SYSC 2004 Object-0riented Software Development

Assignment #4

Use the sample solution to Assignment #3 as your starting point for this assignment. Download file `zuul.zip` extract the files and include the new and updated files from the sample solution to Assignment #3.

Alternatively, if you are 100% sure that you had no issues with Assignment #3 (check the sample solution!!) you may use your solution to Assignment #3 as your starting point. **Note:** If there are errors in your solution to Assignment #3, you will likely lose marks for those errors in Assignment #4, even if they weren't pointed out by the TA who marked Assignment #3. Thus the recommendation to use the sample solution for Assignment #3 as your starting point.

Part 1

1. Modify your `Item` class so that it has a field called `name`. This will allow us to refer to an item with a shorter name than the description. *Do not refactor your game to include a `Player` class*. Change your game so that the player can carry a single item. This includes implementing two new commands: "take" and "drop". "take" is a two-word command. The second word specifies the item that the player is to pick up; e,g., "take book". "drop" is a one-word command, which causes the player to drop whatever item he or she is carrying. Make sure that the commands output their result (e.g. "You picked up book.", "You have dropped book.", "That item is not in the room.", "You have nothing to drop.", "You are already holding something.", etc.). Also, when a room is entered and/or when the player enters "look" the information printed must include whether or not the player is carrying anything, and, if so, what it is.
2. Change the `createRooms()` method in `Game` so that some of the items in some of the rooms are cookies. Change the "eat" command as follows: if the item that the player is carrying is a cookie, print a message stating that the player has eaten the cookie; otherwise, print a message stating that the player has no food. (If you don't like cookies, you may choose another type of food!)
3. Change the game as follows: after starting the game, the player must find a cookie, pick it up and eat it before he or she is permitted to pick up any other items. After eating a cookie, the player is able to pick up any 5 items, one at a time, before he or she becomes hungry. (Remember the player can carry only one item at a time). Then, if the player is not currently carrying a cookie, he or she must then find, pick up and eat a cookie before he or she can pick up another 5 items.

Part 2

Add a *beamer* to the game. A beamer is a device that can be *charged* and *fired*. When you charge the beamer, it memorizes the current room. When you fire the beamer, it transports you immediately back to the room it was charged in. Develop a `Beamer` class as a subclass of `Item`. You'll need to add "charge" and "fire" commands to the game. Note that the player must be carrying the beamer in order to charge and fire it. **The beamer can be charged only if it isn't already charged. In other words, after command "charge", there must be a command "fire" before command "charge" will succeed again.** Change the `createRooms()` method in `Game` so that it creates two beamers, and places each one in a different room.

Part 3

Add a transporter room. Whenever the player *leaves* this room by typing the "go" command, he or she is randomly transported into one of the other rooms in the game. The room the player is transported to is not necessarily one of the rooms that is adjacent to the transporter room; in other words, it's not necessarily one of the rooms that can be reached via one of the exits from the transporter room.

The transporter room must be implemented as a subclass of `Room`. We suggest that the `TransporterRoom` class should override `getExit()`:

```
public class TransporterRoom extends Room
{
    ...

    /**
     * Returns a random room, independent of the direction parameter.
     *
     * @param direction Ignored.
     * @return A randomly selected room.
     */
    public Room getExit(String direction)
    {
        return findRandomRoom();
    }

    /**
     * Choose a random room.
     *
     * @return The room we end up in upon leaving this one.
     */
    private Room findRandomRoom()
    {
        // Implementation omitted.
    }
}
```

It's up to you to come up with a good design for `findRandomRoom()`. Hint: One good way involves adding a static field to the `Room` class.

Change the `createRooms()` method in `Game` so that it creates a transporter room that is connected to (at least) one other room.

Submission

Note that you should not have changed the `Command` or `Parser` classes, and **you will not submit them**. If you changed either of those classes, change them back and check that your code still works.

Submit files "Beamer.java", "CommandWords.java", "Game.java", "Item.java", "Room.java", and "TransporterRoom.java" from your *zuul* project using the SYSC 2004 submit program for Assignment #4. Late submissions will **not** be accepted by the submit program or the instructors!
