CMPSC 201 Final Project - Final Report

Katie Burgess and Andrew Rankins-Bell

December 8, 2021

1 Introduction

For my final project, I decided to create a combination program between Python and Java, connected through a Bash file. The bash file, when run, executes all the commands necessary to complete all tasks encompassing our project. Our program is based on a grocery store, with 10 randomized products that the user can decide to buy based on how much money they have. This program will also give suggestions to the user to buy items that the user has bought before. These suggestions actually last between sessions and restarting your computer due to the way we programmed it. I defined the problem for this project as: "I want to create a base prototype for an online shopping site".

I had multiple motivations to create this program. I wanted to display the strengths of both Python and Java, showing the benefits of using both languages. I also wanted to become more proficient in Java, as it is a language that is still widely used in the industry today. Lastly, I wanted to become more familiar with creating programs that took in information from and put information into files, specifically csv files. This is because csv files are widely used, especially in research and databases.

1.1 Proposal

When we proposed the project, this is what we initially thought our program would look like:

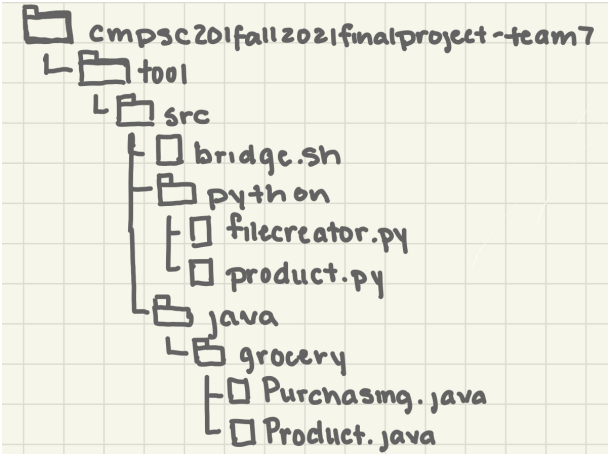
" We chose to do the combination project idea. For this project, we will be emulating a grocery store using Python and Java code, combined using a bash file. We will have three files total, one for each language mentioned. We are currently planning on having this program do tasks such as: say which product is the cheapest, tell you how much it would cost to buy a certain amount of a good, or possibly even tell you how much the prices differ between the most expensive and the cheapest good offered at the grocery store. The Python program will be creating a randomized data file, giving random good names as well as prices using the random library. We will be storing these good names inside of a list and using the randInt function to access a random one. If there are duplicates, the program in Java will be able to name the cheaper one. The Java program will have a 2D list, storing the name of the good as well as its price. You will be able to enter a number to decide what you

want to do with the groceries offered in the randomized grocery store. I think 100 randomized items with around 10 different goods to choose from in the randomization is a good starting point for our program, which we may expand upon if we have time. We will be putting the results in an output file through Java. Java is a compiled language, meaning that it is faster than Java, which is an interpreted language. This is why we are going to be creating the file in Python, which is higher-leveled than Java, giving it more ease of use. Java is faster, which is why it will be using the file Python creates and computing all the data for our project. There is a need for a tool that is able to determine what goods prices are most affordable in this ever changing world of inflation. This tool has the ability to help families that are most in need of resources find the prices that are sustainable. This tool could be combined with bargain hunting to positively impact lives."

Now that the development process is over, there are a lot of things that have changed. Most notably, the 2D list changed into an array of class objects. These changes overall changed the program into a much more OOP-like program that is much more efficient as well.

2 Programming the Project

Through starting early and working diligently on this program, as of writing this progress report the program is completely finished and operational. Our program consists of two outside folders, src and data. Src then contains two folders inside of it, python and java, which hold the files written in each language. Java then contains the grocery folder for the grocery package used in the Java section of the project. The folder structure of the program is illustrated below:



Everything is organized in a way that the folder explains exactly what will be contained inside of it. It makes organizational sense to organize things like this, as you will never be left searching for a file that you need. You will always know exactly where everything is stored.

2.1 Approach and Challenges

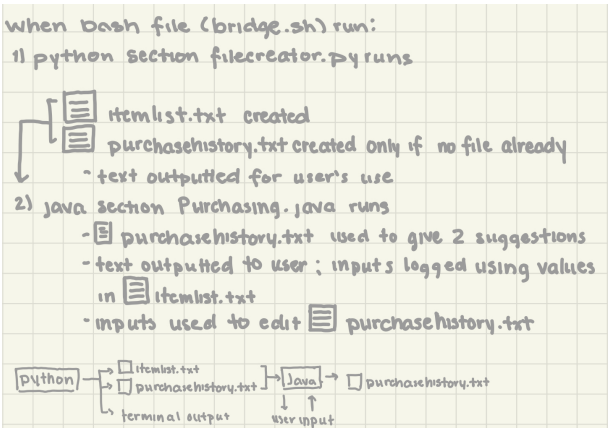
In order to program this project, I wanted to make sure I followed a standardized outline to make sure I was coding in the most efficient way possible. I wanted to try to improve upon my habit of rushing into coding without giving what exactly I was going to do much thought. In order to do this, I came up with this outline I was going to follow. I added sections based off of what I ended up needing:

1. Outline the structure I wanted to have for the overall project
2. Outline code for each needed file in pseudo code
3. Programmed Python section, testing usability and adding changes/corrections
4. Programmed prototype Java section, testing each section before moving on to programming the next
5. Added changes to Java section, added 'append' functionality.
6. Outlined and added commands to bash file, correcting file routing errors and finalizing program through testing

Like previously mentioned, this outline needed changes throughout the process. These changes were as a result of the skills I set out to build upon through this project. I wanted to become a better programmer in Java, as the department now doesn't focus on it as much as it used to. Because of this, I don't know nearly as much about Java as I do Python. I didn't realize just how different Python and Java truly are, leading me to have to learn how to use Array Lists, append to files, and use packages in Java. I also ended up adding classes in both Python and Java to make sure I was writing code that was appropriate for the OOP (object-oriented programming) style of coding. I feel like I learned a lot through programming this project and look forward to possibly using Java in more projects and classes in the future.

2.2 How the languages interact and why

These languages work together well, with Python creating the itemslist.txt and purchasehistory.txt files managed by the program. These files, written as csv files, manage different lists of products. The Java file then inputs these files and uses them as well as user input to append to the purchasehistory.txt file and create a simulation of a grocery store. This structure is written out and illustrated in the diagram below:



This combination is valuable as it gives you the benefits of both languages. Python has a lot of built in functions for developer support, which makes creating files and randomizing values a lot easier than it is in Java or C, for example. Java runs faster than Python, making the more computation and time consuming processes of taking in user input and doing calculations more efficient than it would be otherwise.

This idea was explored in our progress report, where we wrote:

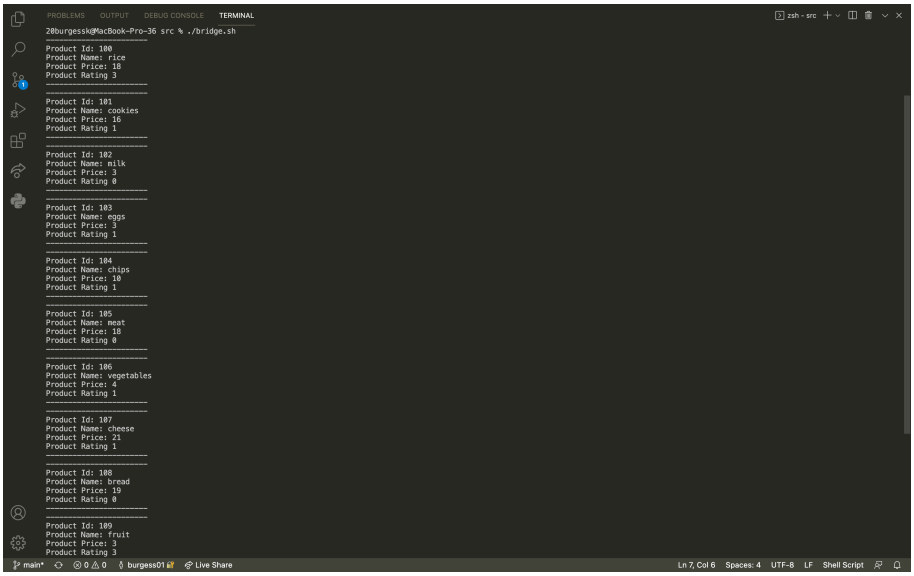
"Because Java is a compiled language, it is faster than Python, which is an interpreted language. This is why the file was created in Python, which is higher-leveled than Java, giving it more ease of use. Java is faster which is why it is utilizing the files Python creates and computing all the data for the project.".

2.3 Python Files

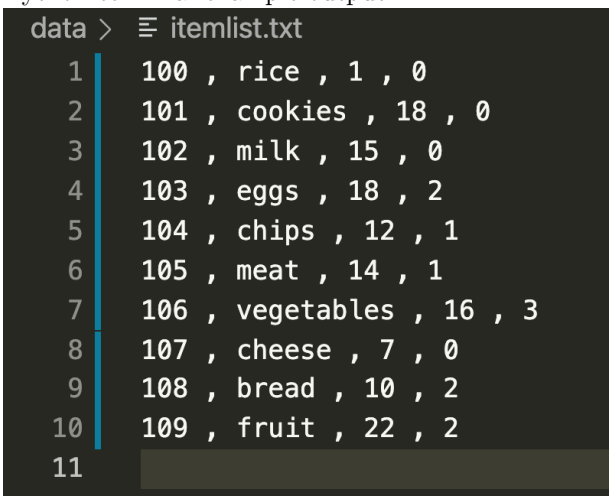
There are two files inside of the Python file folder: filecreator.py and product.py. Product.py is the name of the file for the product class used in filecreator.py. Product is a very basic class, with characteristics for ID, name, price, and rating. This class also has a report() function, which when ran prints out all the information (the ID, name, price, and rating) about that specific class object.

Filecreator.py does exactly what it's name suggests: it creates files. First, this file creates the itemlist.txt file. This file, once created, contains 10 randomized products. These products have standardized names and IDs, but the prices and ratings are randomized, going from 0 to 25 and 0 to 3 respectively. Next, filecreator.py checks to see if a purchasehistory.txt file already exists. If it does, then the file does not edit it. If there isn't one, filecreator creates one. This file is created having three product files, with price and rating being randomized here as well. Lastly, filecreator iterates through all of the products created for itemlist.txt and prints them out using the previously mentioned report() product class method. Examples of the terminal output and text file contents from the Python section

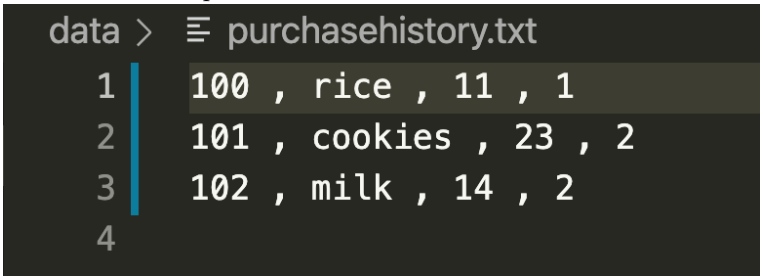
is:



Python terminal example output



itemlist.txt example contents



purchasehistory.txt example contents

2.4 Java Files

There are two files inside of the Java file folder: Product.java and Purchasing.java. Product.java is the file for the Product class used in the Java side of the program. This class is instantiated with four variables, int ID, string name, double price, int rating; the same variables used in the Python

product class. This version of the product class has 4 class functions: getId(), getName(), getPrice(), and getRating(). All of these do what their names imply: getting the object's ID, name, price, or rating, respectively.

Purchasing.java imports the Product class as a package named grocery, as well as many other imports in both java.util and java.io in order to complete the tasks outlined for this file. Purchasing.java includes two functions: suggestions and main. Suggestions is a static funtion that takes in no parameters. Inside of this function is a try/except structure. Inside of the try, suggestions uses a BufferedReader combined with a FileReader to read in purchasehistory.txt from the data folder. Each line is read in separately, with a string ArrayList called items grabbing the item name from every line. After this, two random values are chosen from this using the random library in Java. In order to make this a flexible program, the catch branch is included. Catch is very basic, with it catching any exception that arises in this function and printing it out to the user as a string using the .toString() function.

The main function allows for users to interact with the randomized item list file that Python created. Using the same BufferedReader and FileReader combination that was used in the suggestions function, main reads in all the lines in itemlist.txt, converting each one into a, object of the Product class. After this is finished, the suggestions function is called and displays the two randomized suggestions. Next, an ArrayList named boughtItems of type Product is created. While the user input is larger than 100, the user is repeatedly prompted to enter the ID of the item they want. The price of that item is removed from the amount of money the user said they had.

If the price is higher than the amount of money remaining, the user is told they don't have enough money to buy that item. After the user is done buying items, the bought items appended to the boughtItems ArrayList are appended to the purchasehistory.txt file. In order to append to and not overwrite the file's contents, You have to create an object of FileWriter and then PrintWriter, with the new PrintWriter using the FileWriter object as a parameter. Every item in boughtItems is appended to the purchasehistory file.

Below you can see an example interaction with a user and the corresponding additions to purchasehistory.txt:

```
We recommend that you buy more milk based on your purchase history.
We also recommend that you buy more cookies based on your purchase history.
How much money did you bring to the store?
25
What do you want to buy? Enter the item ID. (If you are done shopping, enter '0'.
109
 fruit was added to your cart. You have10.0left to spend.

What do you want to buy? Enter the item ID. (If you are done shopping, enter '0'.
105
 meat was added to your cart. You have5.0left to spend.

What do you want to buy? Enter the item ID. (If you are done shopping, enter '0'.
107
You don't have enough money to buy this item.
What do you want to buy? Enter the item ID. (If you are done shopping, enter '0'.
0
Thank you for shopping with us!
```

data > purchasehistory.txt

```
1  100 , rice , 11 , 1
2  101 , cookies , 23 , 2
3  102 , milk , 14 , 2
4  109, fruit ,15.0,3
5  105, meat ,5.0,0
6
```

As you can see, the user bought fruit and meat, which was added after the python-generated rice, cookies, and milk entries.

2.5 Bash File

The bash file is where all of the files are called at the same time. The bash file navigates around the different folders of the project to make it easier to call the different programs needed for the combination. The bash file starts in the src file that it is located in. After that, the bash file navigates to the python folder and runs the filecreator.py program, which is explained in depth in the Python section, section 2.3. After those files are created and the terminal output is displayed, the bash file navigates backwards into src again. Here, the product package is compiled, preparing it to be run. After compilation of the Java files, bash navigates to the corresponding folder: java. In the java folder, the newly compiled java program Purchasing is run. The specifics of what occurs when Purchasing is run is discussed in depth in the Java section, section 2.4.

Bash files are a very interesting and useful idea learned in this class. Because of this, below I have included an image of my bash file. Bash files have many useful implications for making programs that are very efficient and implement the advantages of multiple different languages (this concept is explored more in depth for this particular program in section 2.2.

src > $ bridge.sh

```
1  #!/bin/sh
2  cd python
3  python3 filecreator.py
4  cd ..
5  javac java/grocery/*.java
6  cd java
7  java grocery.Purchasing
8  cd ..
```

2.6 In Summary

In summary, our final project for this class, programming languages, was a success. I feel like we both learned a lot about many topics, such as utilizing bash files, coding in Java, using text files for both input and output, as well as appending to a file instead of entirely overriding it.

The results of this project is a combination program that completes everything that we set out for it to do, even if the implementation differed from how we first imagined it would look like. The code seamlessly runs from the python sections to the Java sections, and a user wouldn't even know that it is running using multiple different languages.

There were some challenges to get to this point though. I feel like our communication and teamwork could have been improved on and that was the largest setback for our team. Despite this, our implementation was finished before the progress report was due and nothing was late or delayed before the due dates. Also, Java being a language that neither of us was familiar with held us back as well, as errors that I got stumped both of us. Luckily, Professor Mohan was really helpful and walked me through the errors. I feel like I learned so much more about coding in Java because of that meeting. I would feel comfortable taking on extra projects in Java because of it, and look forward to most likely using the language in my internship this summer.

The largest reward of this project for me was growing as a person. I feel like this project gave me a new and improved outlook on group projects. I feel less focused on the specifics of everything and instead focus more on learning and making sure things get completed. I look forward to taking Software Engineering next semester and using these new skills and my new outlook in this group project focused class.

3 Sources

https://kb.iu.edu/d/agsz

https://docs.python.org/3/library/random.html

https://theconversation.com/the-pandemic-has-made-it-even-harder-for-one-in-three-americans-to-obtain-healthy-affordable-food-169985

https://www.ers.usda.gov/data-products/food-price-outlook/summary-findings/