How to Run

Note: this project was completed using Eclipse IDE version 2022-06. As such the tutorial to run this project will assume this IDE is being run.

Files of Note
P1-code.zip. This file holds all the source code, this needs to be unzipped and copied into a Eclipse java project.

Additionally, the build path needs to be configured to include the JavaFX library. Instructions on how to do this can be found here.

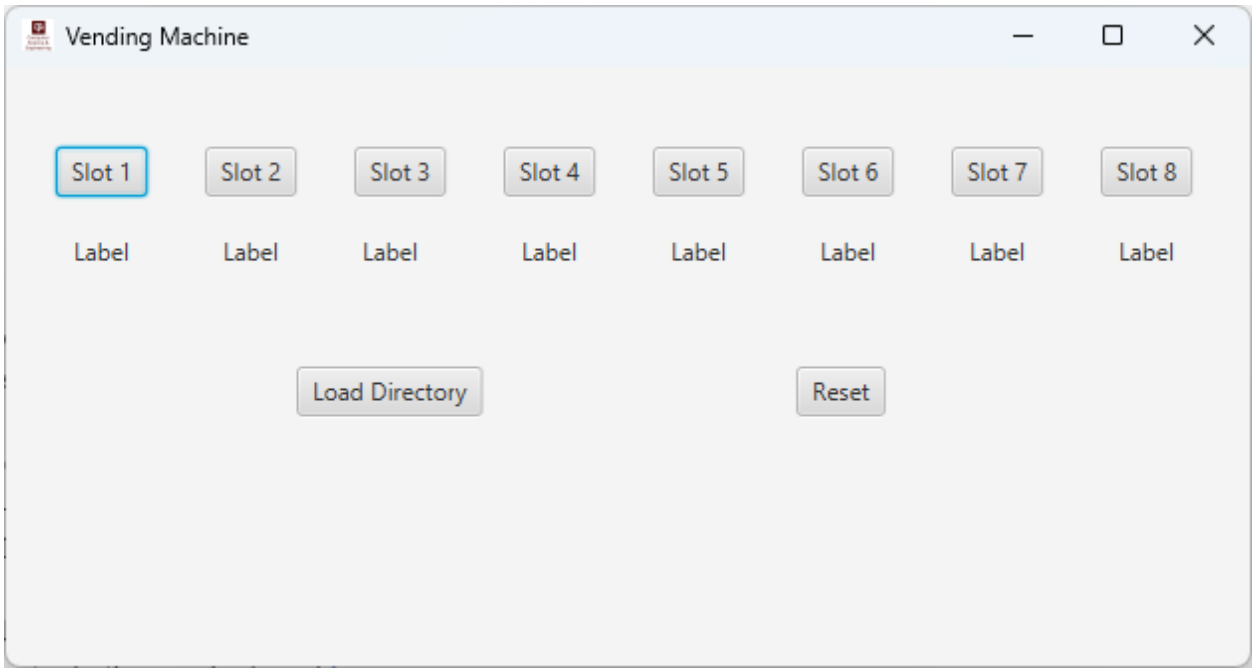
There are three main packages, application, vending, and resources. The resources package only contains image files and does not need to be modified.

The vending package holds all the java code for the vending machine. And the application package holds the JavaFX part.

The non-JavaFX version of this project will run automatically when the command “make vending” is ran in the directory this project is located. 2

How it Looks Running

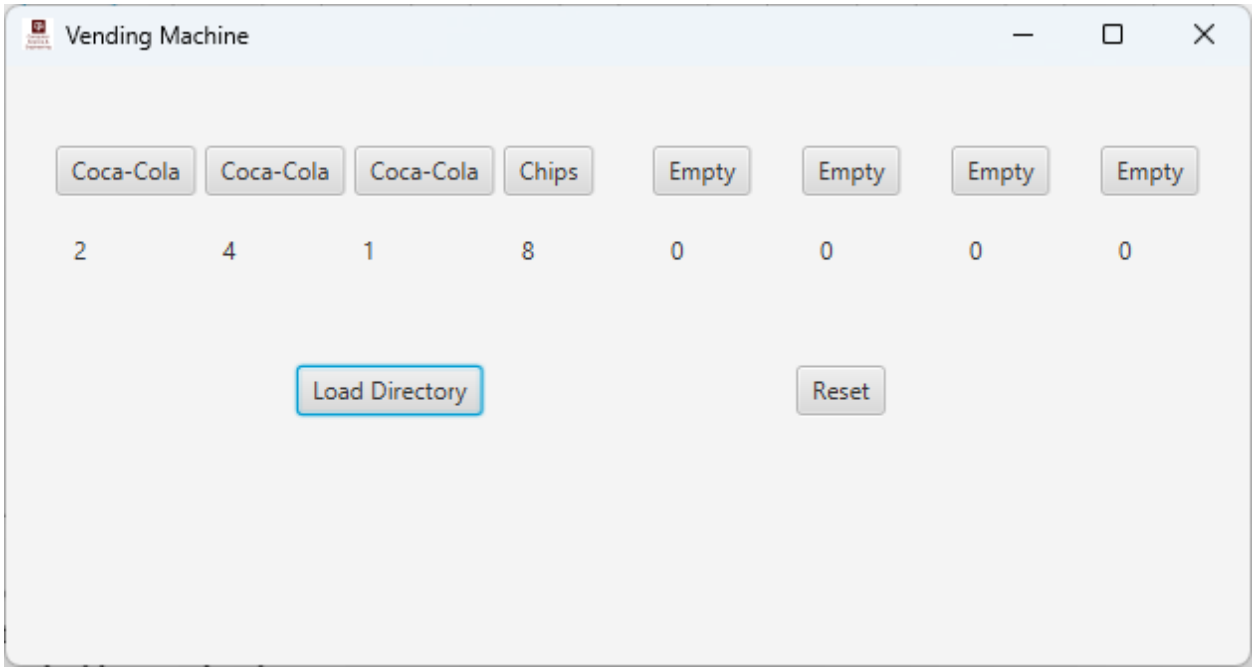
When running Main from Eclipse, the following screen will be displayed.



The eight buttons labeled Slot 1 through Slot 8 are the eight slots of the vending machine. Each button corresponds to removing an item from this slot according to the selection protocol described later in this document.

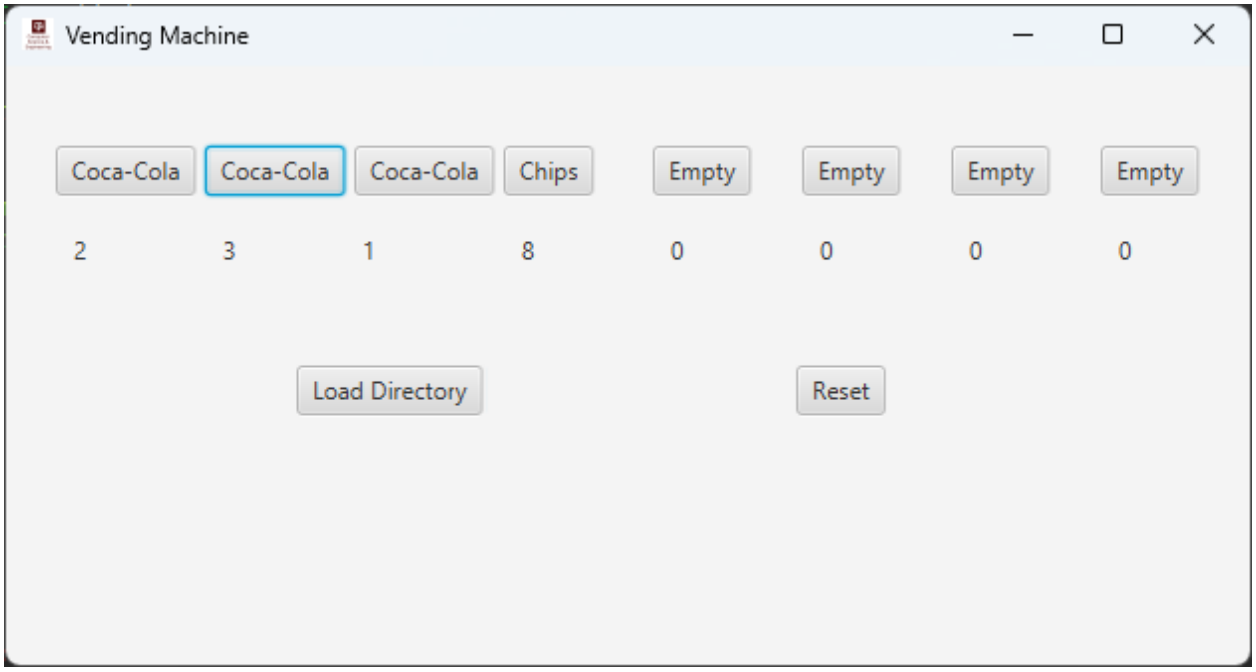
Pressing these buttons will initially do nothing. First, a directory needs to be loaded by clicking the Load Directory button. When pressing, your local file explorer will open and ask for an input directory file.

When a file is selected and parsed by DataFile.java and Vending.java, the GUI will be updated to reflect the contents of the directory. As an example, loading directory1.txt into the GUI will update the GUI as follows.

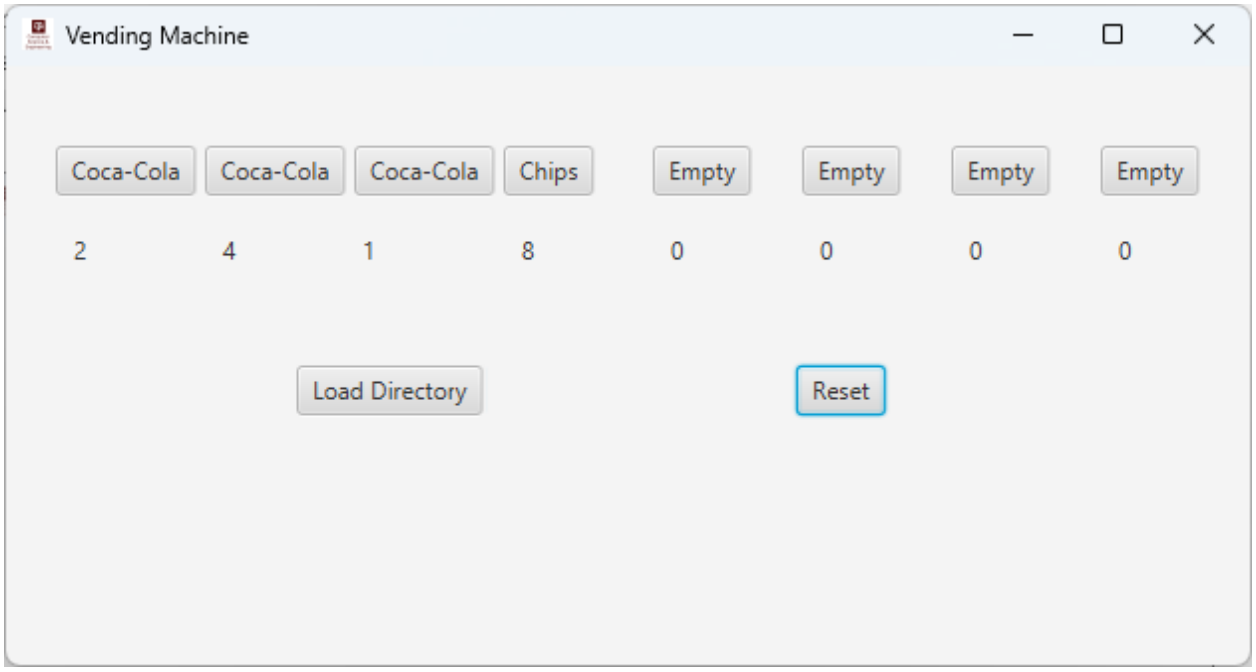


After loading the directory, these slot buttons will reflect the contents indicated from the directory file. If nothing was originally in that slot in the directory file, the corresponding slot in the vending machine will also be empty. The labels below each slot is a label indicating the quantity of the item in the slot.

When a button is pressed, an item is unloaded from the slot using the selection protocol described later in this document. For example, clicking slot 2, will update the GUI as displayed below.



The Reset button will cause the Vending machine to reload items from the directory file selected from the “Load Directory” button. The result of this is displayed in the image below.



For the GUI part, the program is automatically run. Loading a default Directory and test inputs. The file the directory and the inputs are loaded from can be selected in Driver.java displayed below.

```java
/**
 * Change the files names here to change the directory and the input files.
 */
DataFile myData = new DataFile("./Directory/directory1.txt", "./Input/input1.txt");
```

When completed running, the following will be displayed in the terminal.

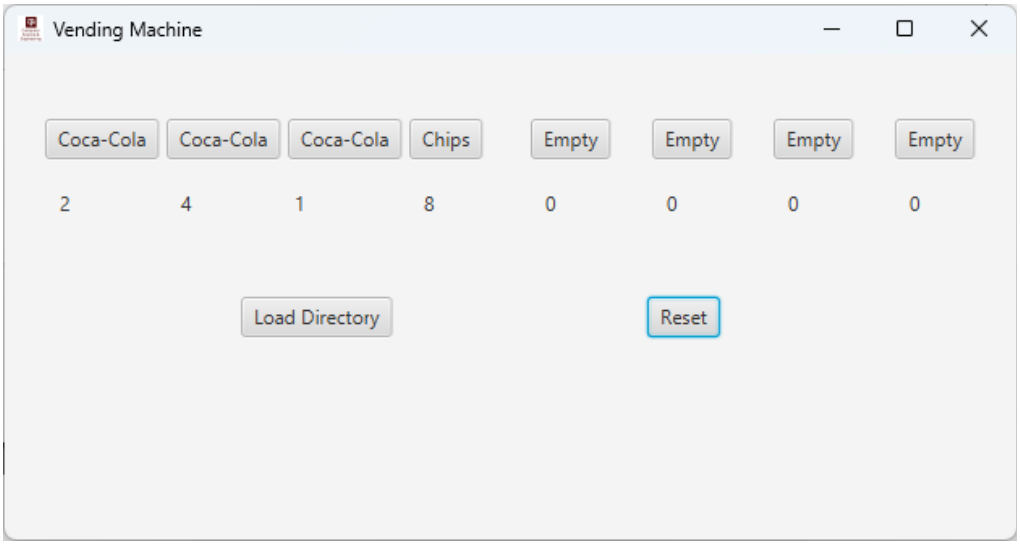
```
pineapple@Pineapple:/mnt/c/MrPineapple/Texas A&M/CSCE 314/ProjectPart2$ make vending
javac --release 11 -d bin/ -sourcepath src/ src/vending/Driver.java
java -cp bin/ vending.Driver
Items originally there:
Coca-Cola: (soda): 2
Coca-Cola: (soda): 4
Coca-Cola: (soda): 1
Chips: (Snack): 8
<empty>
<empty>
<empty>
<empty>


_______________________________
Items removed final count:
Coca-Cola: (soda): 1
Coca-Cola: (soda): 2
Coca-Cola: (soda): 1
Chips: (Snack): 8
<empty>
<empty>
<empty>
<empty>
```

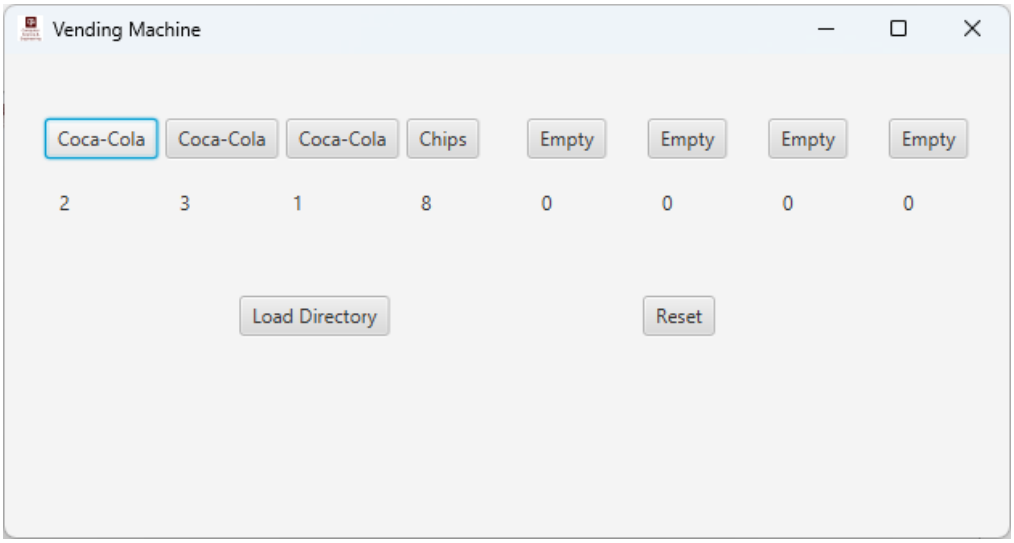
Maintaining Selection Protocol

As mentioned numerous times previously in this document, when selected a slot, the item in that slot is not immediately unloaded. When selecting a slot, the slot with the largest quantity of that item will be unloaded instead. If two slots have the largest quantity, the farthest left slot will be selected.

As an example, loading the same directory as before, the GUI will display the following.



Pressing slot 1, will actually unload from slot 2 and update the GUI accordingly.



Likewise, the final contents of the vending machine will be displayed in the non-JavaFX version of the code.

```
pineapple@Pineapple:/mnt/c/MrPineapple/Texas A&M/CSCE 314/ProjectPart2$ make vending
javac --release 11 -d bin/ -sourcepath src/ src/vending/Driver.java
java -cp bin/ vending.Driver
Items originally there:
Coca-Cola: (soda): 2
Coca-Cola: (soda): 4
Coca-Cola: (soda): 1
Chips: (Snack): 8
<empty>
<empty>
<empty>
<empty>


_____________________________
Items removed final count:
Coca-Cola: (soda): 1
Coca-Cola: (soda): 2
Coca-Cola: (soda): 1
Chips: (Snack): 8
<empty>
<empty>
<empty>
<empty>
```

The actual code behind the selection protocol is displayed as below.

```
public void unloadItem(final int index) {
    if (index ⩾ this.slots.size())
        return;
    final Queue<Item> s = this.slots.get(index);
    if (s.isEmpty())
        return;
    int max = 0, j = 0;
    for (final Integer i : this.findProduct(s.peek().getName())) {
        final Queue<Item> slot = this.slots.get(i.intValue());
        if (slot.size() > max) {
            max = slot.size();
            j = i.intValue();
        }
    }

    this.slots.get(j).poll();
}

public ArrayList<Integer> findProduct(final String product) {
    ArrayList<Integer> index = new ArrayList◇();
    for (int i = 0; i < this.slots.size(); ++i) {
        final Queue<Item> slot = this.slots.get(i);
        if (slot.isEmpty())
            continue;
        if (slot.peek().getName().equals(product))
            index.add(Integer.valueOf(i));
    }
    return index;
}
```