The University of Texas at El Paso
Department of Computer Science
CS 3331 – Advanced Object-Oriented Programming, Spring 2022
Instructor: Daniel Mejia
Programming Assignment 4

Academic Integrity Statement:

This work is to be done in a team. It is not permitted to share, reproduce, or alter any part of this assignment for any purpose. Students are not permitted from sharing code, uploading this assignment online in any form, or viewing/receiving/modifying code written from anyone else besides the members of their team. This assignment is part of an academic course at The University of Texas at El Paso and a grade will be assigned for the work produced individually by the student.

Instructions:

This assignment is to be done in teams of 3 people. Your code must be written in Java. You must submit your assignment through Github Classroom. In the comment heading of your source code, you should write your names, date, course, instructor, programming assignment 4, lab description, and honesty statement.

Scenario:

You have set up the foundation of your bank. You now have many customers who are using the bank.

Part A:

Read the requirements described in Part B to complete Part A. Part A should be completed before Part B

1. Write a UML Class Diagram to structure your code using the classes, requirements, and concepts described in Part B
2. Write a level II UML Use Case Diagram based on Part B
    a. 4 Use Cases
    b. 2 Actors
    c. 2 extends
    d. 3 includes
3. Write 2 use case scenarios based on Part B
4. Write a simple UML state diagram
    a. Main Menu to completion of Depositing Money into an account

Part B

1. Conduct a code review on your teammates code
    a. Understand what they did
    b. Understand their approach
    c. Ask questions
    d. Give constructive feedback
    e. Discover and communicate the best way to incorporate their code
2. Combine code from all team members into a single program
    a. Do not simply use one person's entire code
    b. Using Javadoc – write where each piece of code came from (e.g., Taken from Bob, Taken from Alice)
3. Refactor your code to be a singular cohesive program
    a. Remove redundant code
    b. Add code comments
    c. Verify that you are using Object-Oriented Programming
        i. You are properly using abstraction
        ii. You are properly implementing inheritance
        iii. You are properly implementing aggregation and/or composition
        iv. You are properly implementing associations
        v. Objects are created appropriately
        vi. Objects have appropriate interaction with each other
    d. Verify methods are correctly written
        i. Consider method overriding
        ii. Consider method overloading
        iii. Appropriate naming conventions
        iv. Appropriate logic inside of the code
    e. Verify/Modify your code to ensure:
        i. You are using appropriate data structures
        ii. You are using appropriate algorithms
        iii. Your algorithms have reasonable time/space complexity
    f. Readability/Style
        i. Ensure that you are using a standard style for coding throughout
        ii. The following is a standard coding style used by Google: https://google.github.io/styleguide/javaguide.html
            1. Follow these industry standards
4. Use an additional design pattern as part of your refactoring process
    a. Select one of the design patterns discussed in class or use one that you have researched on your own
    b. Discuss its use in the lab report
    c. At least 2 design patterns in total

5. Handle new information related to the customer
    a. From input file
6. Make prompts self-explanatory
    a. Users should not need any background knowledge to use your bank system
    b. Users should intuitively be able to use the system
    c. Think about how you may use a bank system
7. Implement at least 1 (one) interface that is used in at least 2 (two) classes
8. Handle all exceptions appropriately
    a. Create one user defined exception and use it
9. Terminate the program by using writing "EXIT" in the main menu
    a. This should only occur when in the main menu. Do not end the program in the middle of a process (i.e., deposit, withdraw, pay, etc.)
    b. Write a new event list (new csv file – do not overwrite the original input) with the updated values (seat information, money earned, etc.)
    c. Write a new customer list (new csv file – do not overwrite the original) with the updated values (tickets purchased, transaction count, etc.)
10. Bank Statement for five customers
    a. Team Member 1
    b. Team Member 2
    c. Team Member 3
    d. Dr. Mejia
    e. 1 Disney character on the list
11. Create a testing suite
    a. Create Junit tests for at least 5 functions
12. Write the Javadoc for your system
13. Demo with another team, only functionality – DO NOT share code
    a. Make sure that they understand what you're talking about
        i. Teams should use the Javadoc as a starting point
    b. Make sure that they cannot break your code
    c. Teams should focus on ensuring all requirements are met (functionality)
    d. Teams should try to break the system
14. Adjust your functionality based on the demos with your classmates
    a. Fix everything that they were able to break
    b. Handle all exceptions gracefully
15. Write the lab report describing your work (template provided)
    a. Any assumptions made should be precisely commented in the source code and described in the lab report.

b. Write an additional section describing the demo of your classmates
    i. What questions did you have about your classmate's functionality?
    ii. What concerns do you have about your classmate's functionality?
    iii. How did you try to break it?
        1. What test cases did you use?
a. Explain why and how this is a black or white box texting

16. Complete an individual code review as a team code (template provided)
17. Schedule a demo with the TA/IA
18. **If final submission is past the deadline** Your report must have an additional section entitled "Why I submitted late". In that section, explain the reason why your submission was late. (Note: you will still be penalized the typical late penalty)

Deadlines:

March 24, 2022, by 11:59pm – Blackboard:

1. Plan of Action
    a. Must submit by the deadline (-20 pts on PA4 for late submissions)

March 31, 2022, by 11:59pm (Current Progress Commit) – GitHub classroom:

1. UML Class Diagram Progress (.pdf)
2. UML Use Case Diagram Progress (.pdf)
3. UML State Diagram Progress (.pdf)
4. Current Progress Source Code (.java) – Commit current progress up to this point

For each item (1-4)

a. Should be a significant amount of work done (as determined by instructional team)
b. TA/IA will review for progress only

April 10, 2022, by 11:59pm - GitHub Classroom:

1. UML Class Diagram (.pdf)
2. UML Use Case Diagram (.pdf)
3. UML State Diagram (.pdf)
4. Lab report (.pdf file)
5. Source code (.java files)
6. JavaDoc (entire doc folder)
7. Updated Balance Sheet (.csv)
8. Bank Statements for Five Customers
9. Log (.txt)

