# Lab 5: Designing a class hierarchy

CSCI 245 ⋆ Programming II: Object-Oriented Design ⋆ Fall 2019

Devin J. Pohly <`devin.pohly@wheaton.edu`>

The main goal of this lab, and the project that it will grow into, is to gain experience designing the relationships among classes. This will also give you experience in working on a group project, which will involve using a revision control system (in our case, Git). Furthermore, in this project you will begin to see how various things we have been talking about in class are applied, including polymorphism and encapsulation; Java Collections classes like `ArrayList`, `HashMap`, and `HashSet`; and class extension (subclassing).

## Introduction and set-up

In this lab we initiate a running example that will come up in future lectures and labs: An "adventure"-style computer game. The central element of this kind of game is that it takes place in a world composed of locations (which we will call "rooms," even though it's possible that these locations could be interpreted as being outside) among which the user moves. In these rooms the user's character can find and use items, interact with other characters, and move to a different room. The object of the game might be to find something, rescue someone, or escape a cave or dungeon. Making this text-based gives it a element of retro-charm.

Need inspiration? Check out the following examples:

- The original: Colossal Cave Adventure, also called "`ADVENT`" for historical reasons (you can find some history and a play link at `http://rickadams.org/adventure/`)
- A themed tribute: `http://www.douglasadams.com/creations/infocomjava.html`
- A fun parody: `http://www.homestarrunner.com/dungeonman.html`

The code for the game we will work on is very flexible and (if designed well) extensible. It can be used to play games with a wide variety of scenarios, maps, and objectives. You will be working on it as a team, but each member of the team will be working on a different feature.

Teams are chosen at random and announced in lab. Get together with your teammates and choose a team name, then follow the GitHub invitation link on Schoology. You will be prompted to sign in to GitHub if you have an account, or to register if you do not. GitHub will create a repository for you with the lab's starter code and give you a URL, and you can clone it using the following command:

```
$ git clone https://yourusername@github.com/devinpohly/yourreponame
```

The code is in a folder/package called `game`. Make a new Eclipse project (remember, set up the project to be in the folder *containing* the package folder `game`). You may find it easier to run the program from the command line than through Eclipse. From the folder containing the `game` folder, run it with

```
$ java game.PlayGame
```

That is, the class `PlayGame` has the `main` method. The game software is in working order and can be played. However, right now the game is pretty lame. It consists of four rooms (each described only as "a room"), laid out as a 2-by-2 grid, with each room connected to its two adjacent rooms (not to the diagonal room). It is also a pretty annoying game, since there is no object and no way for the game to end.

In each turn:

- a description of the current room is displayed,
- the user is prompted, and
- the user enters a command to be interpreted.

## Goals

Your first task is to inspect the code to understand how it is set up. Then confer with each other and assign to each person at least one of the following tasks:

- Add the concept of an *item*. An item is something that is found in a room (a key, a weapon, food...) and possibly can be picked up by a user, carried around, and used. Design this in such a way that it is easy to add new kinds of items and so that kinds of items that have things in common can share code. **Note** that this will require changing other things so that a room can contain an item and so that a user can carry an item around.
- Add more commands that the game understands. Obvious additions are "help" (give a list of commands) and "look" (describe the room, where you can go from here, any items that are present, etc.). Design this in such a way that it is easy to add new commands, and for some commands to be room-, item-, or game-specific. **Hint**: Adding things to the if-else block in `Parser.executeTerm()` will not make this easy.
- Modify the code so that rooms can give the option of moving in various directions other than "north", "south", etc.—for example, northwest, up/down, upstream/downstream, inside, outside. Be creative! **Note** that this will affect the list of commands; design this is such a way that it will later be easy to add new kinds of rooms giving various options of movement. **Hint**: Adding a new instance variable to `Room.java` for every new movement option will not make this easy.

## Process

The first thing you must each do is read this lab description (which, presumably, you are currently doing). Then you will confer as a group and assign tasks to everyone.

Next you will take some time to think through what you need to do for your task and *plan* and *design*. You should do this in conjunction with whomever is working on tasks that affect or are affected by your task.

Once you have planned out your changes and considered how it will affect other parts of the program, begin to implement. In order to test, you will have to integrate your work using Git. Be sure that by the end of the lab period you have used Git for at least the following:

- Make a commit to your local repository (ideally, make several commits so that they are small, logical units)
- Push your local commits to GitHub
- Pull your teammates' commits from GitHub

This way we will be able to help you through if you run into any issues.

## To turn in

Work until lab time is finished. Ideally by the end of lab, your group will have done all four tasks at least to some extent, will have integrated your changes, and will have a working version. Be sure everyone's changes have been pushed to GitHub. We will resume working on this (with additional specifications) next week.