Programmation Orienté Objets en JAVA

Daniele Varacca
Departement d'Informatique
Université Paris Est

2014

UPEC
UNIVERSITÉ PARIS-EST CRÉTEIL VAL DE MARNE
Connaissance - Action

Constructeur dans les sous-classes

- le mot clé super
- le constructeur par défaut

Projet

Projet

```
Well, Twilight Sparkle the Unicorn, you've got a long journey ahead.

What would you like to do? -
1. Travel to the Everfree Forest
2. Head to the shops
3. Quit - 1

     Carefully picking your way through the Everfree Forest, full of fauna and
creeping plants, you failed to notice the rustling in the bushes ahead. Danger
is upon you! A Griffon appears!
It moves out into full view, blocking your way along the path . . .

You're using your Hooves (1d20 dmg)

You also have your Horn (1d12 dmg)

Would you like to to:
1. Attack
2. Block
3. Feint
4. Other
5. Switch your method of combat - 5

You're now using your Horn (1d12 dmg)

You also have your Hooves (1d20 dmg)

Would you like to to:
1. Attack
2. Block
3. Feint
4. Other
5. Switch your method of combat - 1
```

UPEC Connaissance-Action
UNIVERSITÉ PARIS-EST CRÉTEIL VAL DE MARNE

Exemple simplifié

```
abstract class Character {
   ...
   int life;
   int ability;
   abstract void attack (Character defender) ;
   abstract void defend ();
   abstract String greeting();
}
```

UPEC
UNIVERSITÉ PARIS-EST CRÉTEIL VAL DE MARNE
Connaissance - Action

Exemple simplifié

```
abstract class FriendlyCharacter extends Character
   ...
   String greeting() {
        return "Hello␣I'm␣your␣friend";
    }
}

class Monk extends FriendlyCharacter {
   ...
   void attack (Character defender) {
      defender.life = defender.life - this.ability;
    }
   void defend () {
      this.life = this.life + 1;
    }
}
```

UPEC Connaissance-Action | UNIVERSITÉ PARIS-EST CRÉTEIL VAL DE MARNE

Un autre personnage

Je veux créer un autre type de moine, le saint.

```
class Saint extends FriendlyCharacter {
   ...
   void attack (Character defender) {
      defender.life = defender.life - this.ability;
    }
   void defend () {
      this.life = this.life + 2;
    }
}
```

- pas de relation de sous-typage
- la méthode attack est la même

Héritage

Je peux aussi créer une sous-classe d'une classe concrète!

```
class Saint extends Monk {
   ...
   void defend () {
      this.life = this.life + 2;
    }
}
```

La méthode defend est *rédefinie*

Héritage

Je peux aussi rédefinir d'autres méthodes

```
class Saint extends Monk {
   ...
   void defend () {
      this.life = this.life + 2;
    }
   String greeting() {
       return "Hello␣I'm␣your␣friend␣" +
              "and␣I␣will␣save␣your␣soul";
   }
}
```

Héritage

Si ce que je veux c'est juste *modifier* un comportement.

```
class Saint extends Monk {
   ...
   void defend () {
      super.defend()
      this.life = this.life + 1;
    }
   String greeting() {
       return (super.greeting() +
                 "␣and␣I␣will␣save␣your␣soul");
   }
}
```

Une autre utilisation du mot clé super

Héritage

Je peux aussi ajouter des nouvelles méthodes:

```
class Saint extends Monk {
   ...
   void defend () {
      super.defend()
      this.life = this.life + 1;
    }
   String greeting() {
       return (super.greeting() +
                 "␣and␣I␣will␣save␣your␣soul");
   }
   void heal (Character c) {
       c.life = c.life + 5;
   }
}
```

Héritage

Je peux aussi ajouter de nouveaux champs:

```
class Saint extends Monk {
   ...
   int saintity;
   void defend () {
      super.defend()
      this.life = this.life + 1;
    }
   String greeting() {
       return (super.greeting() +
                 "␣and␣I␣will␣save␣your␣soul");
   }
   void heal (Character c) {
       c.life = c.life + saintity;
   }
}
```

Héritage

class B extends A

- B est un sous-type de A;
- Dans B je peux implémenter des méthodes abstraites de A
- Dans B je peux rédefinir des méthodes de A
- Je peux ajouter de nouveaux champs et des nouvelles méthodes

UPEC
UNIVERSITÉ PARIS-EST CRÉTEIL VAL DE MARNE
Connaissance - Action

Encore sur la rédefinition

Je peux me tromper à écrire:

```
class Dwarf extends FriendlyCharacter {
   ...
   String greetings() {
       return ("Hello, I will guide you " +
                "through darkness");
   }
}
```

Je voulais rédefinir. Aucune erreur ne sera signalée

Encore sur la rédefinition

Utiliser une *annotation*

```
class Dwarf extends FriendlyCharacter {
   ...
   @Override
   String greetings() {
       return ("Hello, I will guide you " +
               "through darkness");
   }
}
```

Je signale au compilateur que je voulais rédefinir. Une erreur sera signalée

Compléments

- Utilisation de l'api
- import
- Collections
- Boucle for sur les listes

UPEC
UNIVERSITÉ PARIS-EST CRÉTEIL VAL DE MARNE
Connaissance - Action