Programmation avancée
TP 2 – Héritage

A-G. Bosser, ENIB
V. Padovani, PPS - IRIF

Un Terminator est une machine programmée pour tuer. Il y a deux sortes de Terminators : les T800 et les TX. Les T800 se ressemblent tous et peuvent être amicaux ou hostiles. Les TX sont des Terminators dits *Métamorphes* : ils sont capables de prendre l'apparence d'un être humain. Un TX a également comme particularité de pouvoir reprogrammer un T800 pour le rendre hostile.

Le but de ce TD est d'écrire une hiérarchie de classes modélisant les comportements des différentes sortes de Terminators. Les classes seront déclarées dans des fichiers distincts.

Exercice 1 : Le scénario du film

La classe `Scenario` sera la classe principale du programme. Définissez la méthode `main` de cette classe. Pour l'instant, elle se contentera d'afficher `"Dialogue entre machines."` Dans chacun des exercices suivants, vous testerez le fonctionnement des nouvelles classes dans cette méthode.

Exercice 2 : Les Terminators

La classe `Terminator` représente tous les modèles de Terminators possibles. Définissez-là, sachant que :

1. Tout Terminator possède un numero de série.
   Ajoutez à la classe un champ privé entier représentant ce numéro.
2. Tout Terminator, reçoit, à sa construction, un numéro de série.
   Ajoutez à la classe un constructeur prenant ce numéro en argument.
3. Lorsqu'il se présente et s'il s'agit d'un modèle standard, un Terminator dit :
   `"Je suis un Terminator et mon numéro de série est ... "`.
   Ajoutez à la classe une méthode `sePresente()` affichant ce message.
4. Lorsqu'il termine quelqu'un et s'il s'agit d'un modèle standard, un Terminator dit :
   `"Vous êtes terminé !"`.
   Ajoutez une méthode `termine()` affichant ce message.

Exercice 3 : Les T800

Le T800 est un modèle particulier de Terminator. Définissez une classe `T800`, héritière de `Terminator`. Complétez cette classe, sachant que :

1. Un T800 est soit amical, soit hostile. Par défaut, à sa construction, il est amical.
   Ajouter à la classe un champ privé booléen représentant l'état courant d'un T800, ainsi qu'un accesseur `setAmical` permettant de modifier la valeur de ce champ.

2. Comme tout Terminator, un T800 reçoit à sa construction un numéro de série.
   Ajoutez à la classe un constructeur prenant ce numéro en argument, et invoquant le constructeur de la classe parente `Terminator` à l'aide du mot-clef `super` [1].
   Que se passe-t-il si vous commentez cette invocation ? Pourquoi ?
3. Un T800 peut se comporter de manière non standard quand il "termine" une cible. S'il est amical, et seulement dans ce cas, au lieu du message habituel, il dira plutôt :

   ```
   "Hasta la vista, Baby..."
   ```

   Redéfinir dans la classe `T800` la méthode `termine()`. Selon la valeur du champ (amical ou hostile) la méthode : affichera le message précédent ; ou invoquera l'implémentation parente via le mot-clef `super` [2]

Exercice 4 : Les métamorphes

Les Métamorphes forment l'ensemble des modèles de Terminators capables de changer leur apparence. Définissez une classe `Metamorphe`, héritière de `Terminator`. Complétez cette classe, sachant que :

1. Tout Métamorphe est capable de prendre l'apparence de n'importe quelle autre entité humanoïde, mais à sa construction, il ne ressemble à personne en particulier.
   Ajouter à la classe un champ privé de type `String`, représentant l'apparence courante d'un Métamorphe, sous la forme d'une chaî ne caractères. La valeur d'initialisation de ce champ sera, par défaut, la chaîne de caractère `"personne"` en toutes lettres.
2. Lorsqu'un Métamorphe se présente, il commence par énoncer la phrase habituelle d'un Terminator standard, puis il complète cette présentation en indiquant son apparence courante :

   ```
   "Je suis sous l'apparence de ... "
   ```

   Redéfinir dans la classe la méthode `sePresente()` pour obtenir ce comportement – il vous faudra, encore une fois, accéder à l'ancienne implémentation via `super`.
3. Un metamorphe peut changer d'apparence.
   Ajouter à la classe un accesseur `setApparence` prenant en argument une nouvelle apparence, toujours sous la forme d'une chaîne de caractères.

Exercice 5 : Les TX

Le TX est un modèle particulier de Métamorphe. Définissez une classe `TX`, héritière de `Métamorphe`. Complétez cette classe, sachant que :

1. Comme tout Terminator, un TX reçoit à sa construction un numéro de série, mais également une apparence.
   Ajouter à la classe le constructeur correspondant.

1. Si le paramètre du constructeur s'appelle `numero`, cette invocation s'écrira `super(numero);`, elle lancera l'éxécution du constructeur de `Terminator` en lui passant l'argument `numero`. Le champ correspondant est défini comme privé dans la classe `Terminator` : même si ce champ *existe* dans les instances de `T800`, il est inaccessible en écriture sans un nouvel accesseur dans `Terminator`. Mais le super-constructeur de `T800` *voit* ce champ, donc peut l'initialiser.
2. Cette invocation s'écrit `super.termine()`.

2. Un TX est capable de reprogrammer un T800 pour le rendre hostile.
   Ajouter à la classe une méthode `void reprogramme(T800 terminator)`. Son invocation devra rendre hostile le Terminator argument.

Exercice 6 :

Il est clair que l'on ne souhaite construire aucun Terminator qui ne soit pas un modèle de série. Transformez les classes `Terminator` et `Metamorphe` en classes abstraites pour empêcher la création d'instances de ces classes. Il suffit d'ajouter le mot-clef `abstract` devant leur déclaration. Assurez-vous que vous avez testé toutes les classes et toutes les méthodes dans votre classe `Scenario`.

Exercice 7 : Pour les plus rapides : les humains pris pour cibles

Dans cet exercice, vous allez définir de nouvelles classes, et sans doute modifier votre programme. Vous serez moins guidés dans votre modélisation, il pourrait donc être utile de réfléchir un peu avant de vous lancer dans l'écriture du code.

1. Un être humain est caractérisé par son nom, son âge, et son sexe. Son identifiant (ce qui permet de le distinguer de tous les autres humains) est son numéro de sécurité sociale.
2. Un terminator est créé pour tuer les humains que l'on lui a donnés pour cibles, avec une priorité de "terminaison" pour chaque cible. Plus sa priorité est élevée, plus il est urgent de terminer une cible.
3. Tout T800 a un unique propriétaire, un être humain, auquel il obéit s'il est amical : dans ce cas, il prendra pour cibles l'ensemble de celles désignées son propriétaire. S'il devient hostile, il prendra pour cibles non seulement les précédentes, mais aussi son proprétaire, avec la priorité la plus haute possible.

Ecrire une méthode qui, pour chaque être humain, est capable, étant donné un tableau de T800, d'afficher la liste des terminators lancés à ses trousses, ainsi que la priorité avec laquelle il a été pris pour cible par chaque terminator.