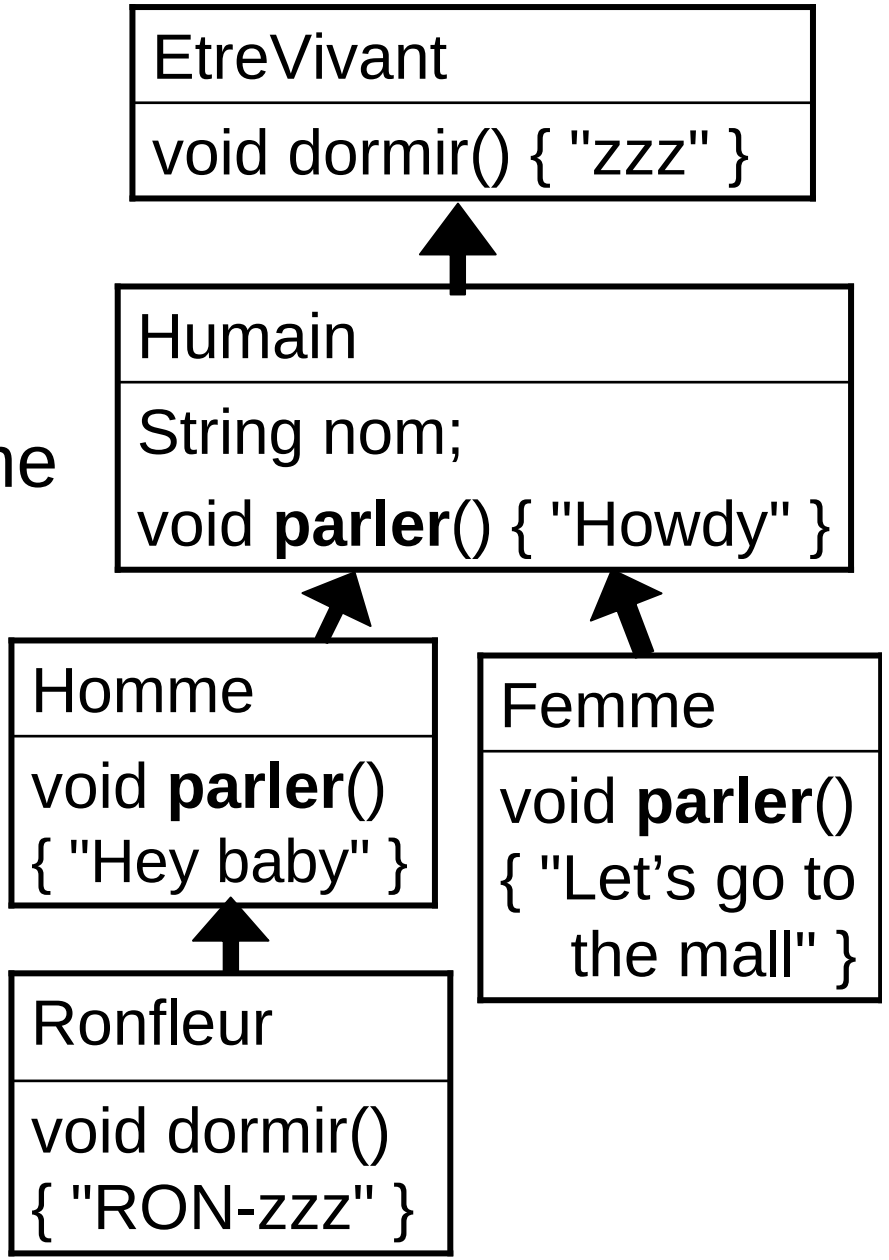
Héritage

- Une classe plus spécia-lisée peut redéfinir des méthodes de sa classe parente
  - Elles doivent avoir la même signature (arguments, type de retour)
- Java « détecte » le type de l'objet et appelle la version de la méthode la plus spécialisée possible



Héritage

```
class Humain{
    String name;
    void parler(){ S.o.p("Howdy! I'm " + name + ".");
}
class Homme extends Humain{
    void parler(){ S.o.p("Hey baby!"); }
}
class Femme extends Humain{
    void parler(){ super.parler();
                   S.o.p("Let's go to the mall!"); }
}
```

- C’est le type réel de la variable qui compte

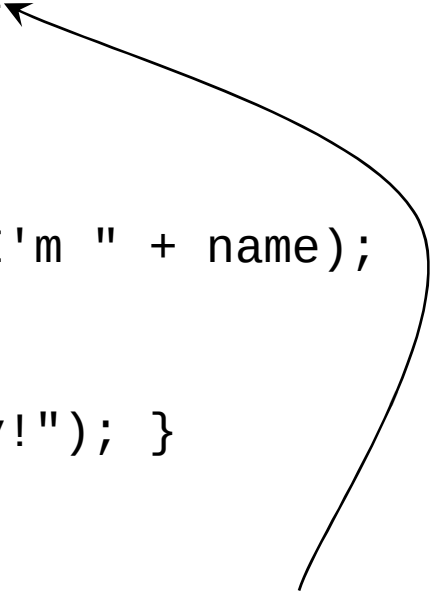
```
Humain h = new Homme("Gaston");    h.parler(); // Hey baby!
```

- super ≈ this avec le type parent (ignore les redéfinitions de méthodes de cette classe)

```
Femme f = new Femme("Robin");
f.parler(); // Howdy! I’m Robin. Let's go to the mall!
```

Héritage

```
class EtreVivant{
    void dormir(){ S.o.p("zzz"); }
}
class Humain extends EtreVivant{
    String name;
    void parler(){ S.o.p("Howdy! I'm " + name);
}
class Homme extends Humain{
    void parler(){ S.o.p("Hey baby!"); }
}
class Ronfleur extends Homme{
    void dormir(){S.o.p("RONron"); super.dormir(); }
}
```



- Ce n’est pas limité au parent immédiat
- Utilisable partout (≠ constructeur 1e ligne uniquement)

```
Humain h = new Ronfleur("Robert");
h.parler(); // Hey baby!
H.dormir(); // RONron zzz
```

Héritage

- Tous les objets, en Java, héritent (automatiquement) de la classe prédéfinie `Object`
- Vous pouvez redéfinir ses méthodes
  - `protected void finalize()` : destructeur
  - `public String toString()` : ce qui s'affiche quand vous printez l'objet (défaut "Type@0121545")
  - `clone`, `equals`,... voir javadoc
- `getClass()` (méthode `final`, impossible à redéfinir)

```
class Humain {
     String name;
     public String toString() { return "Humain " + name; }
}
System.out.println("H = X" + h + "Y"); // H = XHumain TotoY
```