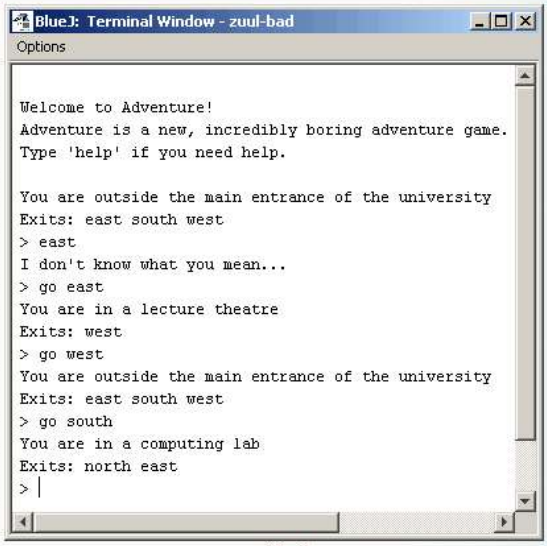
Itération 1 : Projet de programmation d'un jeu d'aventures

Pierre Lefebvre

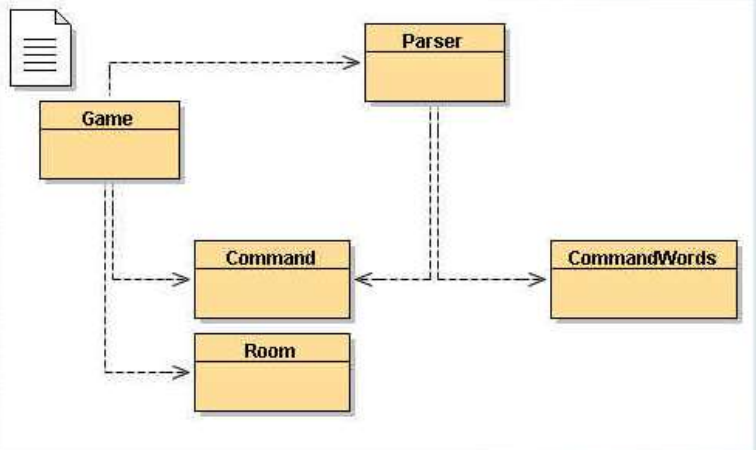
Le projet : Un jeu d'aventure



zuul-bad

zuul-with-images

La version initiale du jeu est constituée de 5 classes



- La classe CommandWords recense les mots-clés autorisés et détermine si une commande de l'utilisateur est valide.
- La classe Command engendre une commande de l'utilisateur.
- La classe Room permet de créer les pièces du jeu, une pièce représente une scène.
- La classe Parser permet de lire le texte entré par l'utilisateur et le transforme en commande.
- La classe Game est la classe principale du jeu (méthode play)

- *Lire le chapitre 7 jusqu'à l'exercice 7.1 non inclus.*
- Faire l'exercice **7.1** (découverte zuul-bad) :
  Pour un premier contact, parcourir les sources du projet `zuul-bad` (*ne pas chercher à comprendre le code source pour l'instant*).
  Pour pouvoir l'utiliser puis en modifier la programmation, il faut enregistrer ce fichier zuul-bad.jar, puis utiliser (juste pour la première fois) "Ouvrir non-BlueJ ..." dans BlueJ ; ensuite, il suffira d'utiliser "Ouvrir un projet ..." .

  (This project is called 'bad' because its implementation contains some bad design decisions, and we want to leave no doubt that this should not be used as an example of good programming practice!)
  Execute and explore the application. The project comment gives you some information about how to run it.
  While exploring the application, answer the following questions (and write the answers into the report) :

  - *What does this application do?*
  - *What commands does the game accept?*
  - *What does each command do?*
  - *How many rooms are in the scenario?*

  Draw a map of the existing rooms.

- Faire l'exercice **7.2** (rôle des classes) :

  After you know what the whole application does, try to find out what each individual class does.
  Write down for each class the purpose of the class.
  You need to look at the source code to do this. Note that you might not (and need not) understand all of the source code. Often, reading through comments and looking at method headers is enough.

- Exercice **7.2.1** : Lire la documentation de la classe `Scanner` et comprendre comment elle est utilisée dans zuul-bad.
- Apprentissage : `Scanner`
- *Lire le chapitre 7 jusqu'à l'exercice 7.3 non inclus.*
- Faire l'exercice **7.3** (scénario libre) :

  Design your own game scenario. Do this away from the computer. Do not think about implementation, classes, or even programming in general. Just think about inventing an interesting game. This could be done with a group of people.
  The game can be anything that has as its base structure a player moving through different locations. Here are some examples:

- *You are a white blood cell traveling through the body in search of viruses to attack…*
- *You are lost in a shopping mall and must find the exit ...*
- *You are a mole in its burrow and you cannot remember where you stored your food reserves before winter ...*
- *You are an adventurer who searches through a dungeon full of monsters and other characters ...*
- *You are from the bomb squad and must find and defuse a bomb before it goes off ...*

Make sure that your game has a goal (so that it has an end and the player can 'win').
Try to think of many things to make the game interesting (trap doors, magic items, characters that help you only if you feed them, time limits, whatever you like). Let your imagination run wild.
At this stage, do not worry about how to implement these things.

1) Ne pas se limiter et imaginer le jeu idéalement souhaité à la fin du projet.
Pour donner une idée des fonctionnalités attendues d'ici à la fin du projet, voici un petit résumé du minimum qui sera demandé : gérer des déplacements dans au moins 4 directions plus haut et bas ; pouvoir ramasser/porter/reposer des objets avec une certaine limite de poids/prix qui peut-être augmentée en mangeant un "gateau magique" ; charger/utiliser un "téléporteur" pour vous ramener dans un lieu précédemment visité ; prévoir une pièce qui lorsqu'on la quitte vous téléporte de façon aléatoire dans un autre lieu du jeu ; gérer des "passages" entre lieux empruntables dans un seul sens et/ou fermés à clé ; gérer des personnages fixes ou mobiles, pouvant dialoguer ou non, pouvant échanger des objets ou non ; ...
2) Ne pas se focaliser sur la future interface avec l'utilisateur (graphique, souris, clavier) mais s'en tenir au scénario.

- Exercice **7.3.2** : Dessiner un plan du jeu faisant bien apparaître la géographie des lieux, le nom des lieux, ainsi que les passages possibles entre-eux; *(Ce peut être un plan dessiné à la main, puis scanné.).*
- *Lire le chapitre 7 jusqu'à l'exercice 7.4 non inclus.*
- Faire l'exercice **7.4** (zuul-v1, rooms, exits)

Open the `zuul-bad` project, and save it under a different name (e.g. `zuul-v4`). This is the project you will use to make improvements and modifications throughout this chapter.
You can leave off the -bad suffix, since it will soon (hopefully) not be that bad anymore. As a first step, change the `createRooms` method in the `Game` class to create the rooms and exits you invented for your game. Test!

- *Lire le chapitre 7 jusqu'à l'exercice 7.5 non inclus.*

Faire l'exercice **7.5** (printLocationInfo).
Implement and use a separate `printLocationInfo` method in your project, as discussed in this section. Test your changes.

- *Lire le chapitre 7 jusqu'à l'exercice 7.6 non inclus.*
- Faire l'exercice **7.6** (getExit)

  Make the changes we have described to the `Room` and `Game` classes.

- Faire l'exercice **7.7** (getExitString).

  Make a similar change to the `printLocationInfo` method of `Game` so that the details of the exits are now prepared by the `Room` rather than the `Game`. Define a method in `Room` with the following signature `public String getExitString()`.

- *Lire le chapitre 7 jusqu'à l'exercice 7.8 non inclus.*
- Faire l'exercice **7.8** (HashMap, setExit).
- Exercice **7.8.1** : Ajouter dans le scénario (si ce n'est pas déjà le cas) et dans son jeu au moins un lieu sur un plan différent des autres lieux et nécessitant donc un déplacement vertical.
- *Lire le chapitre 7 jusqu'à l'exercice 7.9 non inclus.*
- Faire les exercices **7.9** (keySet) et **7.10** (getExitString CCM).
- Apprentissage : `HashMap` et `Set`
- *Lire le chapitre 7 jusqu'à l'exercice 7.11 non inclus.*
- Faire l'exercice **7.11** (getLongDescription).
- *Lire le chapitre 7 jusqu'à l'exercice 7.14 non inclus.*
- Faire les exercices **7.14** (look) et **7.15** (eat).
- *Lire le chapitre 7 jusqu'à l'exercice 7.16 non inclus.*
- Faire l'exercice **7.16** (showAll, showCommands).
- Apprentissage : `for ( typeElement element : tableau )`
- Faire l'exercice **7.17** (changer Game ?).
- Exercice **7.17.1** : S'assurer que chaque classe et chaque méthode ont leur commentaire javadoc, sinon compléter.
- Exercice **7.17.2** : Générer la javadoc grâce à BlueJ.

Bibliographie :

- Objects First with Java, A Practical Introduction using BlueJ de David J.Barnes et Michael Kölling.
  Prentice Hall / PearsonEducation, 2006