Travaux Dirigés 3 Bis - DS de l’an dernier 2ème session

Print this page

Ce TD est une adaptation du DS de 2ème session de l'an dernier. La coïncidence est que le sujet est très proche de ce que nous avons vu en TP.
L’objectif n’est pas de répondre à l’énnoncé à l’aide d’une IA, les réponses permettent de vérifier les connaissances acquises en cours et permettre de revenir dessus pour vous préparer à l’examen.

A la fin, vous pourrez cliquer sur “Print this page” pour imprimer vos réponses en PDF et les déposer sur le e-campus.

> Normalement, vos réponses sont sauvegardées toutes les secondes dans le localStorage de votre navigateur, donc si vous actualisez la page, vous ne devriez pas perdre vos réponses.

Le code source est fourni en annexe, en bas de page mais vous pouvez l’ouvrir dans un nouvel onglet : Code source

But du programme

Le programme proposé en annexe permet de simuler la création de personnages dans un jeu type RPG (Role Play Game) et de leurs interactions dans le jeu. Les personnages peuvent être de différents types, jouables ou non et sont créés avec des caractéristiques précises (nom, attaque, défense, soin, arme…). Dans ce programme, tous les personnages sont créés de la même façon.

Compréhension du programme et Analyse de la conception (12 points)

Avant de répondre aux questions, étudiez-bien le code. Vous pouvez dessiner rapidement au brouillon le diagramme de classes si cela vous aide à le comprendre (nom des classes/interfaces et liens entre elles).

1. Ecrivez ce qui s’affiche à l’écran lorsque le programme est exécuté jusqu’au commentaire « //FIN QUESTION 2.1 »

```
Attention :
- Il y a deux sauts de lignes (car on appelle println et non print)
- La valeur par défaut d'un int est 0 et non null ou encore une chaîne vide


Je suis un personnage du nom de Garrosh
J'ai 150 points de vie
J'ai 100 points d'attaque
J'ai 5 points de défense
J'ai 0 points de soin
Ma classe est : guerrier


Je suis un personnage du nom de Anduin
J'ai 100 points de vie
J'ai 10 points d'attaque
J'ai 2 points de défense
J'ai 100 points de soin
Ma classe est : soigneur


Je suis un personnage du nom de Eric
J'ai 0 points de vie
J'ai 0 points d'attaque
J'ai 0 points de défense
J'ai 0 points de soin
Je suis un personnage non joueur
```

2. Pourquoi la définition de la classe « PNJ » est nécessaire, alors que la classe est vide ?

Elle permet quand même de faire la différence entre un personnage et un PNJ, elle est en effet vide, mais le constructeur par défaut est appelé et les attributs de la classe parent sont conservés

3. Pourquoi appeler la méthode « super() » dans les constructeurs des classes « Warrior », « PNJ » et « Healer » ? (ne répondez pas “pour appeler le constructeur parent”)

Normalement le constructeur par défaut est automatiquement appelé via "super", toutefois, dans le cas ou la classe parent ne définit pas de constructeurs sans paramètres, les classes dérivées doivent appeler explicitement le constructeur hérité.
Au passage cela permet d'initialiser les personnages de la même manière indépendamment de leur classe et factoriser le code commun

4. Pourquoi l'attribut « weapon » de la classe « AbstractCharacter » est-il déclaré protected ? (ne répondez pas, pour qu'elle soit accessible dans les classes filles)

```
Attention à ceux qui m'ont répondu
"Pour que l'attribut weapon ne soit pas ne soit pas modifiable par les classes filles ce qui
impliquerait que un personnage pourrait modifier les attribut d'une armes."
"C'est pour ne pas avoir a appeler les accesseurs/mutateurs"
Et tous ceux qui m'ont fait une réponse pour laquelle l'attribut aurait pu être public !

S'il n'est pas protected :
Il aurait pu être public, mais nous n'aurions pas utilisé l'encapsulation et cela aurait induit qu'on puisse équiper
un PNJ ou un soignant d'une arme
S'il avait été private, les guerriers n'auraient pas pu équiper d'armes

Le fait qu'il soit protected laisse la responsabilité au classe filles de s'en servir ou non, c'est le cas pour
Guerrier qui prends en paramètre une arme, mais on pourrait très bien imaginer que tous les soigneurs aient une arme
par défaut qu'on ne puisse changer
```

5. Donnez les étapes pour ajouter un nouveau type de personnage jouable "Hunter" et l'utiliser dans la fabrique « CharacterFactory » ? (ne répondez pas « Il faut créer une classe « Hunter »).

```
Nouvelle classe qui hérite de AbstractCharacter
Définir son constructeur en appelant le constructeur parent
Éventuellement lui faire implémenter IAttack
Enrichir l'énum du type de personnage
Modifier la fabrique
Éventuellement rajouter un cas de test dans initList()
```

6. Pourquoi dans le main() de la classe « Jeu », est-il impossible d'appeler directement la méthode « attack » sur l'objet « warrior » ? Que faudrait il faire pour que l'on puisse l'appeler ?

```
Attention à ceux qui ont la solution radicale d'implémenter l'interface dans l'interface ICharacter

Réponse Nathan :

Dans le main() de la classe "Jeu", il est impossible d'appeler directement la méthode
"attack" sur l'objet "warrior" car la référence "warrior" est déclarée comme une interface
"ICharacter", et non comme une classe "Warrior".
On effectue un castage
Warrior warrior = (Warrior) factory.createCharacters(CharacterType.Warrior, "Diablo");
warrior.attack(healer);
```

Développements en Java (10 points)

1. Ecrivez la méthode toString() de la classe « Hunter » (créée à la question précédente) pour que la sortie soit la suivante :

« Je suis un personnage du nom de Rexxar

J'ai 150 points de vie

J'ai 50 points d'attaque

J'ai 30 points de défense

J’ai 10 points de soin

Ma classe est : chasseur »

```
Ne redéfinissez pas tout et profitez de l'héritage pour appeler super et simplement rajouter :

Pensez au Override

@Override
public String toString() {
  return super.toString() +
    System.getProperty("line.separator") + "Ma classe est : chasseur";
}
```

2. Ecrivez la méthode « heal() » de la classe « Healer ». (Cette méthode fonctionne comme la méthode « attack() » de la classe « Warrior » à la différence qu’elle redonne des pv au lieu d’en enlever).

```
@Override
 public void heal(ICharacter dest) {
 int heal= this.getHeal();
 if (this.isActionCritique(5)) {
 heal *= 2;
 }
 if (heal > 0)
 ((AbstractCharacter) dest).setHeal(heal);
 }
```

3. Que fait la méthode « isActionCritique() » de la classe « AbstractCharacter » ? (ne répondez pas qu’elle compare le taux à un nb aléatoire).

```
Attention au vocabulaire :
paramètre et attribut

Cette méthode permet de dire si une attaque est considérée comme une attaque critique. Elle génère un nombre
aléatoire
et renvoie true ou false en fonction du pourcentage de chance passée en paramètre de la fonction.
```

4. Ecrivez la méthode « setHeal() » de la classe « AbstractCharacter » en réfléchissant à son fonctionnement. (Attention, un personnage ne peut pas avoir plus de points de vie après un soin qu'il n'en avait à sa création).

```
protected void setHeal(int heal) {
this.pv += heal;
if (this.pv > this.pvBase)
this.pv = this.pvBase;
}
```