Java Coursework

Abstract

This coursework consists of the design and process of creating an Intelligent Building with a graphical user interface. The program runs within an animation loop with animated items and people, multiple tools to interact with the buildings such as adding more people, looking at a live graph of the building's temperature and many more. The design aspect is looked in detail on how the structure and the things within the building are implemented and how they interact with one and another. A user manual is shown to help navigate around the interface. Many tests have been conducted in order to make the program robust. This has also been documented in this coursework.

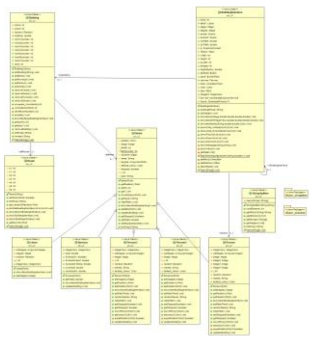
Introduction

The course work as initially broken up into two parts. The first part consisted of creating a basic non-graphical intelligent building, where a person moves from one room to another. Not a lot of freedom was given when programming. However, a string splitter class was provided to help with splitting the string that contained information about the building and rooms. This was done in order to have a solid structure to build upon when implementing the GUI. The major part of course work was to successfully implement a graphical simulator of an Intelligent Building where there are various rooms within the building with multiple items/living things that are responsive to the environment. This had to be done using JavaFX using some of the key concepts of object-oriented programming, encapsulation and inheritance.

Design

Object-oriented programming (OOP) concepts let us create working methods and variables, then re-use all or part of them without compromising security.[1] Using this concept, for this coursework, multiple items have been created which interact with one and another within a room, all inheriting methods from an abstract class, item. In total 10 classes were created to achieve the goal of a lively Intelligent Building. They are called room, stringsplitter, building, buildingInterface, items, person, person2, person3, chest and furnace.

Figure 1. Shows a class diagram of all the classes with in the program and their relationship with each other



HD image: https://imgur.com/a/kPuGnGY

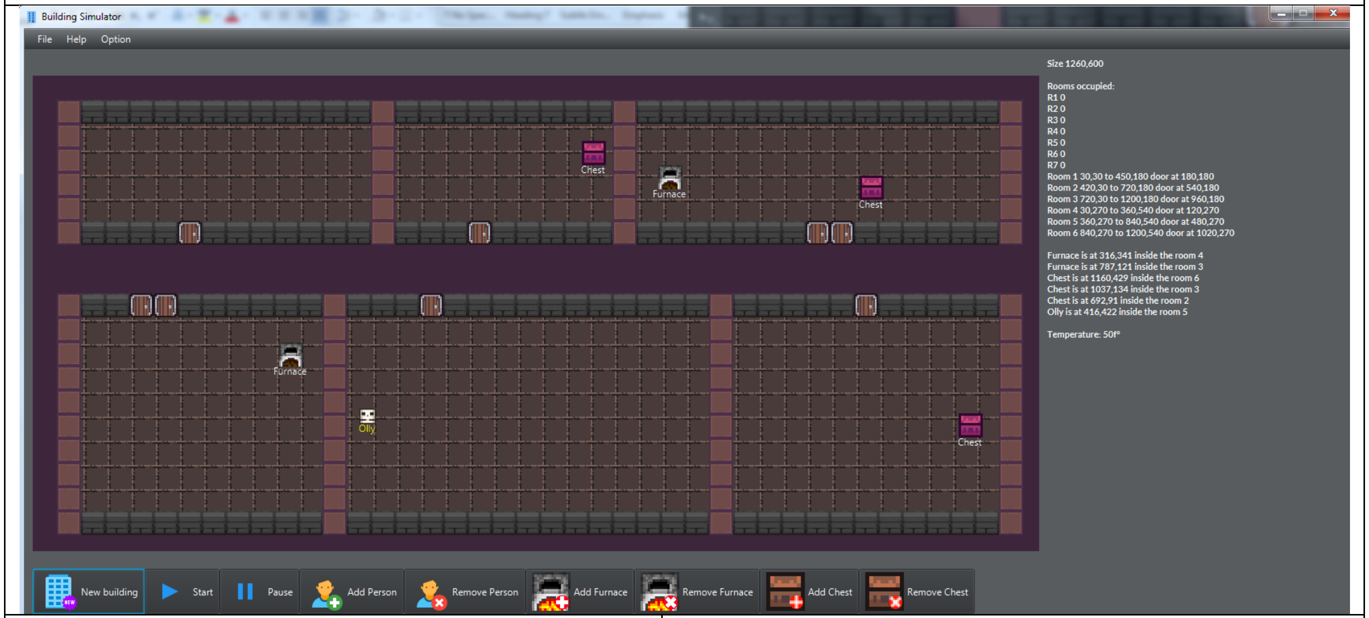
The room class is used to create a room using the information from a given string. The string contains the coordinates of the corner of the room, the opposite location from the corner, the door location and the door size, which is by default 1. It uses another class called Stringsplitter as a helper class to split the string and store them as coordinates for the room. This is a separate class because typically there is more than one room and having its own class for the room acts like an object, which makes it easy to re-use in the future such as when loading a building with lots of rooms.

The building class shown in figure 1, is where the rooms are set up and stored within an ArrayList of rooms. This class does this by using the room class to do create multiple instances of a room within the building, as shown in figure 1 by the arrow extending from the building class to the room class. The interaction between items and update status also take place in the building class, which then updates the animation or information of the item. The status/information of each room and items are stored as a string, which is later on shown in the building interface as a list of building contents. All the items are stored within an array list. Therefore, the building class also uses the item class to update all the items at once.

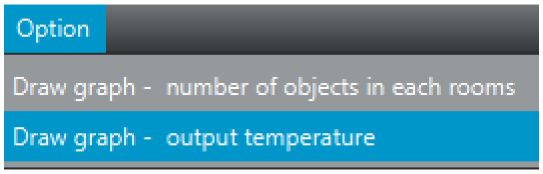
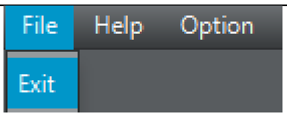
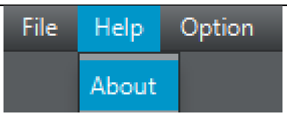
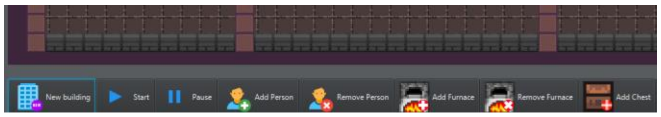
The items class is an abstract class. An abstract class is used to provide a base for subclasses to extend and implement the abstract methods and override or use the implemented methods in abstract class. [2] This is used on top of the hierarchy of items so that no redundant classes would have to be created since the item class defines some common behaviour that can be inherited by multiple subclasses. This is shown in figure 1 on how multiple subclasses of items like a person, person2, chest, furnace are inheriting the common behaviours of the abstract class, item. Again arrows are used to indicate this. The method Super() is used to for this. For example, a subclass such as a furnace has an output temperature and it uses the update function for a different purpose as to a person would.

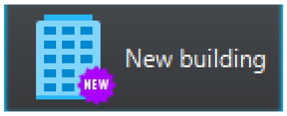
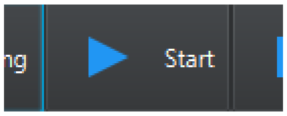
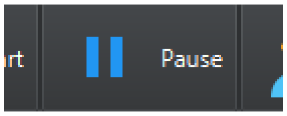
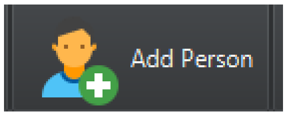
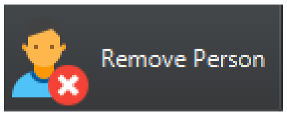
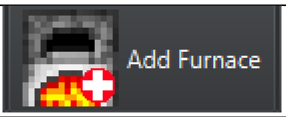
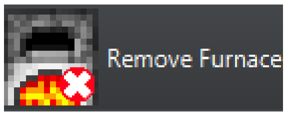
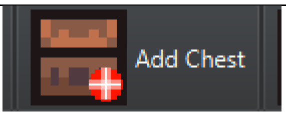
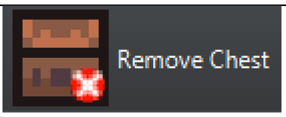
The buildinginterface class is where the GUI is set up by inheriting the application class. It handles and instantiates the buildings, the items, graphs, animation loop, menus, toolbar and window panels. A controller for a GUI is very useful as it is clear where everything is being handled. This class mainly connects to the person and the building class to call a function such as removePerson, addPerson for the toolbar, the building class is used, as shown in figure 1 to instantiate the building size and the rooms and its coordinates within the building.

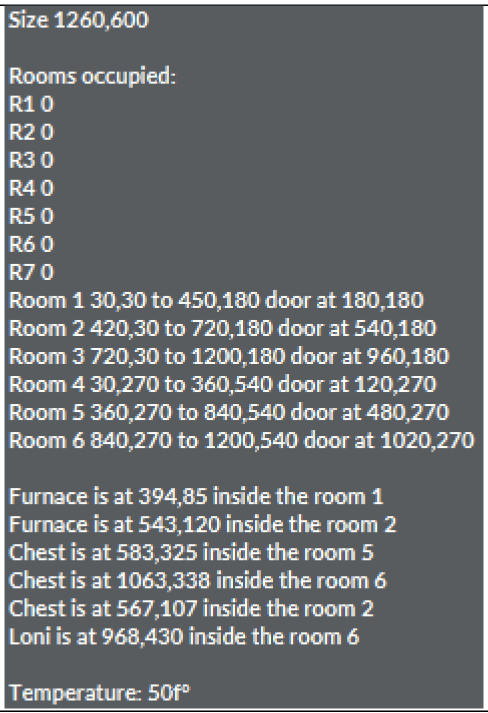
User manual



```
Size 1260,600

Rooms occupied:
R1 0
R2 0
R3 0
R4 0
R5 0
R6 0
R7 0
Room 1 30,30 to 450,180 door at 180,180
Room 2 420,30 to 720,180 door at 540,180
Room 3 720,30 to 1200,180 door at 960,180
Room 4 30,270 to 360,540 door at 120,270
Room 5 360,270 to 840,540 door at 480,270
Room 6 840,270 to 1200,540 door at 1020,270

Furnace is at 316,341 inside the room 4
Furnace is at 787,121 inside the room 3
Chest is at 1160,429 inside the room 6
Chest is at 1037,134 inside the room 3
Chest is at 692,91 inside the room 2
Olly is at 416,422 inside the room 5

Temperature: 50f°
```

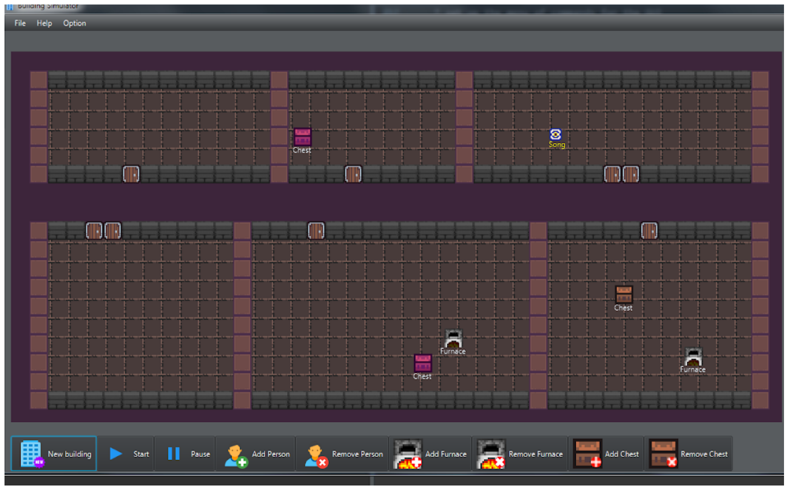
Option menu To create graphs for temperature and room count for the items, click on the 'Option' drop-down menu located on the top left of the program and select the type of chart desired.	 Option Draw graph - number of objects in each rooms Draw graph - output temperature
Exit from the program To exit out of the program, click on the 'File' drop-down menu on the top left of the program and select the 'Exit' menu item.	
Help menu To help about the program, click on the 'Help' drop-down menu located on the top left of the program and select the 'About' menu item.	
Tool bar The tool bar which is located at the bottom of the application will allows the users to interact with the buildings.	 New building Start Pause Add Person Remove Person Add Furnace Remove Furnace Add Chest

New building The new building button loads another building. This button is located on the bottom of the program.	
Start The start button begins the animation timer, for the program to animate. This button is located on the bottom of the program.	
Pause The pause button is used to stop the animation timer, for the program to stop animation. This button is located on the bottom of the program.	
Add person The add person button proceeds to add a random person, within a random room. The random person could be one of three types, normal person, person with different behaviour or a ghost. This is randomly selected and the button is located on the bottom of the program.	
Remove person The remove person button proceeds to remove a person, within a random room. This button is located on the bottom of the program.	
Add Furnace The add furnace button proceeds to add a furnace within a random room.	
Remove Furnace The remove furnace button proceeds to remove a furnace, within a random room. This button is located on the bottom of the program.	
Add Chest The add chest button proceeds to add a chest within a random room.	
Remove Chest The remove chest button proceeds to remove a chest within a random room.	

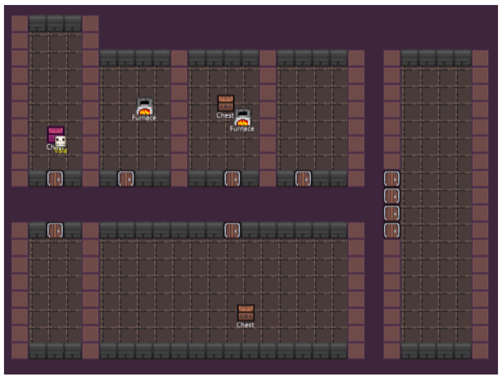
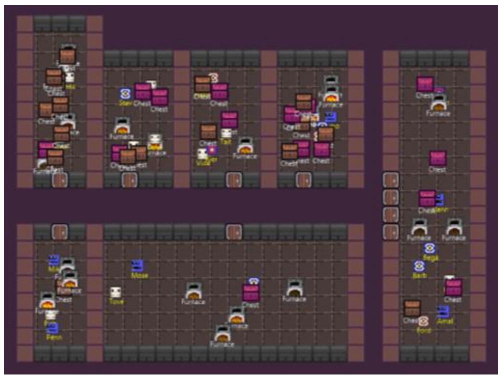
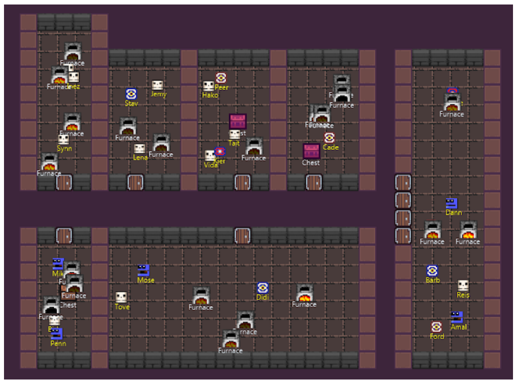
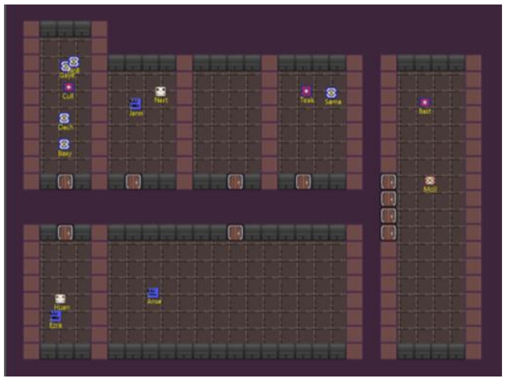
Building content On the right-hand side of the program, contains all the list of content for the building such as the rooms occupied by the items, the rooms positions and its door, the items location and its room location, the temperature of the building.	

```
Size 1260,600

Rooms occupied:
R1 0
R2 0
R3 0
R4 0
R5 0
R6 0
R7 0
Room 1 30,30 to 450,180 door at 180,180
Room 2 420,30 to 720,180 door at 540,180
Room 3 720,30 to 1200,180 door at 960,180
Room 4 30,270 to 360,540 door at 120,270
Room 5 360,270 to 840,540 door at 480,270
Room 6 840,270 to 1200,540 door at 1020,270

Furnace is at 394,85 inside the room 1
Furnace is at 543,120 inside the room 2
Chest is at 583,325 inside the room 5
Chest is at 1063,338 inside the room 6
Chest is at 567,107 inside the room 2
Loni is at 968,430 inside the room 6

Temperature: 50f°
```

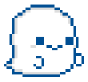
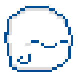
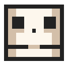
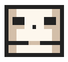
Testing

Test	Expected	Output
Start and compile from a jar file on a different machine	It should load all the content of the building as a graphical user interface from the jar file	As expected, the jar file does work on a different machine. It was able to load all the images successfully.
Output Screenshot		
		

Tests	Expected	Output
• Load new building • Add person • Add chest • Add furnace	Should load a new building with all the content of the building as a graphical user interface. It should also be able to handle adding a new person, chest and furnace.	The output showed no errors, everything was loaded and added to the program. I did not come into any errors.
Output Screenshot		

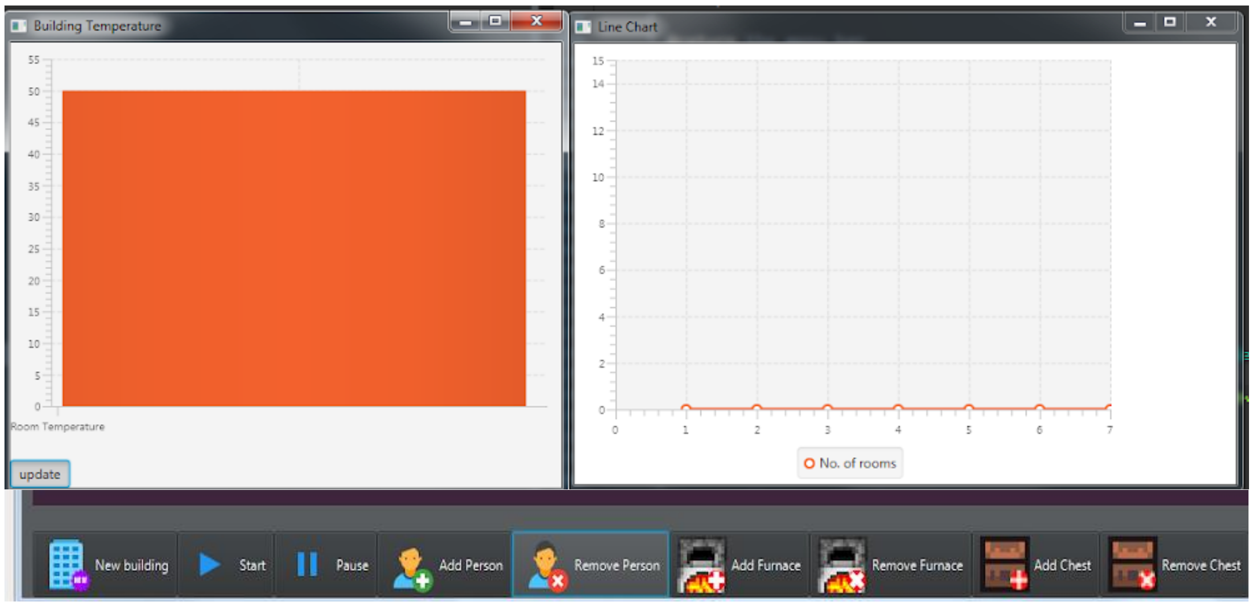
Load new building	Add chests
	
Add furnaces	Add persons
	

Tests	Expected	Output
Start and stop animation	Expected the all the items to animate with there individual frames when the button start has been pressed and stop all animation when the stop button has been pressed.	No lagging function while animated. But when ran on low spec computer, it will start to jitter. There may be a memory resource leak or the animation loop is not letting go of the resource allocated.

Example	Frame 1	Frame 2	Frame 1	Frame 2
				

Tests	Expected	Output
• Show graphs • Remove person • Remove furnace • Remove chest	A person should be removed when clicked on the remove button, similarly with the remove furnace and remove chest. Clicking on draw graphs should draw the graphs of either the number of people in each room or the temperature of the building.	The graph does show all the info requested. One of the graphs, which is to graph the number of items in the room does not update automatically. The animation must be stopped and then clicked on draw graph to update it. The other graphs to show the temperature of the room works automatically once you clicked on the update button. The items being removed had no issues. There was a concurrent exception error when trying to remove furnace using the ghost but it was handled using try and catch.

Screenshot



Conclusion

During the end of the deadline, I ran into multiple issues such that all my sprites and icons got permanently deleted, which had set me back. I luckily had backups for the previous version but I had to redo all the animation framework. Overall this project was rather enjoyable and challenging throughout the course. Initially, not much freedom was given during the first part of the coursework, which was very hard but this was done in order to have a good structure to build upon when implementing the GUI. During this course work, I was able to innovate and think outside the box to create what I had imagined, making sprites and tile mapping was rather enjoyable. Although it is not quite where I would like to finish it, the experience I have learnt from using the methodology of object-oriented was to my satisfaction and I hope to learn and implement it more in the future.

Reference

1. Stackify. (2019). *What Are OOP Concepts in Java? The Four Main OOP Concepts in Java, How They Work, Examples, and More*. [online] Available at: https://stackify.com/oops-concepts-in-java/

2. JournalDev. (2019). *Abstract Class in Java - JournalDev*. [online] Available at: https://www.journaldev.com/1582/abstract-class-in-java