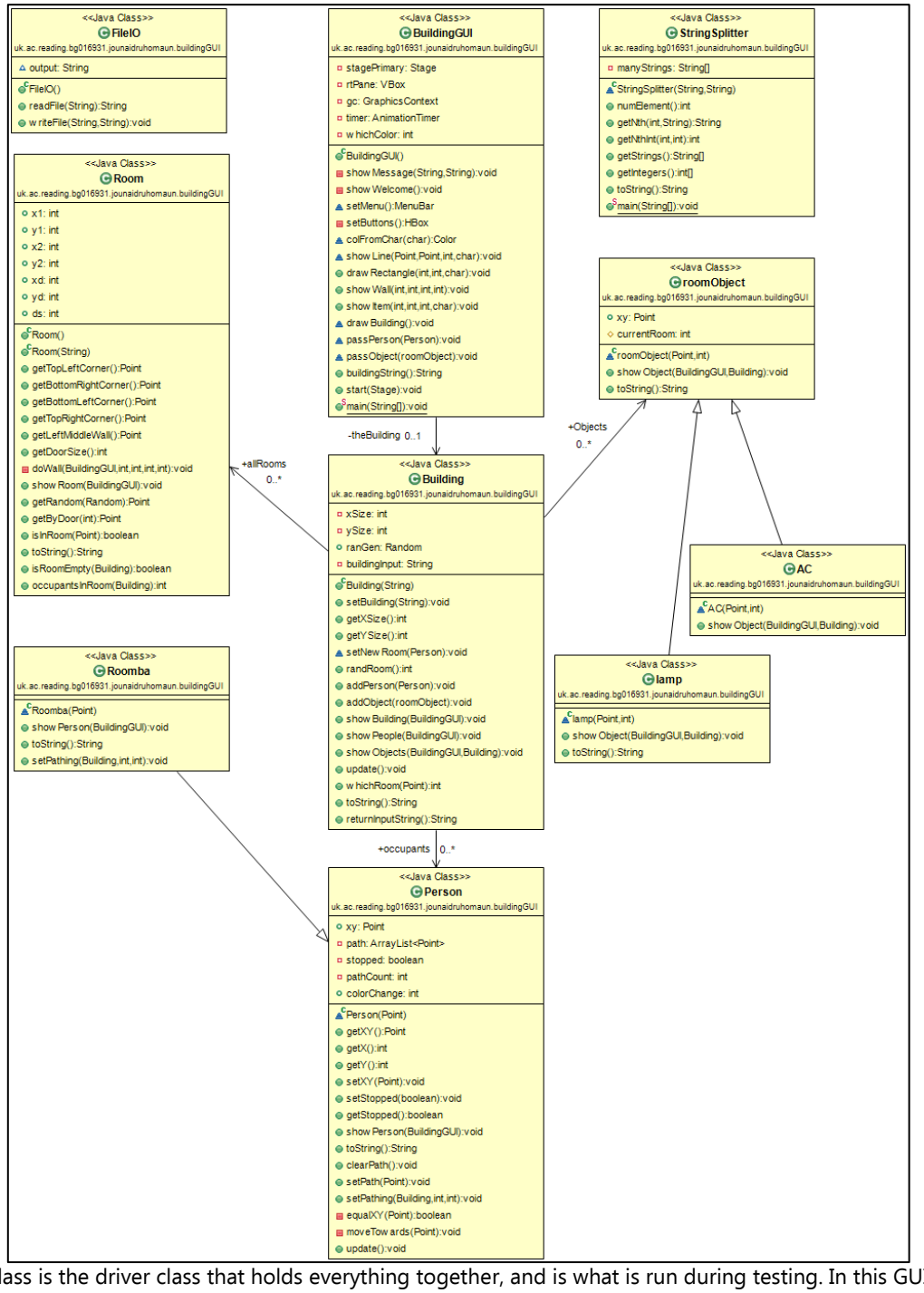
Major Java Coursework #1 (CS2JA16)
Jounaid Ruhomaun – (bg016931)

In this coursework we were tasked with creating a Graphical User Interface for the Smart Building program we developed in the first term. My code is based on Richard Mitchells BuildingGUI program available on blackboard. Classes for Building, Room, Person, StringSplittler and BuildingGUI make up basis of this program, for which I will modify to include the different 'things' that will make up my Smart Building. I will create: A file I/O class to deal with the file handling, A 'Roomba' (automatic hoover) class that will inherit the Person class, a generic parent 'room object' class that different objects can inherit from, an 'air conditioner' class (child of room object) that is placed in a room and turns on when a certain amount of 'occupants' are in the room and finally a lamp class (also child of room object) which turns on if there is at least 1 occupant in the room. Below I will detail the OOP design, include a brief GUI user manual and finally document tests done on the program.

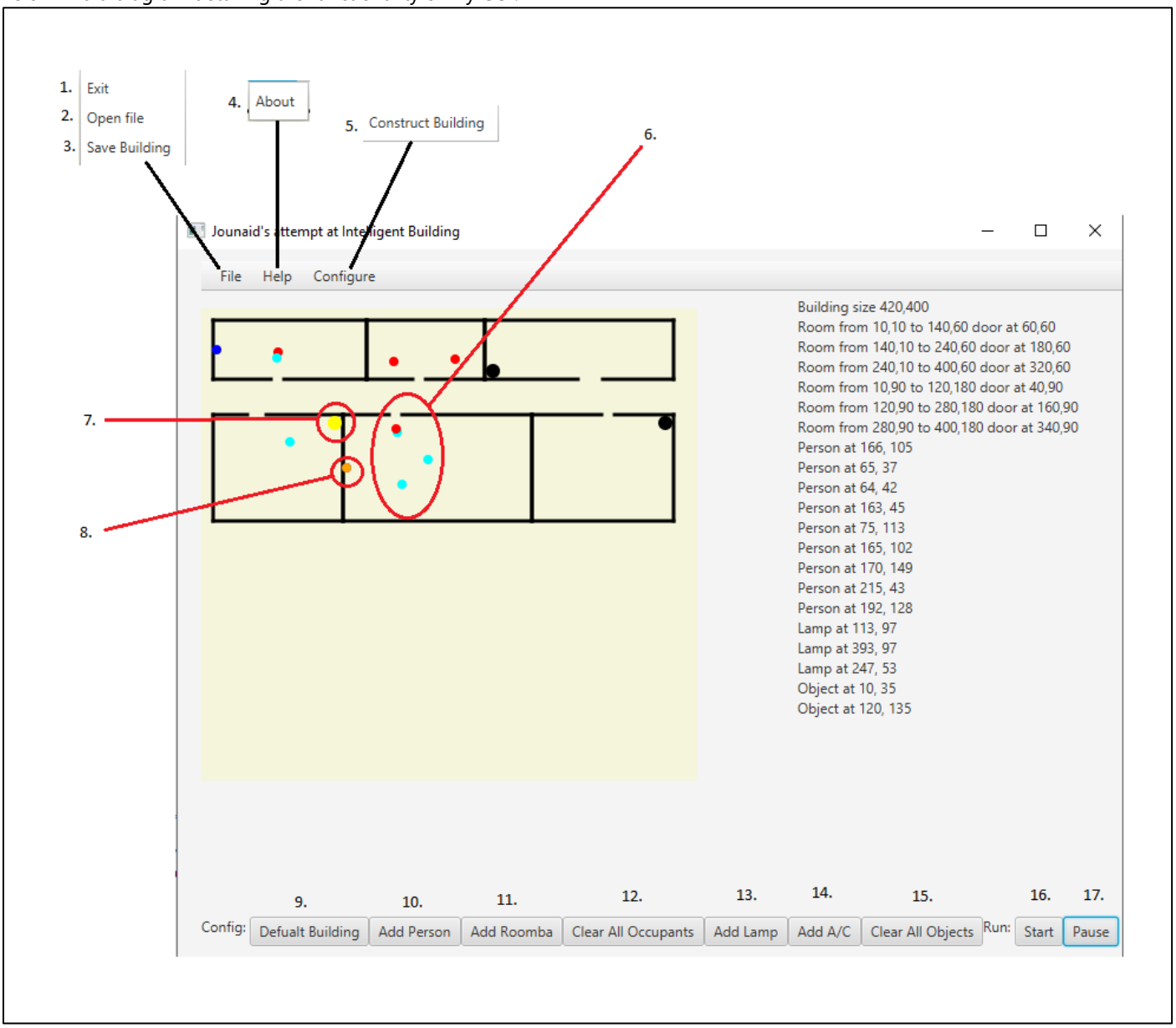
OOP Design:
Below shows a class model representing my program. It provides a static view of the system and shows the associations between classes.



The BuildingGUI class is the driver class that holds everything together, and is what is run during testing. In this GUI, we can instantiate and display one Building class at a time, where each building class can have multiple rooms (stored in an ArrayList). Buildings can also hold roomObject's and Persons, both stored in ArrayLists. The Roomba class is a child of Person, and inherits the abilities to move around the building. The stationary AC and Lamp classes are children of roomObject. The FileIO and StringSplitter classes are for general use and can be called when needed.

GUI User Manual:

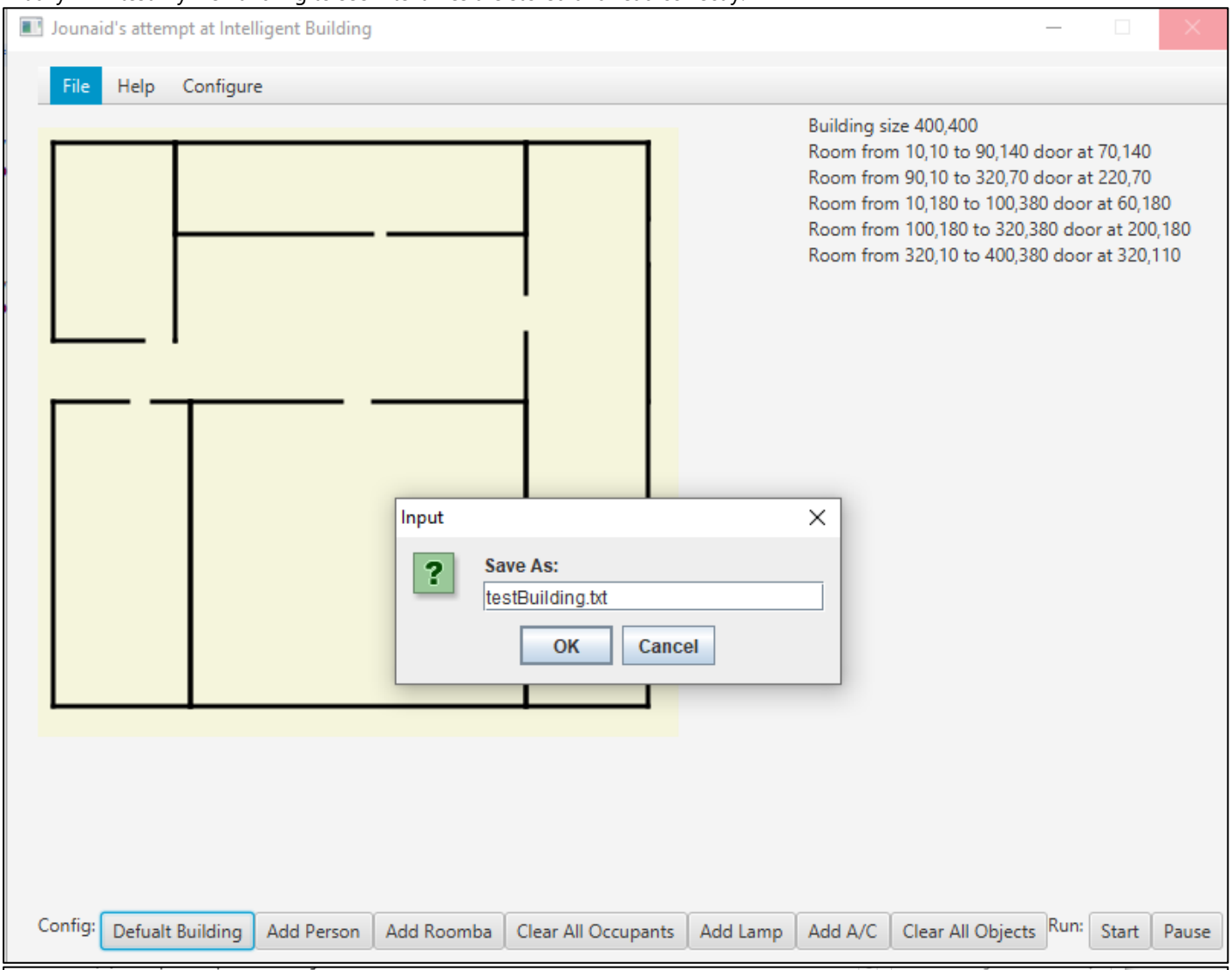
Below find a diagram detailing the functionality of my GUI.

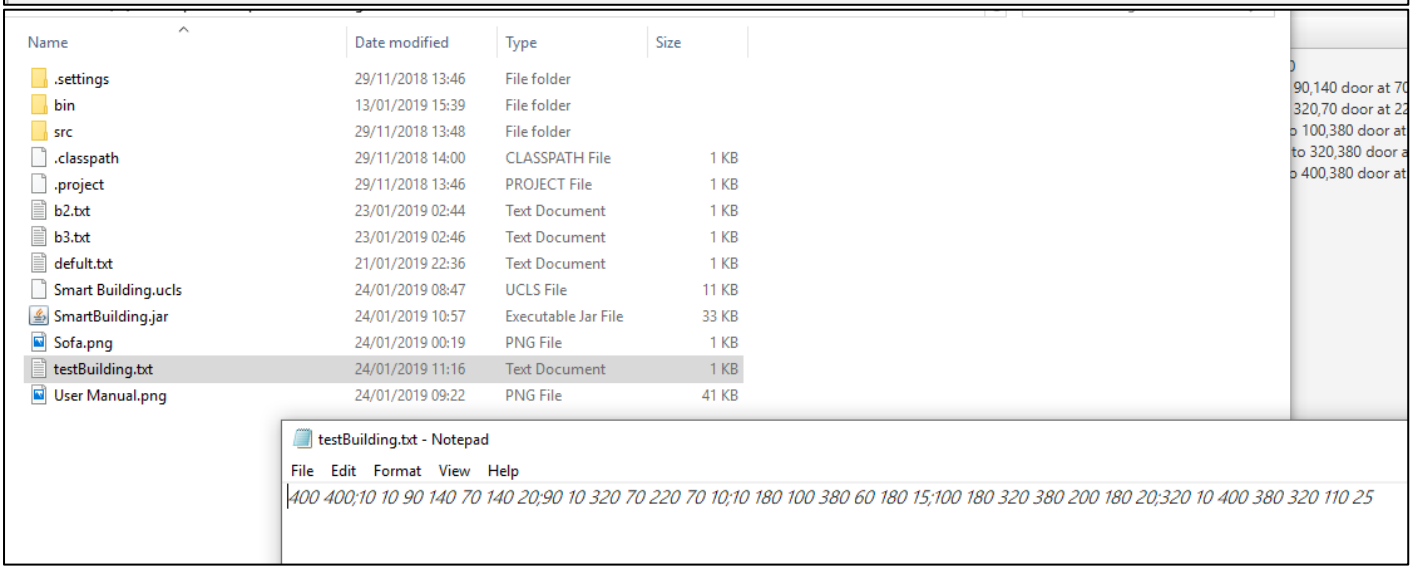


1. Option to exit out of program
2. Option to load building string from file, a dialog box will ask for file name
3. Option to save current building configuration to file
4. Displays an information window
5. Option will open dialog box to input a building string. This building will then be displayed
6. Red and cyan dots represent people moving around the building (male/female)
7. Yellow/black orb's represent lights, which turn on (turn yellow) when an occupant is in the room
8. Blue/orange orb shows A/C. A/C turns on (turn orange) when there are 3 or more people in the room
9. Load the default building file, 'defult.txt'
10. Add a Person object to the building. Person object moves between rooms to a random point
11. Add a Roomba object to the building. Roomba object is locked to its room and moves randomly within it
12. Clear all occupants (moving objects) in the building
13. Add a lamp object to a specified room. Dialog boxes will ask for what room and what corner in that room to place it in
14. Add an A/C to a specified room (input through dialog box). A/C is always in fixed position in room
15. Clears all non-moving objects (lamp/AC)
16. Start the simulator
17. Stop the simulator

Program Testing:

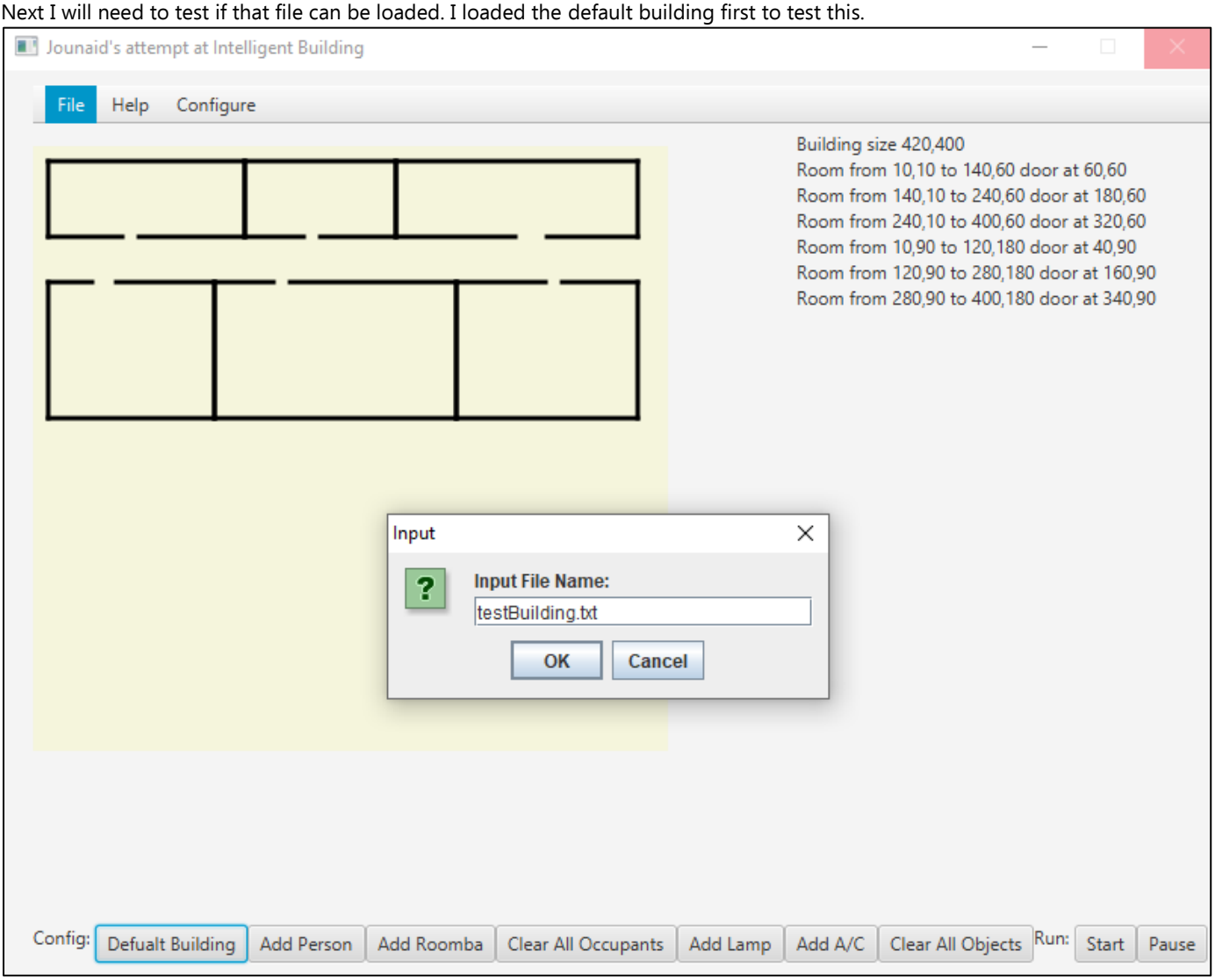
Initially I will test my file handling to see if text files are stored and read correctly.

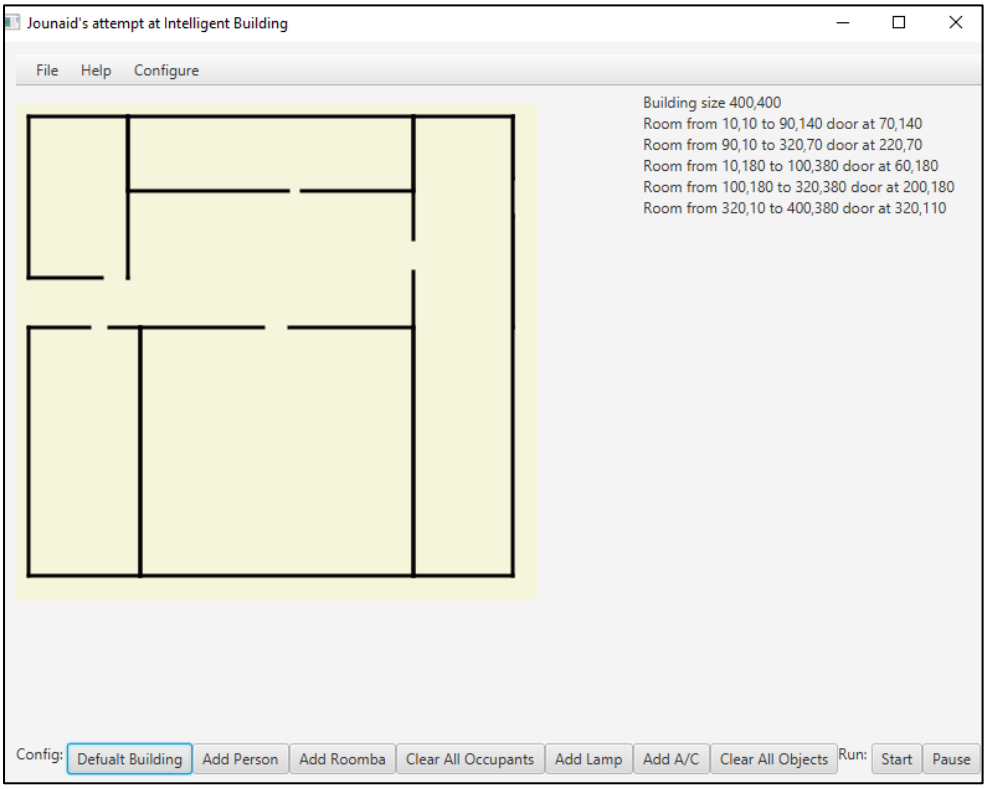




As you can see the save file option is working correctly. The building string was output to the correctly named created txt file.

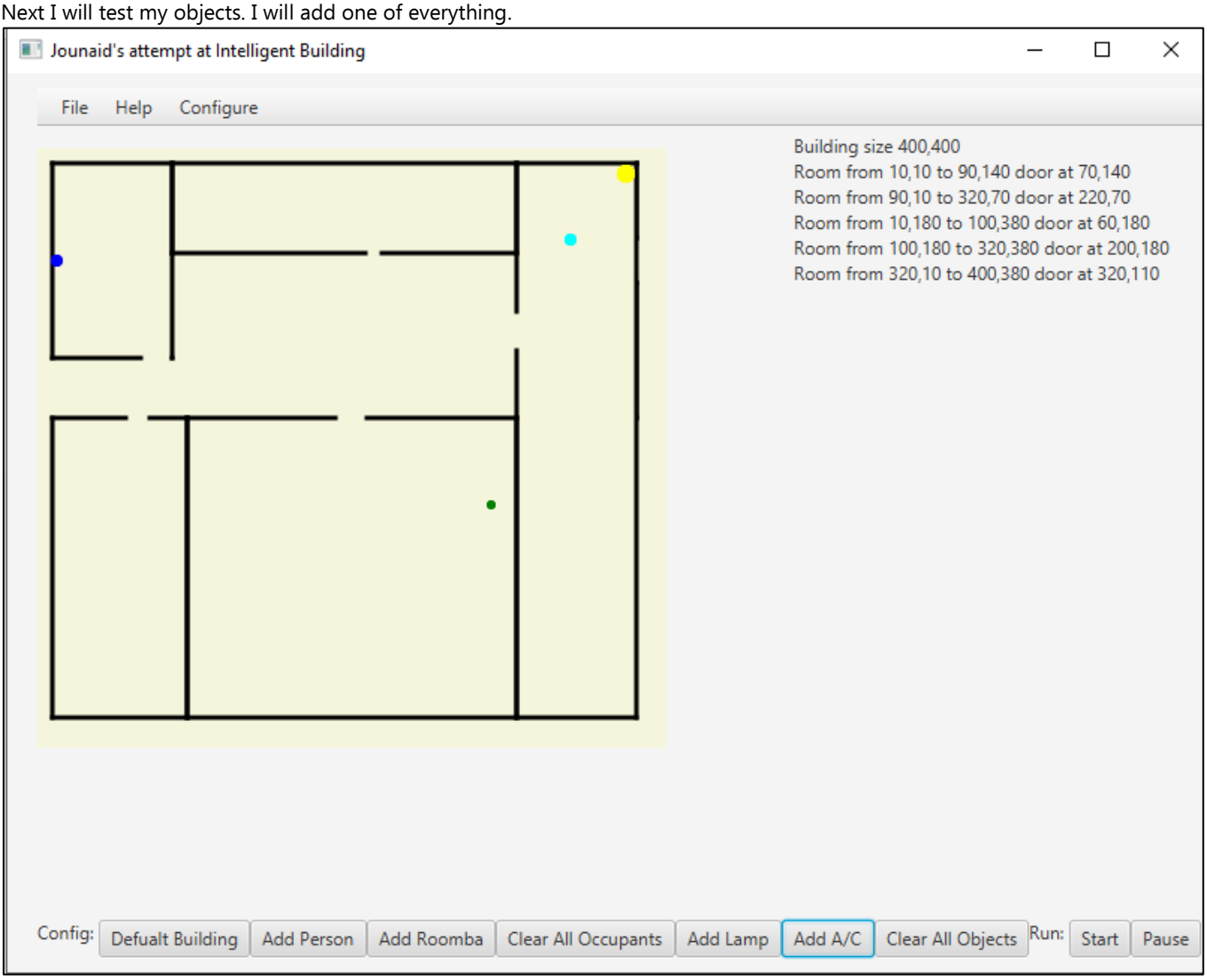
Next I will need to test if that file can be loaded. I loaded the default building first to test this.





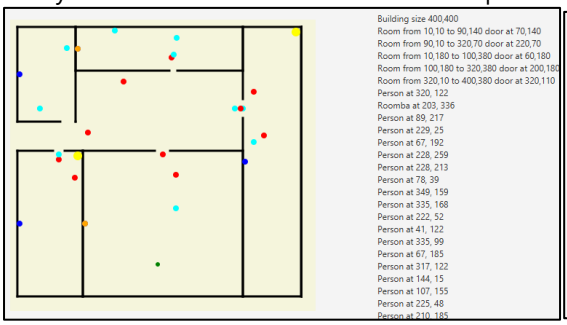
As you can see from the image above and the image to the right, when the correct file name is entered the correct building is loaded from that file.

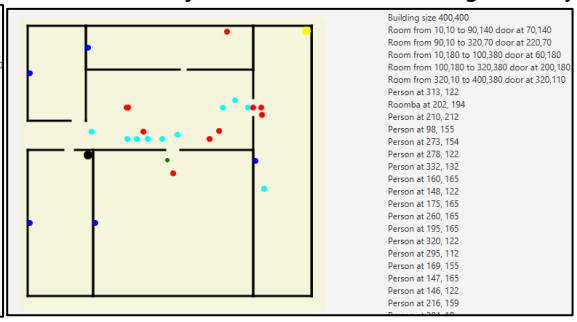
Next I will test my objects. I will add one of everything.

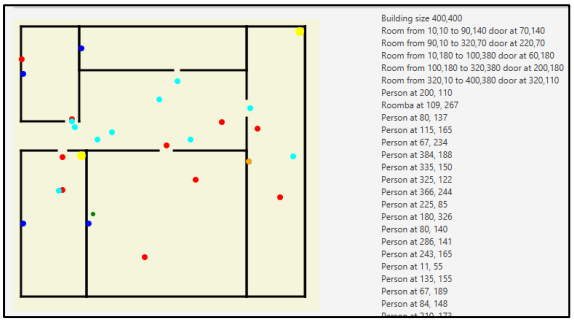


As you can see each object is added correctly.

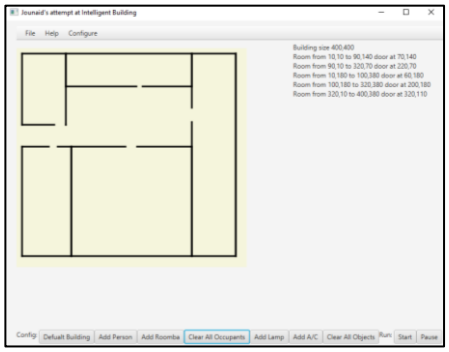
Finally I will test the animation and side description. I will add more objects to show them working correctly.







These three screenshots above and to the left taken consecutively over time shows the animation running correctly, and shows that the side description is accurate and changing. They also show the A/C and Lamp objects turning on and off depending on their conditions.



The final test is to clear the objects, which is shown in the screenshot to the left.

With this test complete I can conclude that everything is working correctly.

Conclusion/Reflection:

To conclude, I believe my project to be successful in demonstrating my Java ability. I used inheritance to extend the functionality of Person and roomObject classes onto Roomba, Lamp and AC respectively. I used '@Overrride' when writing methods in child classes that override their parent methods, for example Roomba has a different setPathing() method than roomObject. Through the tests above I have proven my programs functionality.

In the future I could extend this project by adding more static image objects (inherited classes from roomObject) such as Sofa's, Tables and Chairs. I could also include more different types of People, with different pathing and speeds. I could also add specific rooms such as restrooms where only specific people can enter (either male (red) or female (cyan) people can enter their exclusive restrooms respectively).

NOTICE: A runnable JAR executable is stored with the project files and located in the main BuildingInterfaceGUI folder.