Assignment 3: Haunted House

For this assignment, you will be building a class, creating objects, doing comparisons and using nested control statements. You need to remember what you have learned in class, lab, books and your assignments. Be sure to refer to them when you need to.

There are 2 parts to this assignment. In the first part, you are going to be given a problem and you will then need to create a structure, write algorithms and a flow chart to solve it. In the second part, you’ll be turning this into a java program.

So let’s get started!

**Part 1: Your haunted house adventure!**

Ever since you were a little kid, there has been this one house at the end of the street that is dark and scary. Everyone says that its haunted! Of course, you don’t believe in those things, but it has inspired you, so you’ve decided to create an interactive haunted house adventure for your friends. Here’s the layout of the house:

Master Bedroom

Stairs

Bedroom 1

Bedroom 2

Bathroom

Master Bathroom

Living Room

Stairs

Dining Room

Kitchen

Pantry

Bathroom

Front Door

Up Stairs (2nd Floor)

Down Stairs (1st Floor)

The user *always starts the game at the front door* and must immediately decide where they want to go. Your movement constraints are as follows:

* From the *front door*, the user must choose between going into the *living* *room, dining room* or *up the stairs*.
* To get from one room to the next, there must be *a door*. That means that to get to some rooms, the user must go through other rooms. See the above diagram for more details.
* If the user is in a room that is attached to another room via a door (not the one they just came in—no backtracking![[1]](#footnote-1)), the user must be given the option between going to the other room or exploring an item in the current room.
* If the user is in a room that has no other exit, then they must be given the option between exploring the items in the room.

Here are the possible outcomes for exploring items in each room:[[2]](#footnote-2)

|  |  |  |  |
| --- | --- | --- | --- |
| Floor | Room | Item(s) | Outcome |
| 1 | Living Room | Chest | Ghost escapes and scares you to death |
| 1 | Dining Room | Candelabra | Light up by themselves and see a death shadow |
| 1 | Kitchen | Refrigerator | Open it and find some delicious soul food |
| 1 | Kitchen | Cabinet | The dishes and glasses start flying at you as soon as you open the door. You get hit in the head and feel yourself start moving towards a light |
| 1 | Pantry | Dusty recipe box | You open it up and a recipe for chocolate devils food cake appears our of no where |
| 1 | Pantry | Broom | Flies up in the air as soon as you touch it |
| 1 | Bathroom | Mirror | See a bloody face looking back at you |
| 1 | Bathroom | Shower | Room suddenly steams up and you feel fingers touching the back of your neck |
| 2 | Bedroom 1 | Rocking Chair | Chair starts rocking by itself with no one in it |
| 2 | Bedroom 1 | Window | See a child outside on a swing who suddenly disappears |
| 2 | Bedroom 2 | Doll House | The dolls start dancing on their own |
| 2 | Bedroom 2 | Dresser | A ghost flies out of the dresser as soon as you open it and goes right though your body |
| 2 | Master Bedroom | Jewelry Box | You find the cursed Hope Diamond and feel your doom |
| 2 | Master Bathroom | Intricate Oil Lamp | Rub the lamp and a genie pops out who says he’ll grant you 3 wishes |
| 2 | Master Bathroom | Shower | Suddenly hear singing in the shower, but no one is there |
| 2 | Bathroom | Mirror | See a bloody face looking back at you |
| 2 | Bathroom | Shower | Room suddenly steams up and you feel fingers touching the back of your neck |

**Program Flow:**

* Ask the user to enter their name so that you can personalize their experience. You will want to use their name as they move through the house and make decisions
* Start at the front door as described above.
* Each time the user moves to a new room, you must then ask the user what he or she wants to do next. The options available are derived above. Note that there are sometimes more that 2 options available.
* Should the user reach a room where there is no other exit, they must select an object to explore.
* Once the object is explored, the game is over. This should be indicated clearly to the user (and have fun!).

**Input/Output Requirements:**

* As stated above, you must ask the user for his or her name.
* Welcome the user to the game. Be sure to include their name in your welcome message.
* Using ascii art or graphics, print out an image showing where in the house the user is starting.
* For each step, present the users with their options and ask them what they want to do (hint: put word options in quotes to indicate what they should type in to respond to your question)
* Once they have selected an object to explore, be sure to print out their final outcome.
* At the end of the game, using ascii art or graphics, print out an image showing where in the house the user ended the game.

*For Part 1*, create a *class structure, algorithms* and *flow chart* for your program, and then do several iterations of tests (i.e., analyze it and step through to make sure that it is logically correct). Also write the pseudocode for your *tester class* (where your main will go). Put these in a Word or Open Office document. You’ll turn that document in with the program that you create in Part 2.

**Important! As you are working on this, be sure to break this down into smaller pieces. Take it step-by-step, and don’t try to finish this in one sitting. It will make it MUST easier.**

**Part 2: Creating your Haunted House program**

Once you are done writing and testing your class structure and algorithm, you are ready to start coding!

1. Once again, you first you need to create a project. Here’s a nice tutorial on how to do that in Netbeans. If you are using Dr. Java or Eclipse, just do a quick search on youtube.com and you’ll find lots of candidates.

<http://www.youtube.com/watch?v=ezUHG1cuxkM>

Be sure to give your project a *nice, meaningful name* (and make sure it adheres to Java’s naming conventions).

1. Once you have your shell ready, there are a few things to know before you start translating your algorithm into code
   * At the top of your class file, be sure to include the following:

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// PANTHERID: [PantherID]

// CLASS: COP 2210 – [Semester Year]

// ASSIGNMENT # [#]

// DATE: [Date]

//

// I hereby swear and affirm that this work is solely my own, and not the work

// or the derivative of the work of someone else.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Now start translating your algorithm into java code.
   * Remember to code and then compile frequently. It will make it easier to find any bugs.
   * Also remember that you will need to create a **tester class** (where your main method will reside).
2. Once you get your program running correctly, run through 3 scenarios (we will, of course, be testing your other scenarios, but use these for your output):
   * Finish in the Living Room
   * Finish in the Master Bathroom (choose either option)
   * Finish in the Pantry (choose either option)
3. Here is one more thing to do. *Any input requested from the user and/or output received from the user should be in a window* (see E.1.14 and E.1.15 from lab 1). At this point, you probably have your output going to the console. For your final submission, it needs to go to a window (JOptionPane). Don’t forget any additional libraries that you need to import to do this.

That’s it! *Now you have written your first adventure game*! Of course, you’ll also need to turn it in to Moodle.

**Submission Requirements**

You must upload a zip file to Moodle that includes your **complete source project in Netbeans, ready to load (**We have been very lenient up until now regarding this. *From this assignment on, you* ***will lose points*** *if you do not include your complete project*.), and also contains the output in separate data files, and your Word/Open Office document with your algorithm.

**VERY IMPORTANT:** If you do not provide output in separate, easy to find data files, I will assume that your program does not work on those test cases, and grade accordingly. *Do not embed the output in your source code.*

1. If you want a *challenge*, you may include the ability to backtrack. [↑](#footnote-ref-1)
2. You may, if you wish, add to the number of items available in a room, and/or change the outcome of exploring specific items. However, you cannot decrease the number of options in a room. [↑](#footnote-ref-2)