**CMSC203 Assignment 2 Implementation**

Class: CMSC203 CRN 23118

 Program: Assignment #2

Instructor: Huseyin Aygun

 Summary of Description: This program models a patient and medical procedures performed on them.

 Due Date: 09/29/2025

 Integrity Pledge: I pledge that I have completed the programming assignment independently.

 I have not copied the code from a student or any source.

**Part1: Pseudo Code:** Here is a pseudo code for Assignment 2 program:

**1. Start program**

**2. Create a Scanner object to get user input**

**3. Prompt user for patient details: first name, middle name, last name, address, phone number, etc.**

**4. Create a Patient object with the inputted data**

**5. Display patient information using Patient's toString() method**

**6. Create three Procedure objects with user input for each procedure**

**a. Prompt user for procedure details: name, date, practitioner, charges**

**b. Set the Procedure attributes using user input**

**7. Display each procedure using Procedure's toString() method**

**8. Calculate total charges for the three procedures**

**9. Display total charges**

**Part2: Comprehensive Test Plan**

A good test plan should be comprehensive. This means you should have a few test cases that test when the input is in and out of range, division by 0, incorrect Data type, etc. (Provide valid and invalid input)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cases | Input | Expected Output | Actual Output | Did Test Pass? |
| Case 1 | Valid input | All details correct | All details correct | yes |
| Case 2 | Incorrect data type | Request re-entry of data | Requested re-entry of data | No |
| Case 3 | All blank | All blank output, numbers default to 0 | Error | No |
| Case 4 | Negative ints | Negative cost | Negative cost | Yes |

**Part3: Screenshots related to the Test Plan:**

**Case 1**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Case 2 – (Project outline does not require input validation)**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Case 3**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Case 4**

A screen shot of a computer

AI-generated content may be incorrect.

**Lessons Learned** <Provide answers to the questions listed above>**:**

Write about your Learning Experience, highlighting your lessons learned and learning experience from working on this project.

What have you learned?

I learned how to implement a simple object-oriented program in Java, specifically focusing on creating classes and interacting with user input. The project helped me understand the importance of constructors, accessor/mutator methods, and using methods to manipulate object data.

I also learned how to handle different types of user inputs, how to format output effectively, and how to calculate and display data (like total charges for procedures).

What did you struggle with?

Initially, I struggled with validating user input effectively, especially handling invalid or out-of-range values like negative charges or improperly formatted dates. I also found managing edge cases (e.g., empty fields or incorrect formats) to be a bit tricky. Ensuring that the program would not crash due to invalid input was a key challenge.

Another challenge was creating a clean and effective way of displaying error messages for invalid inputs without overwhelming the user. I had to balance between providing enough information to help the user correct their mistakes and not cluttering the interface with too many warnings.

What would you do differently on your next project?

On my next project, I would spend more time focusing on input validation techniques, including using regular expressions or built-in Java methods to ensure data integrity. I would also use exception handling more extensively to deal with unexpected errors in input or calculation more gracefully.

I would consider using Java’s try-catch blocks more to handle invalid user input more effectively, rather than just displaying error messages. This would improve the robustness of the application.

What parts of this assignment were you successful with, and what parts (if any) were you not successful with?

I was successful in structuring the project and implementing the required functionality (i.e., creating and displaying patient and procedure information, calculating total charges). The program is able to take input, process it, and output the expected results in a straightforward way.

The main area I struggled with was managing edge cases effectively. Some input validation was not as robust as I would have liked, and the program could benefit from more sophisticated error handling and validation techniques.

Additional resources:

N/A

**Check List:** <Provide answers to the column Y/N or N/A >**:**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** |  | **Y/N** | **Comments** |
|  | **Assignment files:** |  |  |
|  | * FirstInitialLastName\_ Assignment#\_Moss.zip |  |  |
|  | * FirstInitialLastName\_Assignment#.docx/.pdf |  |  |
|  | * Source java files |  |  |
|  | **Program compiles** |  |  |
|  | **Program runs with desired outputs related to a Test Plan** |  |  |
|  | **Documentation file:** |  |  |
|  | * Comprehensive Test Plan |  |  |
|  | * Screenshots related to the Test Plan |  |  |
|  | * Screenshots of your GitHub account with submitted Assignment# (if required) |  |  |
|  | * UML Diagram (if required) |  |  |
|  | * Algorithms/Pseudocode (if required) |  |  |
|  | * Flowchart (if required) |  |  |
|  | * Lessons Learned |  |  |
|  | * Checklist is completed and included in the Documentation |  |  |