Advanced Topics in Comp Sci — Project—Elevator

ASSIGNMENT OVERVIEW

In this assignment you'll design and program a simulation of a building with an elevator that delivers guests from one floor to another.

BACKGROUND

It's common to create computer-based models of real world objects and behaviors so that we can study them and learn how to improve on them. A classic challenge is to model a hotel's elevator system.

In this project you'll be writing an `ElevatorDemo.py` program that uses classes that you have created to help implement the demonstration.

PROGRAM SPECIFICATION

Create a Python program `ElevatorDemo.py` that will keep track of and manipulate classes that implement a model of a single-elevator hotel. Your final project will consist of separate files for each class, the `ElevatorDemo.py` program that runs them, and a `README.txt` that documents your Elevator project in detail.

DELIVERABLES

`elevator.zip`

This zipped directory will contain the files as outlined in the specification above.

To submit your assignment for grading, copy `elevator.zip` to your directory in `/home/studentID/forInstructor/` at *crashwhite.polytechnic.org* before the deadline.

ASSIGNMENT NOTES

- Part of the challenge in this assignment is figuring out where to begin. Do you want to start out by working with pencil and paper to design the classes, or do you want to start coding right away? How are the classes going to work together? Do we need a `Hotel` class? Do we need a `Passenger` class and a `Request` class, or can those be abstracted as essentially the same thing (we don't have passengers that don't have a request)? Questions like these will have to be answered at one point or another.

- When writing classes, it's a good idea to jot down what instance variables and methods you think the class should have, and then write the class, all by itself in its own file. Then, in an adjacent window, start up Python in interactive mode, import the module, and start testing it out interactively. Switch back and forth between the two windows, with both of them open so that you can quickly scan both source code and run results..

- During development, use two windows on screen, side-by-side. Use hot-keys to jump back and forth between the two windows, and try to reduce your use of the trackpad/mouse.

- How detailed do you want your demo to be? Will you have an `Elevator` class? (Yes, of course.) Will you have a `Door` class that an elevator object will manipulate, with doors either open or closed? (You might, although I chose to represent that as a boolean variable in the Elevator class

itself.) Will you have a `Button` class do represent the floor requests in the elevator, with the button `on` or `off` when the floor is requested?

- One thing that you'll want to include for every single class you write: a `__str__(self)` method. That method, written by you, will allow you to print out the current state of any object of that class. This is useful for debugging, of course, but it will also be used for printing status messages for the hotel which will appear in your final output demonstrating the operation of the elevator.

- One important thing you'll need to include in your simulation is a *timer*. This timer won't be measuring actual physical time in seconds, but abstract time in your simulation. It takes time for elevators to move from one floor to another, it takes time for doors to open and close, it takes time for passengers to enter an elevator. The amount of time it takes passengers to use the elevator will be a measure of how efficient the system it is.

- Look forward to *refactoring*. Rewriting your program so that it works differently, better, more efficiently, etc. is something you should eagerly anticipate. Consider early versions of your program to be rough drafts of the amazing work that you're going to end up with.

GETTING STARTED

1. Decide which strategy listed above you think you'd like to take in working on this project. Will you start with a drawing? Outlining classes on the whiteboard?

2. Find someone else in the class that you think you might like to work alongside and share ideas with. Oftentimes, in discussing your work with others, you'll identify some problem that needs solving, a problem that you wouldn't otherwise have thought of.

3. Make sure you check with the instructor if you start to run into difficulties. Although some aspects of the project have been specified in this document, there may be additional design decisions that we'll have to take a look at.

4. Reference documents containing working code may be available upon request.

QUESTIONS FOR YOU TO CONSIDER (NOT HAND IN)

1. This is a much more open-ended assignment than many that we do in the this class. Do you find that freedom makes you feel more liberated, or is the lack of structure more overwhelming?

2. What modifications would you need to make to your codebase to install a second elevator in your hotel? Decisions you have made in writing your program might make that relatively easy, or it might necessitate writing a whole new set of code. How valuable is it to consider that kind of possibility even as your writing the code for a single-elevator hotel?

SAMPLE INTERACTIONS

```
Not available at this time.
```