Duplication du code

La duplication de code est un indicateur d'une mauvaise conception. La classe `Game` contient un exemple de duplication de code (code 7.1). Le problème de la duplication de code est que dès qu'une modification est apportée à une version, elle doit être également apportée à l'autre version pour éviter toute incohérence. Cela augmente la charge de travail du programmeur responsable de la maintenance, et cela introduit aussi un risque d'erreur. Fréquemment, un programmeur trouve une copie du code, la modifie et estime que le travail est achevé. Rien n'indique l'existence d'une seconde copie du code, qui peut donc rester inchangée.

Concept

Duplication du code. Lorsque le même segment de code se répète dans une application, c'est le signe d'une conception erronée. À éviter.

Code 7.1 • Quelques parties choisies de la classe (mal conçue) `Game`.

```
public class Game
{
   // code non reproduit

    private void createRooms()
    {
        Room outside, theatre, pub, lab, office;

        // création des pièces
        outside = new Room("à l'extérieur de l'entrée
           principale de l'Université");
        theatre = new Room("dans un amphithéâtre");
        pub = new Room("à la cafétéria");
        lab = new Room("dans la salle informatique");
        office = new Room("au bureau des techniciens");

        // initialisation des sorties des pièces
        outside.setExits(null, theatre, lab, pub);
        theatre.setExits(null, null, null, outside);
        pub.setExits(null, outside, null, null);
        lab.setExits(outside, office, null, null);
        office.setExits(null, null, null, lab);
```

```
        currentRoom = outside;  // début de la partie
                                // ailleurs
    }

// code non reproduit
    /**
     * Affichage du message d'accueil au joueur.
     */
    private void printWelcome()
    {
        System.out.println();
        System.out.println("Bienvenue au jeu de Zuul !");
        System.out.println("Zuul est un nouveau jeu
           d'aventure terriblement ennuyeux.");
        System.out.println("Tapez 'aide' si vous avez besoin
           d'aide.");
        System.out.println();
        System.out.println("Vous êtes dans " +
           currentRoom.getDescription());
        System.out.print("Les sorties : ");
        if(currentRoom.northExit != null) {
            System.out.print("nord ");
        }
        if(currentRoom.eastExit != null) {
            System.out.print("est ");
        }
        if(currentRoom.southExit != null) {
            System.out.print("sud ");
        }
        if(currentRoom.westExit != null) {
            System.out.print("ouest ");
        }
        System.out.println();
    }

// code non reproduit
    /**
     * Essai de déplacement dans une direction.
     * Si une sortie existe, entrer dans la nouvelle pièce,
     * sinon afficher un message d'erreur
     */
    private void goRoom(Command command)
    {
        if(!command.hasSecondWord()) {
            // en cas d'absence du deuxième mot,
            // nous ne savons pas où aller
            System.out.println("Aller où ?");
            return;
        }

        String direction = command.getSecondWord();
```

```
        // nous essayons de quitter la pièce courante.
        Room nextRoom = null;
        if(direction.equals('nord')) {
            nextRoom = currentRoom.northExit;
        }
        if(direction.equals("est")) {
            nextRoom = currentRoom.eastExit;
        }
        if(direction.equals("sud")) {
            nextRoom = currentRoom.southExit;
        }
        if(direction.equals("ouest")) {
            nextRoom = currentRoom.westExit;
        }
        if (nextRoom == null) {
            System.out.println("Il n'y a pas de porte !");
        }
        else {
            currentRoom = nextRoom;
            System.out.println("Vous êtes " +
               currentRoom.getDescription());
            System.out.print("Les sorties : ");
            if(currentRoom.northExit != null) {
                System.out.print("nord ");
            }
            if(currentRoom.eastExit != null) {
                System.out.print("est ");
            }
            if(currentRoom.southExit != null) {
                System.out.print("sud ");
            }
            if(currentRoom.westExit != null) {
                System.out.print("ouest ");
            }
            System.out.println();
        }
    }

// code non reproduit
}
```

Les méthodes `printWelcome` et `goRoom` contiennent toutes les deux les lignes de code suivantes :

```
System.out.println("Vous êtes " +
      currentRoom.getDescription());
System.out.print("Les sorties : ");
if(currentRoom.northExit != null) {   System.out.print("nord ");
}if(currentRoom.eastExit != null) {   System.out.print("est ");
}if(currentRoom.southExit != null) {   System.out.print("sud ");
}if(currentRoom.westExit != null) {   System.out.print("ouest ");
}System.out.println();
```

La duplication de code est habituellement un symptôme de mauvaise cohésion. Le problème a pour origine le fait que les deux méthodes en question réalisent deux tâches : `printWelcome` affiche un message d'accueil et les informations propres à la situation courante, alors que `goRoom` modifie la pièce courante et affiche alors les informations propres à la nouvelle situation.

Les deux méthodes affichent des informations liées à la situation courante, mais aucune des deux ne peut appeler l'autre car elles font également d'autres choses. C'est une mauvaise conception.

Une meilleure conception consiste à utiliser une méthode distincte, plus cohérente, dont l'unique tâche est d'afficher les informations liées à la situation courante (code 7.2). Les méthodes `printWelcome` et `goRoom` peuvent alors toutes les deux appeler cette méthode quand elles ont besoin d'afficher ces informations. Ainsi, nous évitons d'écrire deux fois le même code et nous ne le modifierons qu'une seule fois quand cela sera nécessaire.

Code 7.2 • `printLocationInfo` en tant que méthode distincte.

```
private void printLocationInfo()
{
   System.out.println("Vous êtes " +
      currentRoom.getDescription());
   System.out.print("Les sorties : ");
   if(currentRoom.northExit != null) {
         System.out.print("nord ");
   }
   if(currentRoom.eastExit != null) {
         System.out.print("est ");
   }
   if(currentRoom.southExit != null) {
         System.out.print("sud ");
   }
   if(currentRoom.westExit != null) {
         System.out.print("ouest ");
   }
       System.out.println();
}
```