Assignment 1: The Adventure Game

Due date: The assignment must be handed in to your tutor **at the beginning of your tutorial class** in week 5 (14 - 18 August).

The Game

Your task is to invent and implement an adventure game. You have been given a simple framework (*Zork1*) that lets you walk through a couple of rooms. You can use this as a starting point.

1 Read The Code

Reading code is an important skill that you need to practise. You first task is to read some of the existing code and try to understand what it does. By the end of the assignment, you will need to understand most of it.

2 Make small extensions

As a little exercise to get warmed up, make some changes to the code. For example:

- change the name of a location to something different.
- change the exits – pick a room that currently is to the west of another room and put it to the north
- add a room (or two, or three, ...)

These and similar exercises should get you familiar with the game.

3 Design Your Game

First, you should decide what the goal of your game is. It should be something along the lines of: You have to find some items and take them to a certain room (or a certain person?). Then you can get another item. If you take that to another room, you win.

For example: *You are at Monash University, Peninsula Campus. You have to find out where your lab class is. To find this, you have to find the front office and ask. At the end, you need to find the exam room. If you get there on time, and you have found your textbook somewhere along the way, and you have also been to the tutorial class, then you win. And if you've been to the Seahorse Tavern more than five times during the game, your exam mark halves.*

Or: *You are lost in a dungeon. You meet a dwarf. If you find something to eat that you can give to the dwarf, then the dwarf tells you where to find a magic wand. If you use the magic wand in the big cave, the exit opens, you get out and win.*

It can be anything, really. Think about the scenery you want to use (a dungeon, a city, a building, etc) and decide what your locations (rooms) are. Make it interesting, but don't make it too complicated. (I would suggest no more than 15 rooms.)

Put objects in the scenery, maybe people, monsters, etc. Decide what task the player has to master.

4 Implement the Game

Decide what classes you need to implement the game, then implement and test them.

5 Levels

The **base functionality** that you have to implement is:

- The game has several locations/rooms.
- The player can walk through the locations. (This was already implemented in the code you were given.)

- There are items in some rooms. Every room can hold any number of items. Some items can be picked up by the player, others can't.
- The player can carry some items with him. Every item has a weight. The player can carry items only up to a certain total weight.
- The player can win. There has to be some situation that is recognised as the end of the game where the player is informed that he/she has won.
- Implement a command "back" that takes you back to the last room you've been in.
- Add at least four new commands (in addition to those that were present in the code you got from us).

Challenge tasks:

- Add characters to your game. Characters are people or animals or monsters – anything that moves, really. Characters are also in rooms (like the player and the items). Unlike items, characters can move around by themselves.
- Extend the parser to recognise three-word commands. You could, for example, have a command

  ```
  give bread dwarf
  ```

  to give some bread (which you are carrying) to the dwarf.
- Add a magic transporter room – every time you enter it you are transported to a random room in your game.
- Others. You can invent additional challenge tasks yourself. You have to discuss those with your tutor and get his/her approval before you implement them. Your tutor will advise you if you have picked something that is too difficult or too much work.

6 Submission and Assessment

You have to submit the BlueJ project on a floppy disk. All code must be professionally written (comments and indentation!) and will be marked for

- correctness
- appropriate use of language constructs
- style (commenting, indentation, etc.)
- difficulty (extra marks for difficult extensions)

You also have to submit a report that includes

- the name and a short description of your game
- the description should include at least a user level description (what does the game do?) and a brief implementation description (what are important implementation features?)
- special features of your game
- known bugs or problems (Note: for a bug in your code that you document yourself, you may not lose many marks – maybe none, if it is in a challenge task. For bugs that we find that you did not document you will probably lose marks.)
- a printout of the source of all classes

A perfect implementation of the base tasks can get a distinction (D). To get a HD you need to implement one or more challenge tasks.

The assignment must be handed in to your tutor **at the beginning** of your tutorial class in week 5. **Late submissions will not be accepted!** If you, for any reason, cannot be present at the start of your class, you have to hand your assignment in **earlier**!

The interview

Your work will be assessed during an individual interview. You have to schedule an interview with your tutor (your tutor will let you know the details). It is your responsibility to schedule an interview. Your submission will not be marked without an interview.

You are expected to have written all the code yourself (everything else is plagiarism!) and to be able to explain all of the code you have written in detail. Your mark for this assignment will reflect your understanding of the code that you demonstrate in the interview.