SYSC 2004 Object-Oriented Software Development

Assignment #1

Before you start read the Assignment 1 rubric and watch the Assignment 1 Video provided with the assignment.

You will need `zuul.zip` provided with the assignment. Zuul is a simple text-based adventure game. Experiment with the Zuul game to ensure that you understand what it does. (Read the comments at the top of the "Game" class for hints.) This assignment involves extending the game by adding new functionality.

- Part 1: Two new commands: "look" and "eat".
    - Add the "look" command to the Zuul game. This involves three steps:
        1. Add the "look" command to the list of valid commands in the "CommandWords" class.
        2. Add a "look" method to the "Game" class. This is a void method with one parameter, the command. All it does is output the detailed description of the current room, see "Room" method "getLongDescription".
        3. Update "Game" method "processCommand" to deal with the "look" command. The "look" command should just invoke the "look" method that you added in the previous step.
    - Now add an "eat" command to the Zuul game. The "eat" command should output a message to say that you have eaten and are no longer hungry. The steps to add the "eat" command are similar to those for the "look" command (above).
- Part 2: Commands output by "Game" for design flexibility (e.g. adding a GUI in a future release).
    - Change method "showAll" in class "CommandWords" to "getCommandList" and have it return a String instead of printing the commands.
    - This also requires a change in class "Parser". Method "showCommands" should become "getCommands", and again return a String.
    - Finally, this requires a change in class "Game" in method "printHelp". (This method will now print the information.)
- Part 3: Rooms contain items.
    - Each Item is made up of a description (String) and weight (double).
    - Create an Item class. All you need is a constructor and a method that returns a string representation of the item (containing its description and weight).

    - Each room may contain any number of items, so these will be stored in an ArrayList.
    - You will need a method to add an item object to a room.
    - The information about each item in the room is part of the room's (long) description.
    - Update the "createRooms" method in "Game" to create some items and add them to some rooms so that you can test your code.

- Part 4: Another new command: "back"
    - The "back" command can have only one word. If there is a second word in the command, just print "Back what?".
    - The "back" command takes you to the previous room that you visited. (The one you were in immediately before this one.) Print a message to say that you have gone back.
    - If you are at the beginning of the game, you cannot go back, so just print a message explaining this.
    - You will need to make various changes to implement this command correctly.
    - Hint: If you invoke the "back" command twice in a row, where will you be?

- Part 5: "look" and "eat" are one-word commands.
    - Check that commands "look" and "eat" output "Look what?" and "Eat what?" if there is a second word in the command.

- Part 6: Another new command "stackBack"
    - "stackBack" is a one-word command.
    - "stackBack" allows you to go back one room at a time as far as you like (until you reach the beginning of the game).
    - If you are at the beginning of the game, you cannot go "stackBack", so just print a message explaining this.
    - You will need to use a Stack (see the Java documentation for information on this class).
    - Again, you will need to make various changes to implement this command correctly.
    - Hint: If you invoke the "stackBack" command twice in a row, where will you be? (The answer will normally be different from that of part 4.)
    - Hint: If you invoke "stackBack" enough times, you will eventually return to the beginning of the game.

- Part 7: Tidying up.

- Double check all the files that you added or edited (i.e. "CommandWords.java", "Game.java", "Item.java", "Parser.java", and "Room.java") for good programming style and comments.
- Notes:
  - You should not have changed "Command.java", and **you will <u>not</u> submit that file**. If you did change "Command.java" change it back to the original and check that your code still works.
  - Double-check your code for proper formatting and indentation.
  - Make sure that you have sufficient documentation, both Javadoc comments and regular comments that will help us understand your code.
  - Generate your Javadoc documentation to check that it looks good.

Submission

Ensure that you have **<u>not</u>** changed `Command.java`, and do not submit it!

Submit files: `CommandWords.java, Game.java, Item.java, Parser.java,` and `Room.java`.

Reminders

- You may submit as many times as you like before the deadline.
- All submissions must include all files.
- Only the latest (most recent) submission will be marked.
