Programmation Objet

Florian Boudin

Révision 3 du 24 juillet 2014

Présentation du module

- notions abordées
  - objets, classes, encapsulation, héritage, etc.
- pré-requis : *background* en programmation
- volume horaire
  - 9 CM - 9 TD - 18 TP (48h)
- notation
  - 25% CC, 25% projet, 50% examen

Présentation du module (cont.)

- cours basé sur les tutoriels Java SE de Oracle
  - http://docs.oracle.com/javase/tutorial/
- supports disponibles sur madoc/GitHub
  - http://madoc.univ-nantes.fr/enrol/index.php?id=22688
  - https://github.com/boudinfl/progObjet

Plan

- Introduction à la programmation objet
  - Objets
  - Classes
  - Héritage
  - Interfaces
  - Packages
  - Questions-Réponses

Qu'est ce qu'un objet ?

- regardez autour de vous pour trouver des exemples d'objets réels : une chaise, un tableau, etc.
- les objets réels partagent deux caractéristiques : un *état* et un *comportement*
- exemple : un stylo
  - état : couleur, marque, etc.
  - comportement : écrire, tomber, etc.

Qu'est ce qu'un objet ? (cont.)

- identifier les états et comportements d'objets réels est une bonne façon d'appréhender la programmation objet
- pour chaque objet, il faut se poser deux questions :
  - quels sont les états possibles de cet objet ?
  - quels sont les comportements possibles de cet objet ?
- prenez une minute pour observer les objets autour de vous et écrire vos observations

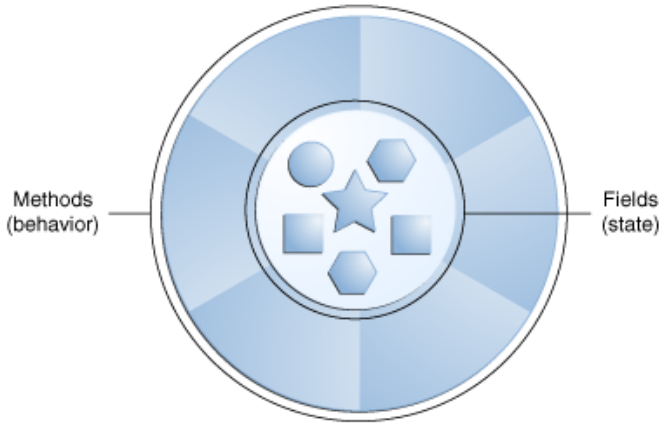
Qu'est ce qu'un objet ? (cont.)

- les objets varient en *complexité*
  - lampe de bureau vs. téléphone mobile
- certains objets contiennent d'autres objets
  - e.g. trousse contient des stylos
- toutes ces observations se traduisent en programmation objet

La notion d'objet

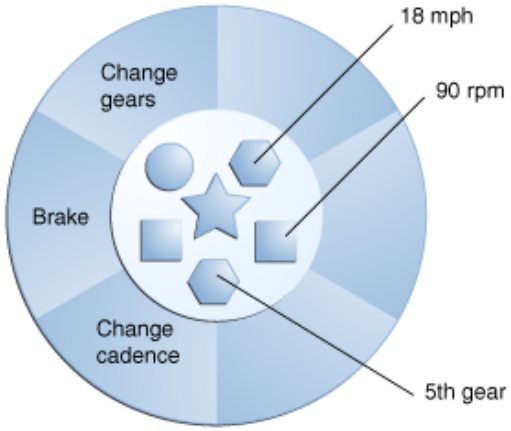
- un objet est une *structure de données*
- un objet est caractérisé par :
  - un état stocké dans des *champs* → variables
  - un comportement défini par des *méthodes* → fonctions
- les méthodes modifient l'état interne d'un objet et permettent aux objets de communiquer

Représenter un objet



Nom de la classe
champs
méthodes

Un exemple d'objet : un vélo



- définir un état et des méthodes permettent de contrôler l'objet
  - e.g. si le vélo n'a que 7 vitesses, une méthode pour changer de vitesse doit rejeter toute valeur < 7 ou > 7

Les avantages des objets

- *modularité*
  - le code source d'un objet peut être écrit et maintenu indépendamment de celui d'autres objets
- *encapsulation*
  - en ne proposant que des méthodes pour manipuler un objet, les détails de son implémentation restent masqués
- *réutilisation du code*
  - un objet existant peut être utilisé dans votre programme
  - déléguer le développement d'objets complexes à des spécialistes

Les avantages des objets (cont.)

- *débogage plus simple*
  - si un objet est problématique, vous pouvez simplement le remplacer par un autre
- analogue à la résolution de problèmes mécaniques
  - si un boulon casse, vous le remplacez, pas toute la machine

Plan

- Introduction à la programmation objet
  - Objets
  - Classes
  - Héritage
  - Interfaces
  - Packages
  - Questions-Réponses

Qu’est ce qu'une classe ?

- il existe des milliers de vélos, tous construits sur le même modèle et partageant les mêmes composants
- une *classe* est une description des caractéristiques d’un ou de plusieurs objets
- un objet créé à partir d’une classe est une *instance* de la classe
  - e.g. le vélo de Pierre est une *instance* de la classe *vélo*

Un exemple de classe `Bicycle`

```
class Bicycle {
  int cadence = 0;
  int speed = 0;
  int gear = 1;

  void changeCadence(int newValue) {
    cadence = newValue;
  }

  void changeGear(int newValue) {
    gear = newValue;
  }

  void speedUp(int increment) {
    speed = speed + increment;
  }

  void applyBrakes(int decrement) {
    speed = speed - decrement;
  }

  void printStates() {
    System.out.println("cadence:" + cadence + " speed:" + speed + " gear:" + gear
  }
}
```

Un exemple de classe `Bicycle`

- les champs `cadence`, `speed` et `gear` représentent l'état de l'objet
- les méthodes définissent les interactions de l'objet avec le monde extérieur
- il n'y a pas de méthode `main`
  - il ne s’agit pas d’une application mais d’un modèle pour des vélos qui pourraient être utilisés dans une application
  - la responsabilité de créer et d’utiliser de nouveaux objets vélo appartient à une autre classe dans votre application

Une première application

```
class BicycleDemo {
  public static void main(String[] args) {

    Bicycle bike1 = new Bicycle(); // Create two different
    Bicycle bike2 = new Bicycle(); // Bicycle objects

    bike1.changeCadence(50); // Invoke methods on
    bike1.speedUp(10); // those objects
    bike1.changeGear(2);
    bike1.printStates();

    bike2.changeCadence(50);
    bike2.speedUp(10);
    bike2.changeGear(2);
    bike2.changeCadence(40);
    bike2.speedUp(10);
    bike2.changeGear(3);
    bike2.printStates();

  }
}

// sortie du programme
// cadence:50 speed:10 gear:2
// cadence:40 speed:20 gear:3
```

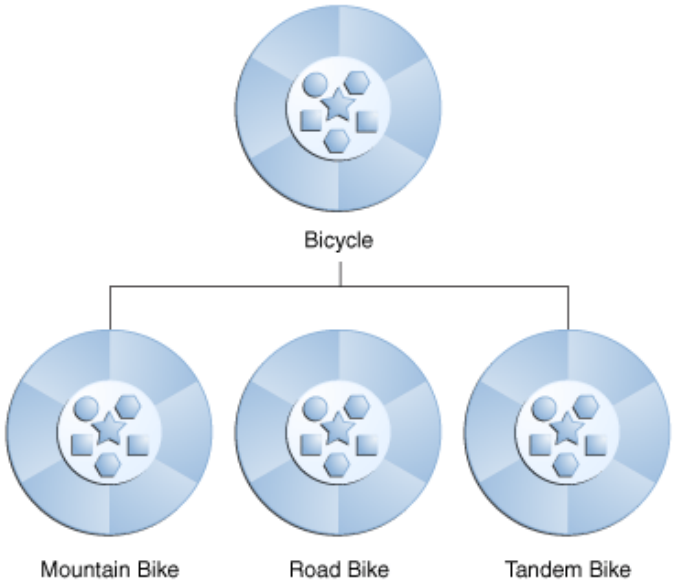
Plan

- Introduction à la programmation objet
  - Objets
  - Classes
  - Héritage
  - Interfaces
  - Packages
  - Questions-Réponses

Qu'est ce que l'héritage ?

- les VTT et les tandems partagent les caractéristiques des vélos mais ils en ont aussi des nouvelles :
  - deux selles et deux guidons pour les tandems
  - un pédalier compact pour les VTT
- en programmation objet, les classes peuvent hériter des champs et des méthodes d'autres classes
- exemple de classe `MountainBike`
  - `Bicycle` est la *super-classe* de `MountainBike`
  - `MountainBike` est une *sous-classe* de `Bicycle`

Exemple de hiérarchie



- syntaxe pour créer une sous-classe :
  - mot-clé `extends` suivi du nom de la classe à hériter
  - en Java, une classe ne peut avoir qu'une super-classe et une super-classe peut avoir un nombre illimité de sous-classes

La sous-classe `MountainBike`

```
class MountainBike extends Bicycle {
  // new fields and methods defining
  // a mountain bike would go here
}
```

- `MountainBike` possède tous les champs et les méthodes de la classe `Bicycle`
- développer les champs et méthodes spécifiques
- bien *documenter* chaque super-classe car son code sera absent de la source des sous-classes

Plan

- Introduction à la programmation objet
  - Objets
  - Classes
  - Héritage
  - Interfaces
  - Packages
  - Questions-Réponses

Qu'est ce qu'une interface ?

- les objets définissent leurs interactions avec le monde extérieur par le biais des méthodes qu'elles exposent
- les méthodes forment l'*interface* de l'objet avec le monde extérieur
  - e.g. les boutons sur votre téléviseur sont l'interface entre vous et le câblage électrique de l'autre côté du boîtier
- dans sa forme la plus commune, une interface est un ensemble de méthodes vides

L'interface de `Bicycle`

```
interface Bicycle {

  void changeCadence(int newValue);
  void changeGear(int newValue);
  void speedUp(int increment);
  void applyBrakes(int decrement);

}
```

- pour implémenter une interface, vous devez utiliser le mot-clé `implements` et changer le nom de votre classe

Implémenter une interface

```
class Bicloo implements Bicycle {

  // remainder of this class implemented
  // as before

}
```

- implémenter une interface permet de définir formellement le comportement d'une classe
- le compilateur vérifie si *toutes* les méthodes définies dans l'interface sont présentes dans le code source de la classe

Plan

- Introduction à la programmation objet
  - Objets
  - Classes
  - Héritage
  - Interfaces
  - Packages
  - Questions-Réponses

Qu'est ce qu'un package ?

- un *package* (paquet) est un mécanisme de nommage qui permet de regrouper les classes et interfaces
- concept similaire au principe des répertoires
- les logiciels peuvent être composés de plusieurs centaines ou milliers de classes
  - garder les choses organisées en plaçant les classes et les interfaces liées en paquets

L'API Java

- Java fourni une bibliothèque de classes que vous pouvez utiliser dans vos applications
  - Application Programming Interface (API)
  - contient tout ce qui est nécessaire pour les tâches de programmation classiques : String, File, Socket, GUI, etc.
- la Java Platform API Specification contient le détails complet de tous les paquets, interfaces, classes, champs et méthodes disponibles
  - http://docs.oracle.com/javase/7/docs/api/

Plan

- Introduction à la programmation objet
  - Objets
  - Classes
  - Héritage
  - Interfaces
  - Packages
  - Questions-Réponses

Questions

1. les objets réels partagent deux caractéristiques : un ____ et un _____
2. l'état d'un objet est stocké dans des ______
3. le comportement d'un objet est défini par des ________
4. masquer les données d'une classe et y empêcher l'accès par un autre moyen que les méthodes est le principe d'_____
5. une ______ est une description des caractéristiques d'un ou de plusieurs objets

Réponses

1. les objets réels partagent deux caractéristiques : un état et un comportement
2. l'état d'un objet est stocké dans des champs
3. le comportement d'un objet est défini par des méthodes
4. masquer les données d'une classe et y empêcher l'accès par un autre moyen que les méthodes est le principe d'encapsulation
5. une classe est une description des caractéristiques d'un ou de plusieurs objets

Questions

1. le comportement commun peut être défini dans une _____ et hérité dans une _____ en utilisant le mot-clé _____
2. un ensemble de méthodes sans implémentation est une _____
3. un _____ est un mécanisme de nommage qui permet de regrouper les classes et interfaces
4. API signifie _____ _____ _____

Réponses

1. le comportement commun peut être défini dans une super-classe et hérité dans une sous-classe en utilisant le mot-clé extends
2. un ensemble de méthodes sans implémentation est une interface
3. un package est un mécanisme de nommage qui permet de regrouper les classes et interfaces
4. API signifie Application Programming Interface

Etude de cas : Tetris

- quels sont les objets/classes nécessaires ?
  - quels sont les états possibles de chaque objet ?
  - quels sont les comportements possibles de chaque objet ?