The full implementation of all changes discussed in this chapter so far is available in your code examples in a project named *zuul-better*. If you have done the exercises so far, you can ignore this project and continue to use your own. If you have not done the exercises, but want to do the following exercises in this chapter as a programming project, you can use the *zuul-better* project as your starting point.

## 7.10 Thinking ahead

The design we have implemented now is an important improvement to the original version. It is, however, possible to improve it even more.

One characteristic of a good software designer is the ability to think ahead. What might change? What can we safely assume will stay unchanged for the life of the program?

One assumption that we have hard-coded in most of our classes is that this game will run as a text-based game with terminal input and output. But will it always be like this?

It might be an interesting extension later to add a graphical user interface with menus, buttons, and images. In that case, we would not want to print the information to the text terminal any more. We might still have command words, and we might still want to show them when a player enters a help command. But we might then show them in a text field in a window, rather than using `System.out.println`.

It is good design to try to encapsulate all information about the user interface in a single class or a clearly defined set of classes. Our solution from Section 7.9 for example, the `showAll` method in the `CommandWords` class, does not follow this design rule. It would be nicer to define that `CommandWords` is responsible for *producing* (but not *printing!*) the list of command words, but that the `Game` class should decide how it is presented to the user.

We can easily achieve this by changing the `showAll` method so that it returns a `String` containing all command words instead of printing them out directly. (We should probably rename it `getCommandList` when we make this change.) This `String` can then be printed in the `printHelp` method in `Game`.

Note that this does not gain us anything right now, but we might profit from this improved design in the future.

**Exercise 7.18** Implement the suggested change. Make sure that your program still works as before.

**Exercise 7.19** Find out what the *model-view-controller* pattern is. You can do a web search to get information, or you can use any other sources you find. How is it related to the topic discussed here? What does it suggest? How could it be applied to this project? (Only *discuss* its application to this project, as an actual implementation would be an advanced challenge exercise.)