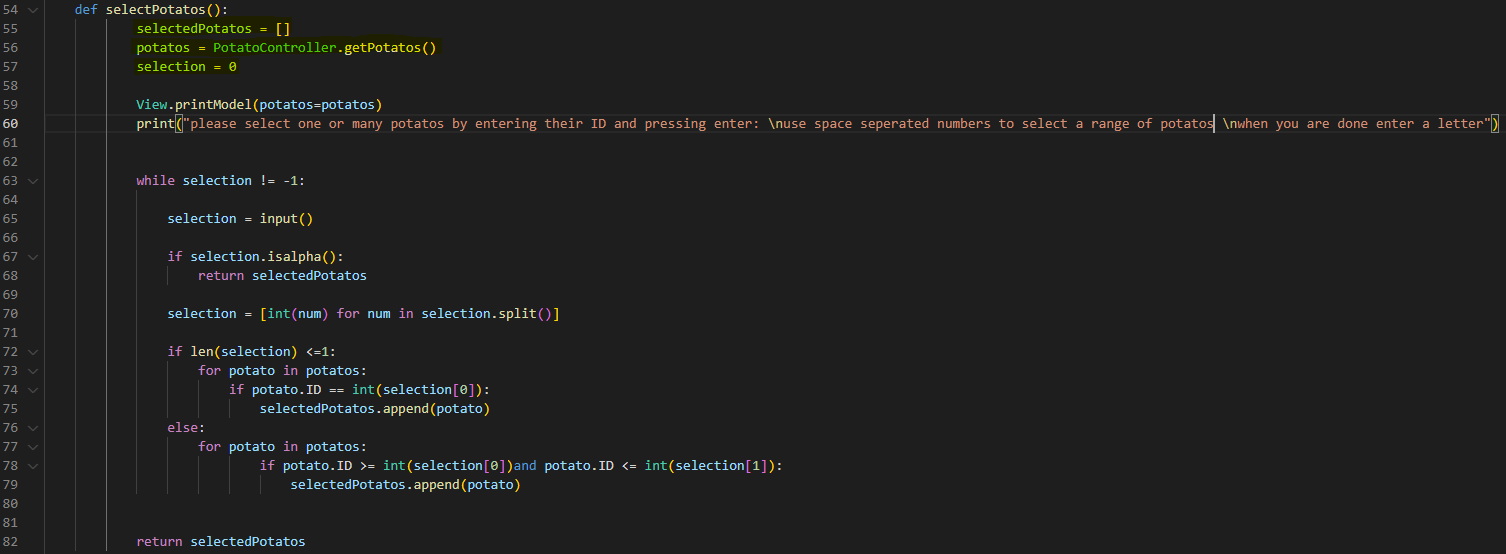
**CST8333 Project By Sebastien Ramsay**

[Rams0130@algonquinlive.com](mailto:Rams0130@algonquinlive.com)

**Evidence of Learning**

**use of variables:**



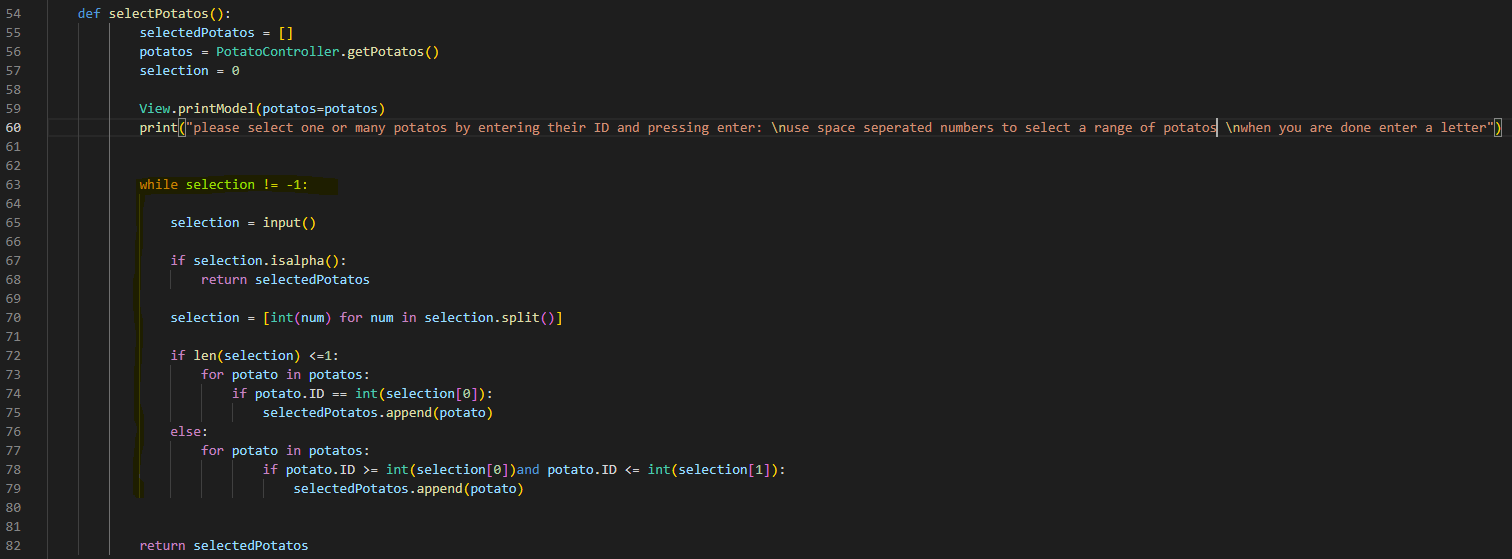
**There are three variables highlighted at the top of this method which is declared in the PotatoView.py file in a View class.**

**The first variable declares a new array to represent the potatos the user has selected.**

**The second variable gets a list of potatos from the controller class.**

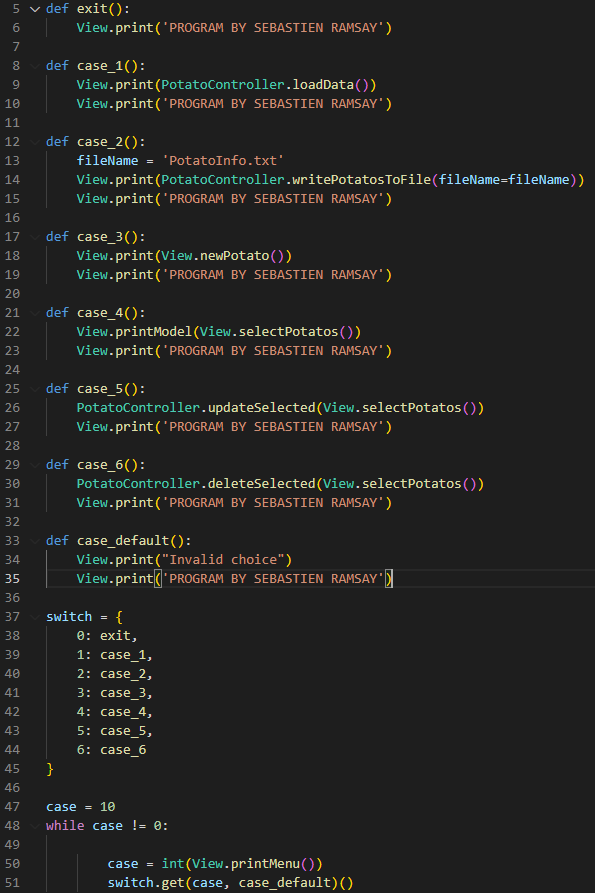
**The third method makes a new integer called selection and sets it to 0 for later use in a while loop.**

**a loop structure:**



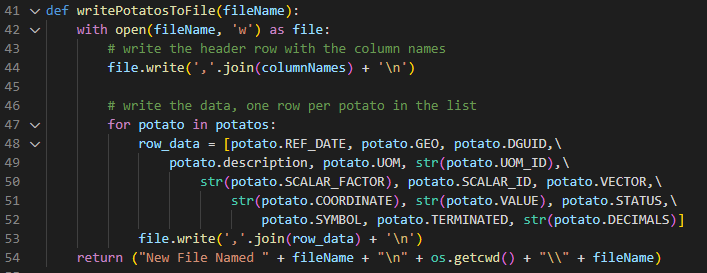
**The while loop structure above is an infinite loop to allow the user to enter as many values as they would like. This while loop is broken by returning the method**

**a decision structure:**



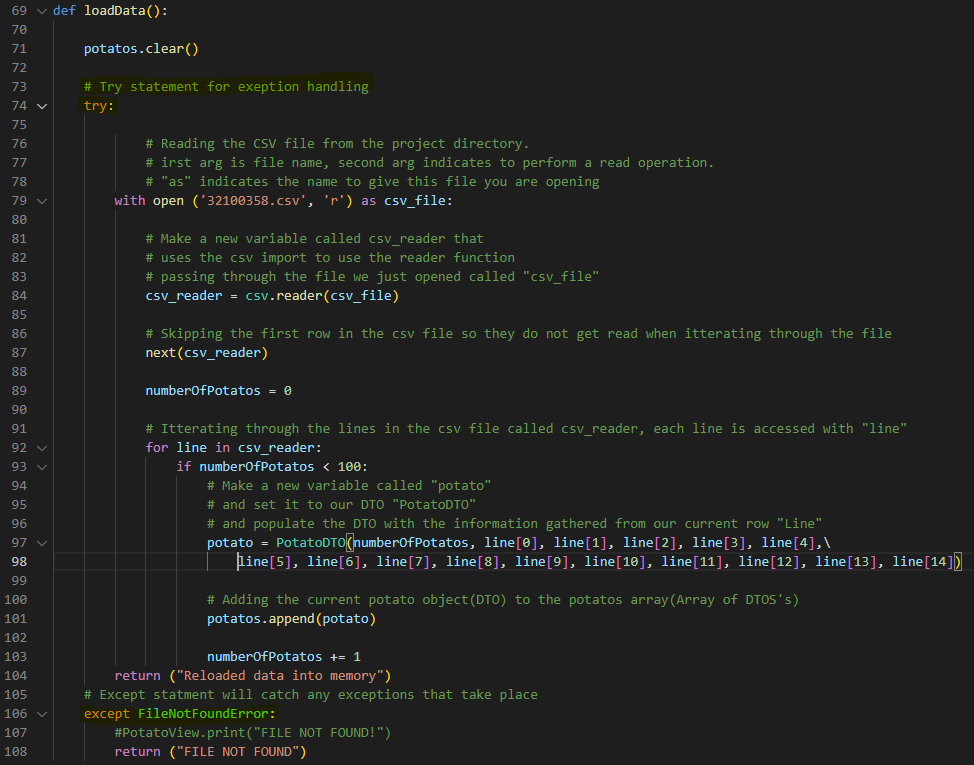
**This decision structure is called a switch statements. This one in particular controls the menu by taking in your selection using View.printMenu() on line 50. The get method is then called on the switch class to make the selection on line 51. If your selection doesn’t exist, case\_default will be called on line 33.**

**File-IO against the dataset:**



**This example of File-IO is writing a file to the current directory. The file is opened as ‘file’ on line 42 and can be written to using file.write.**

**exception handling:**



**This example of exception handling catches FileNotFoundError’s. If the file was not found it will return a string stating so rather than causing an error.**

**use of an API library:**



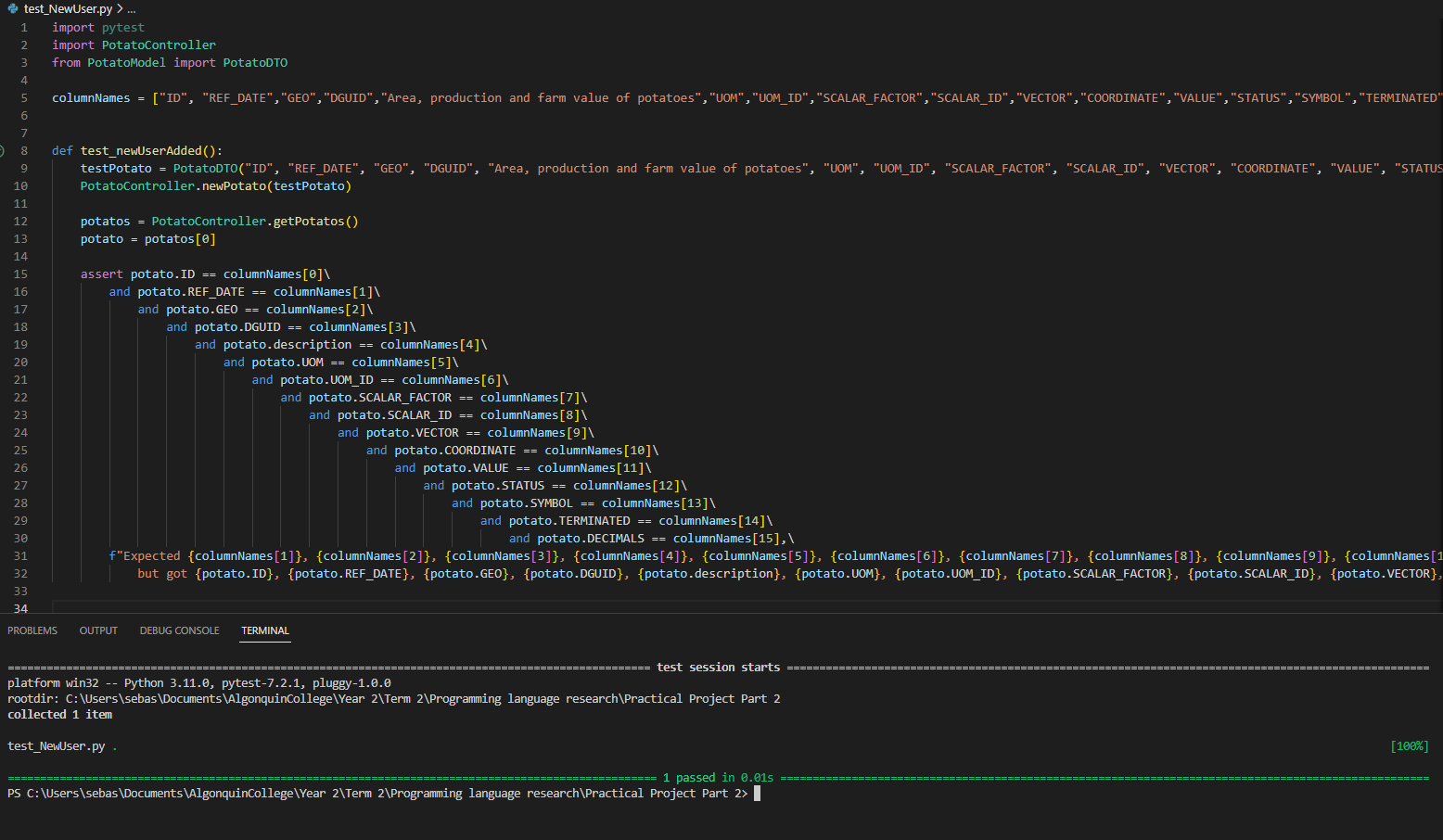
**This is the csv API library I used. It allows me to get information from csv files.**

**an array (or similar data structure):**



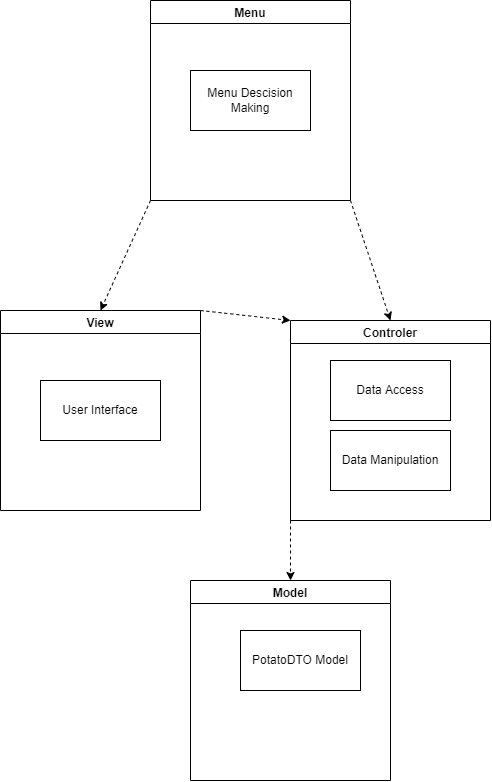
**This is an empty potatos array, it is used to store the list of potatoes loaded into memory.**

**unit testing:**

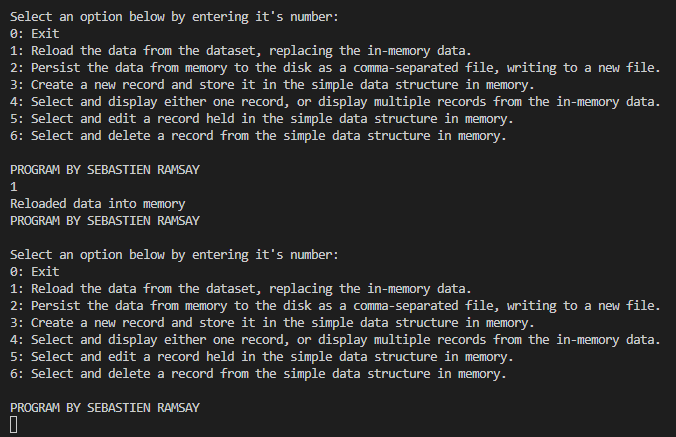


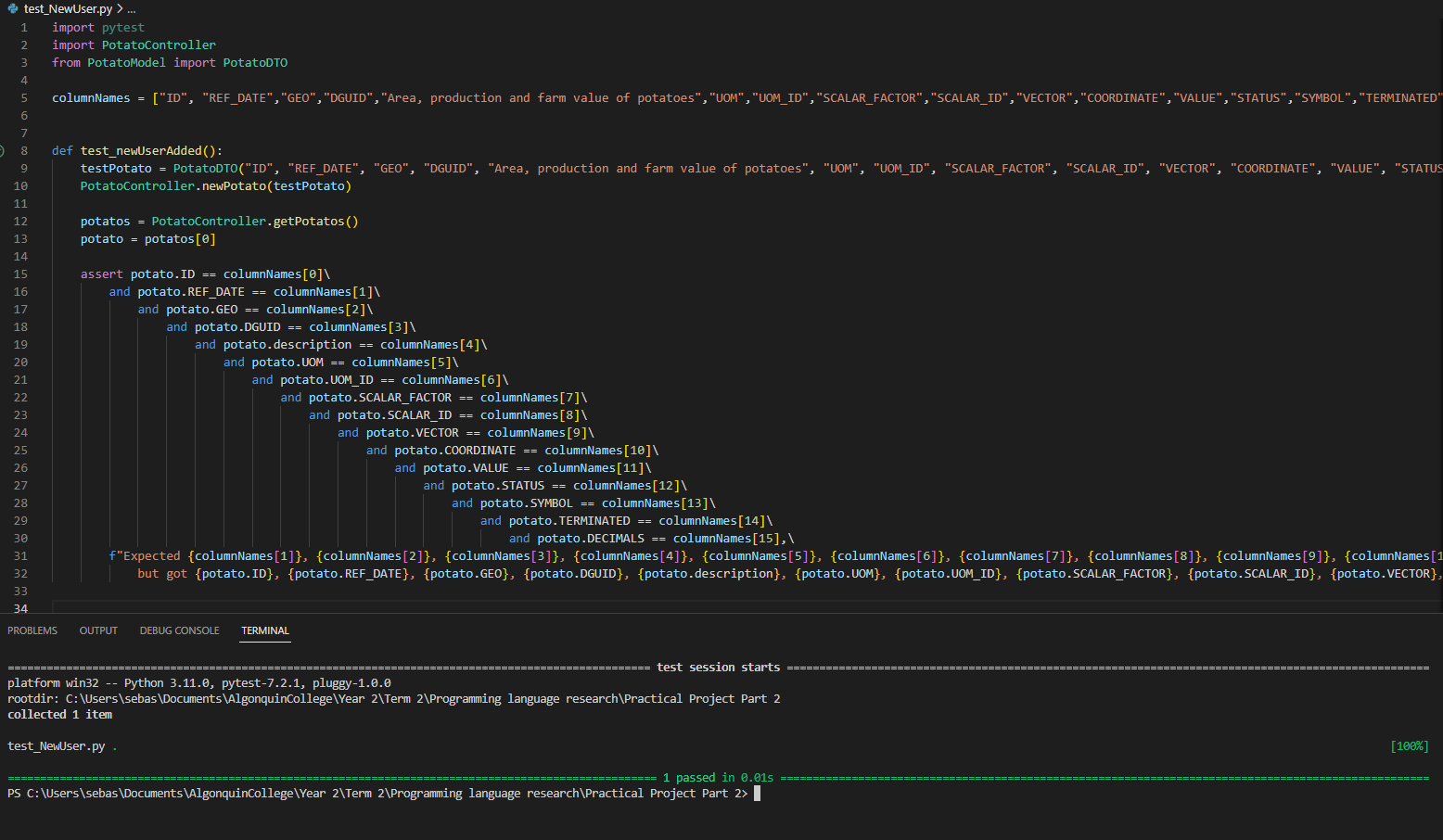
**This is an example of my pytest running and passing. It makes a new PotatoDTO object and checks if the object holds what it is asking it to hold.**

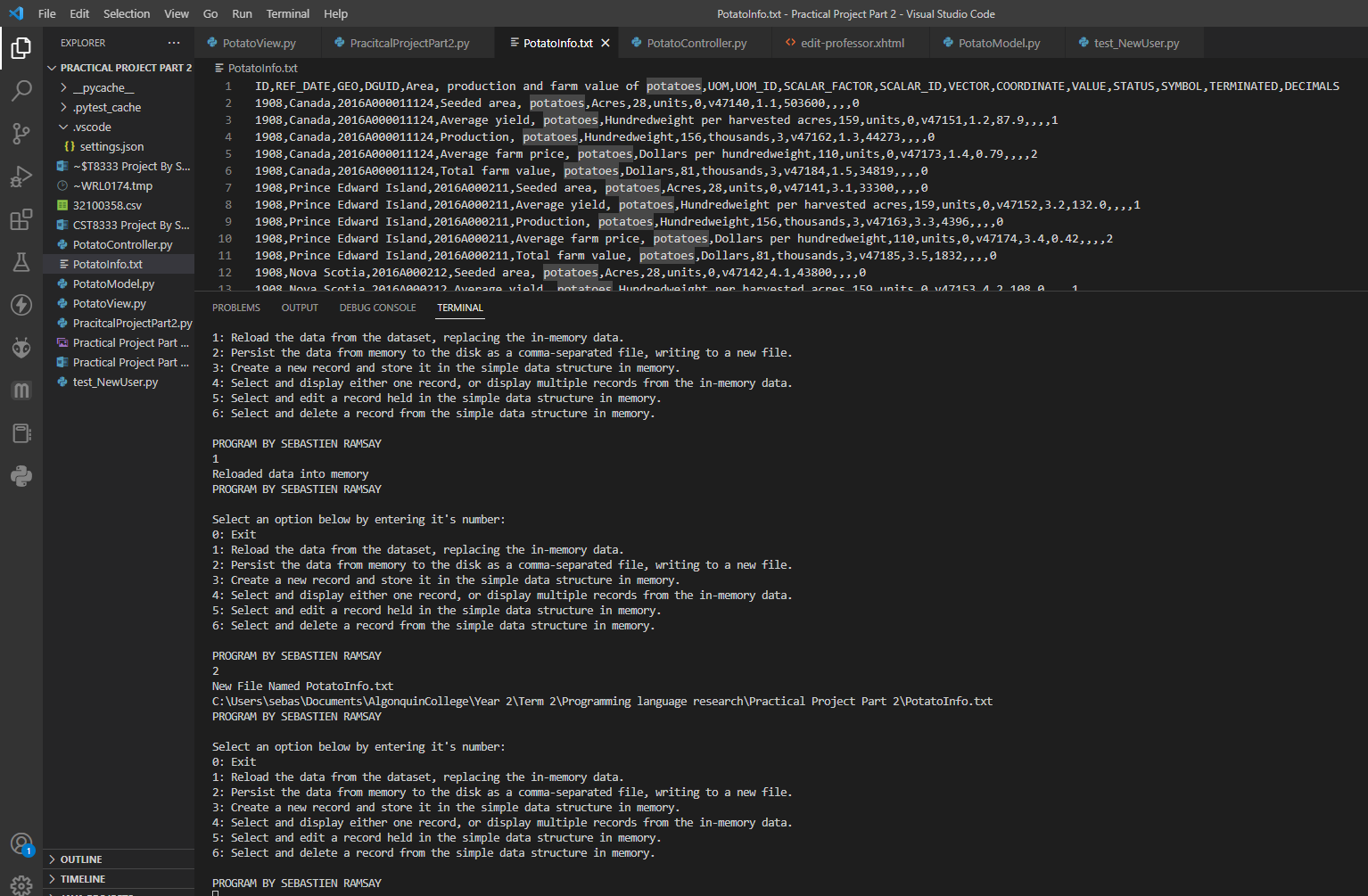
**Program Architecture**

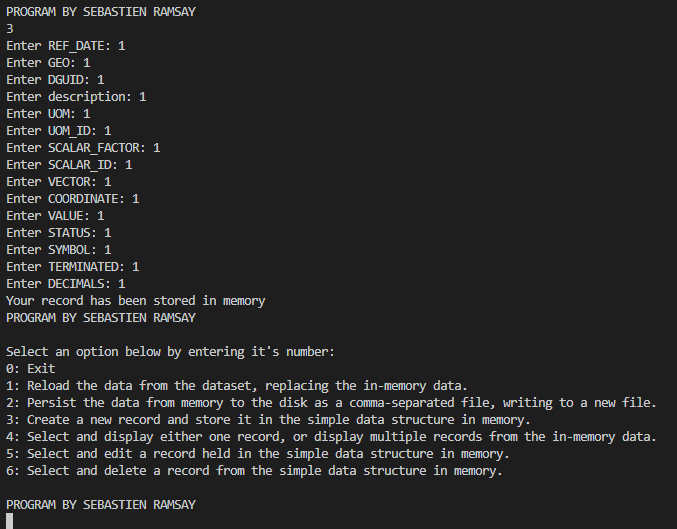


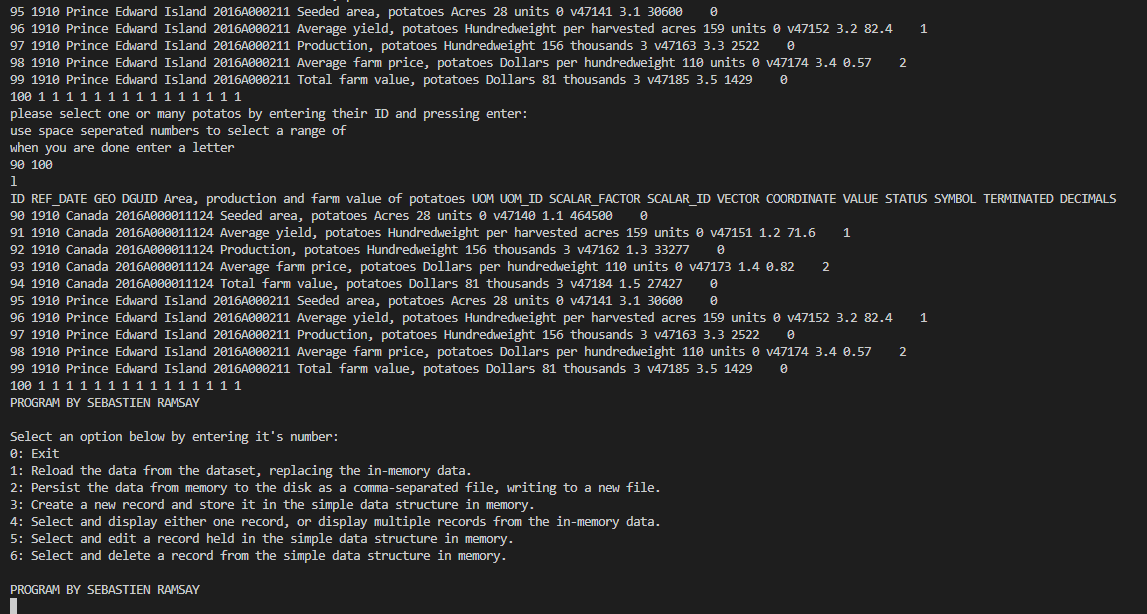
**Program Demonstration via Screen Shots**

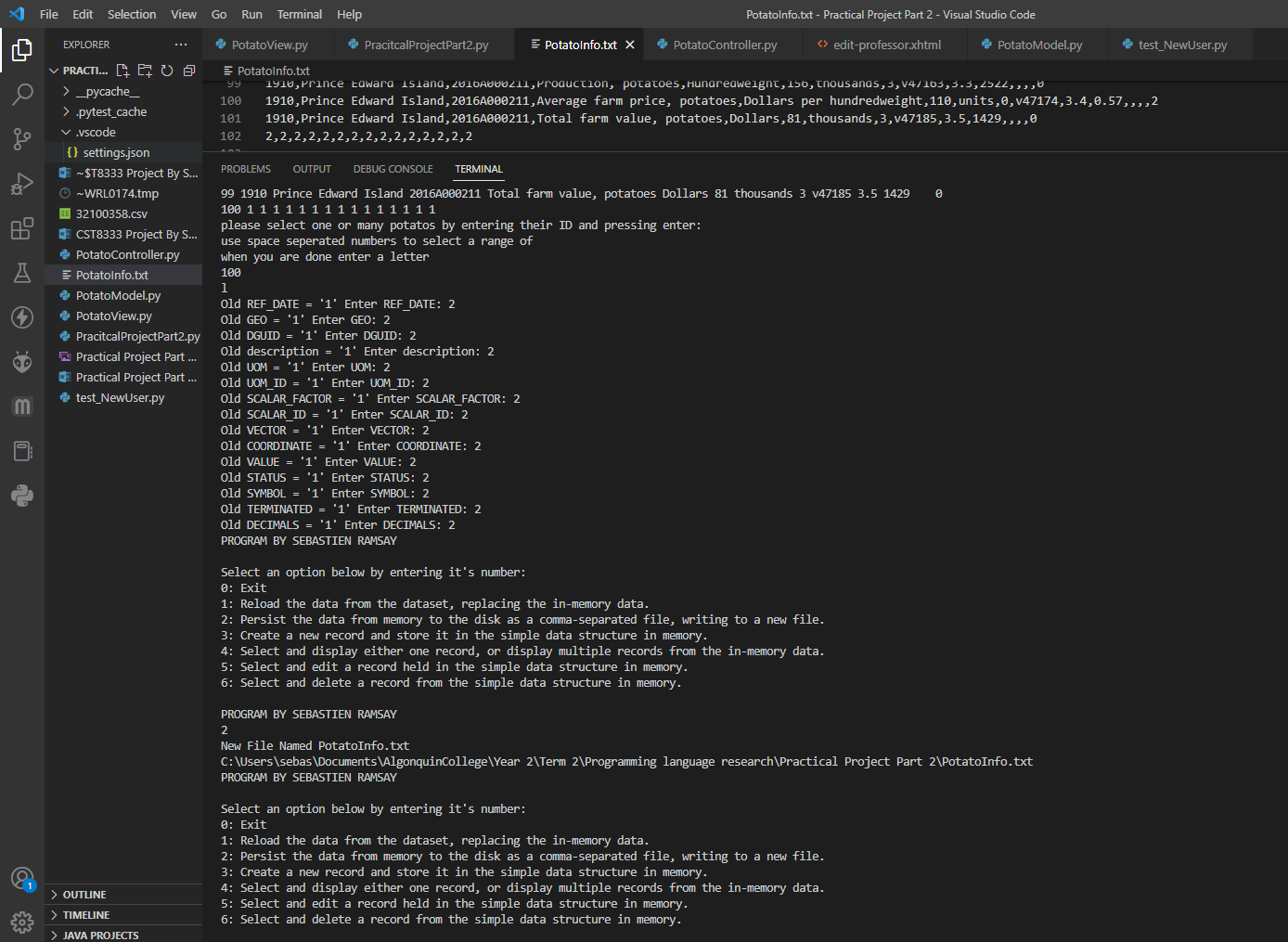


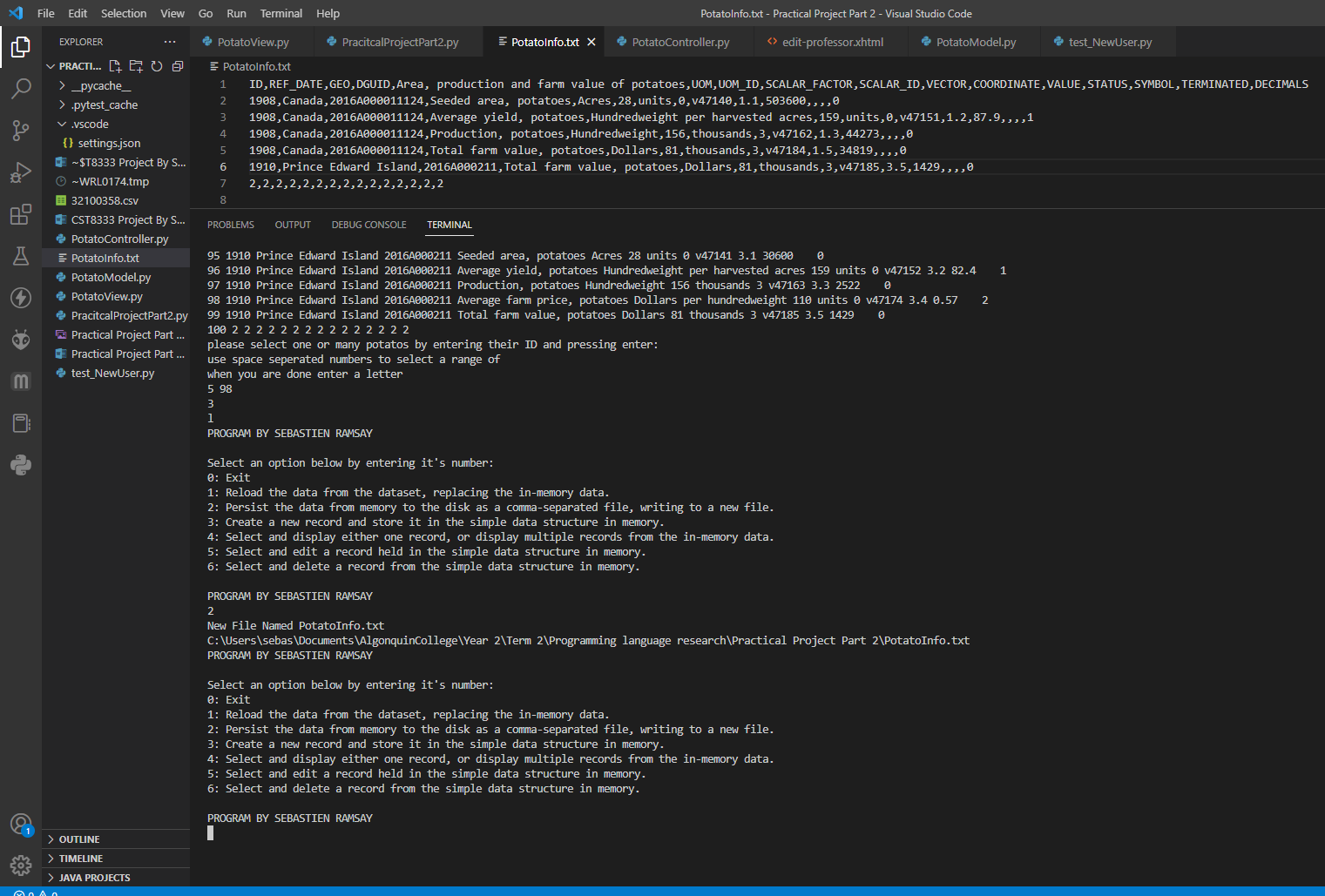




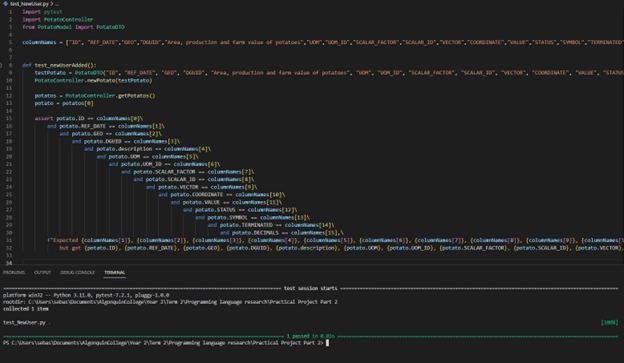








**Unit Testing Demonstration via Screen Shots**



**import csv**

**from PotatoModel import PotatoDTO**

**import os**

**# Making a new array to store DTO's**

**potatos = []**

**# Hardcoding the column names into the program as an array**

**columnNames = ["ID", "REF\_DATE","GEO","DGUID","Area, production and farm value of potatoes","UOM","UOM\_ID","SCALAR\_FACTOR","SCALAR\_ID","VECTOR","COORDINATE","VALUE","STATUS","SYMBOL","TERMINATED","DECIMALS"]**

**# deletes the selectd Potato from the potatos list**

**def deleteSelected(selectedPotatos):**

**"""delete selectedPotatos from potatos array**

**Args:**

**selectedPotatos (Array<potatoDTO>): potatos to be deleted**

**"""**

**for potato in selectedPotatos:**

**potatos.remove(potato)**

**#updates the selected potato by**

**def updatePotatos(updatedPotatos):**

**"""replace in memory potatos with updatedPotatos**

**Args:**

**updatedPotatos (Array<potatoDTO>): potatos that have been updated in the View Class**

**"""**

**for potato in updatedPotatos:**

**#remove the old potato**

**potatos.pop(potato.ID)**

**#add the new one**

**potatos.append(potato)**

**# Write a new file**

**def writePotatosToFile(fileName):**

**"""Write a list of potatoes to a TXT file.**

**Arguments:**

**fileName (str): The name of the file to be created.**

**Returns:**

**str: A message indicating the name of the file and its location.**

**"""**

**# Open a new file called file**

**with open(fileName, 'w') as file:**

**# write the header row with the column names**

**file.write(','.join(columnNames) + '\n')**

**# write the data, one row per potato in the list**

**for potato in potatos:**

**row\_data = [potato.REF\_DATE, potato.GEO, potato.DGUID,\**

**potato.description, potato.UOM, str(potato.UOM\_ID),\**

**str(potato.SCALAR\_FACTOR), potato.SCALAR\_ID, potato.VECTOR,\**

**str(potato.COORDINATE), str(potato.VALUE), potato.STATUS,\**

**potato.SYMBOL, potato.TERMINATED, str(potato.DECIMALS)]**

**#write a row of data comma seperated**

**file.write(','.join(row\_data) + '\n')**

**return ("New File Named " + fileName + "\n" + os.getcwd() + "\\" + fileName)**

**# Add a potato to the list**

**def newPotato(potato):**

**"""Add the potato to the potato array**

**Args:**

**potato (potatoDTO): a single potatoDTO to be added to memory**

**Returns:**

**String: record stored in memory**

**"""**

**potatos.append(potato)**

**return ("Your record has been stored in memory")**

**# Get the list of potatos**

**def getPotatos():**

**"""get the current array of potatos**

**Returns:**

**Array<potatoDTO>: current potatos in memory**

**"""**

**return potatos**

**# replace in memory data with new data read from the csv file**

**def loadData():**

**"""Load data from the csv file using the csv API library**

**Returns:**

**String: data loaded or data failed to load**

**"""**

**#clear in memory data**

**potatos.clear()**

**# Try statement for exeption handling**

**try:**

**# Reading the CSV file from the project directory.**

**# irst arg is file name, second arg indicates to perform a read operation.**

**# "as" indicates the name to give this file you are opening**

**with open ('32100358.csv', 'r') as csv\_file:**

**# Make a new variable called csv\_reader that**

**# uses the csv import to use the reader function**

**# passing through the file we just opened called "csv\_file"**

**csv\_reader = csv.reader(csv\_file)**

**# Skipping the first row in the csv file so they do not get read when itterating through the file**

**next(csv\_reader)**

**numberOfPotatos = 0**

**# Itterating through the lines in the csv file called csv\_reader, each line is accessed with "line"**

**for line in csv\_reader:**

**if numberOfPotatos < 100:**

**# Make a new variable called "potato"**

**# and set it to our DTO "PotatoDTO"**

**# and populate the DTO with the information gathered from our current row "Line"**

**potato = PotatoDTO(numberOfPotatos, line[0], line[1], line[2], line[3], line[4],\**

**line[5], line[6], line[7], line[8], line[9], line[10], line[11], line[12], line[13], line[14])**

**# Adding the current potato object(DTO) to the potatos array(Array of DTOS's)**

**potatos.append(potato)**

**numberOfPotatos += 1**

**return ("Reloaded data into memory")**

**# Except statment will catch any exceptions that take place**

**except FileNotFoundError:**

**#PotatoView.print("FILE NOT FOUND!")**

**return ("FILE NOT FOUND")**

**import pytest**

**import PotatoController**

**from PotatoModel import PotatoDTO**

**columnNames = ["ID", "REF\_DATE","GEO","DGUID","Area, production and farm value of potatoes","UOM","UOM\_ID","SCALAR\_FACTOR","SCALAR\_ID","VECTOR","COORDINATE","VALUE","STATUS","SYMBOL","TERMINATED","DECIMALS"]**

**def test\_newUserAdded():**

**"""Test adding a user to memory. This test makes a new potato and checks if the potato has the requested atributes.**

**"""**

**testPotato = PotatoDTO("ID", "REF\_DATE", "GEO", "DGUID", "Area, production and farm value of potatoes", "UOM", "UOM\_ID", "SCALAR\_FACTOR", "SCALAR\_ID", "VECTOR", "COORDINATE", "VALUE", "STATUS", "SYMBOL", "TERMINATED", "DECIMALS")**

**PotatoController.newPotato(testPotato)**

**potatos = PotatoController.getPotatos()**

**potato = potatos[0]**

**assert potato.ID == columnNames[0]\**

**and potato.REF\_DATE == columnNames[1]\**

**and potato.GEO == columnNames[2]\**

**and potato.DGUID == columnNames[3]\**

**and potato.description == columnNames[4]\**

**and potato.UOM == columnNames[5]\**

**and potato.UOM\_ID == columnNames[6]\**

**and potato.SCALAR\_FACTOR == columnNames[7]\**

**and potato.SCALAR\_ID == columnNames[8]\**

**and potato.VECTOR == columnNames[9]\**

**and potato.COORDINATE == columnNames[10]\**

**and potato.VALUE == columnNames[11]\**

**and potato.STATUS == columnNames[12]\**

**and potato.SYMBOL == columnNames[13]\**

**and potato.TERMINATED == columnNames[14]\**

**and potato.DECIMALS == columnNames[15],\**

**f"Expected {columnNames[1]}, {columnNames[2]}, {columnNames[3]}, {columnNames[4]}, {columnNames[5]}, {columnNames[6]}, {columnNames[7]}, {columnNames[8]}, {columnNames[9]}, {columnNames[10]}, {columnNames[11]}, {columnNames[12]}, {columnNames[13]}, {columnNames[14]}, {columnNames[15]}, {columnNames[16]},\**

**but got {potato.ID}, {potato.REF\_DATE}, {potato.GEO}, {potato.DGUID}, {potato.description}, {potato.UOM}, {potato.UOM\_ID}, {potato.SCALAR\_FACTOR}, {potato.SCALAR\_ID}, {potato.VECTOR}, {potato.COORDINATE}, {potato.VALUE}, {potato.STATUS}, {potato.SYMBOL}, {potato.TERMINATED}, {potato.DECIMALS}."**