**CMSC203 Assignment 2 Implementation (Documentation)**

Class: CMSC203 CRN 40438

 Program: Assignment #2

Instructor: Grigoriy Grinberg

 Summary of Description: Three classes -- A class that defines a Patient with all identifying and contact information, a class that defines a medical procedure performed on a patient, and a driver class to test Patient and Procedure classes.

 Due Date: 07/29/2025

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

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

**Part 1: Pseudo Code:** Here is a pseudo code for Assignment 2 program:

*1. Start the program.*

*2. Prompt the user to enter the patient's:*

*a. First name*

*b. Middle name*

*c. Last name*

*d. Street address*

*e. City*

*f. State*

*g. ZIP code*

*h. Phone number*

*i. Emergency contact name*

*j. Emergency contact phone*

*3. Create a Patient object using the collected information.*

*4. Prompt the user to enter details for Procedure 1:*

*a. Name*

*b. Date*

*c. Practitioner*

*d. Charge*

*5. Create a Procedure object using the Procedure 1 details.*

*6. Prompt the user to enter details for Procedure 2:*

*a. Name*

*b. Date*

*c. Practitioner*

*d. Charge*

*7. Create an empty Procedure object and set its values using setters.*

*8. Prompt the user to enter details for Procedure 3:*

*a. Name*

*b. Date*

*c. Practitioner*

*d. Charge*

*9. Create a Procedure object using a constructor with name and date, then set practitioner and charge using setters.*

*10. Display the patient's information using the ‘toString()’ method of the Patient class.*

*11. Display the details of all three procedures using their ‘toString()’ methods.*

*12. Calculate the total charge for the three procedures.*

*13. Print the total charges.*

*14. Print a statement with the developer’s name and date.*

*15. End the program.*

**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 patient info and 3 valid procedures with realistic names and charges.  Names: John Q Pork  Address: 1000 Bacon St  Procedure 1: Appendectomy,  $6700  Procedure 2: Tonsillectomy, $9999.99  Procedure 3: Cholecystectomy, $11123.44 | Displays all patient and procedure info, total charges = $27,823.43 | As expected | Yes |
| Case 2 | Valid patient info with some **empty strings** (e.g., missing middle name, missing city) and zeroes for charge values    Procedure 1: Name missing  Charges still valid | Should still compile and run. Displays empty fields as blank strings. Charges add correctly. | As expected | Yes |
| Case 3 | Entered **non-numeric charge** (“twenty” for Procedure 1’s charge) | Program throws NumberFormatException at Double.parseDouble(...) and stops execution | As expected | Yes |
| Case 4 | |  | | --- | |  | |  |  |  | | --- | | Entered large test values for charge values to test handling of large numbers. | | Correctly added charges and displayed total charges at runtime. | As expected | Yes |

**Part 3: Screenshots related to the Test Plan:**

**Case 1:**

**A screenshot of a computer

Description automatically generated**

**Case 2:**

**A screenshot of a computer

Description automatically generated**

**Case 3:**

**A screenshot of a computer

Description automatically generated**

**Case 4:**

**A screenshot of a computer

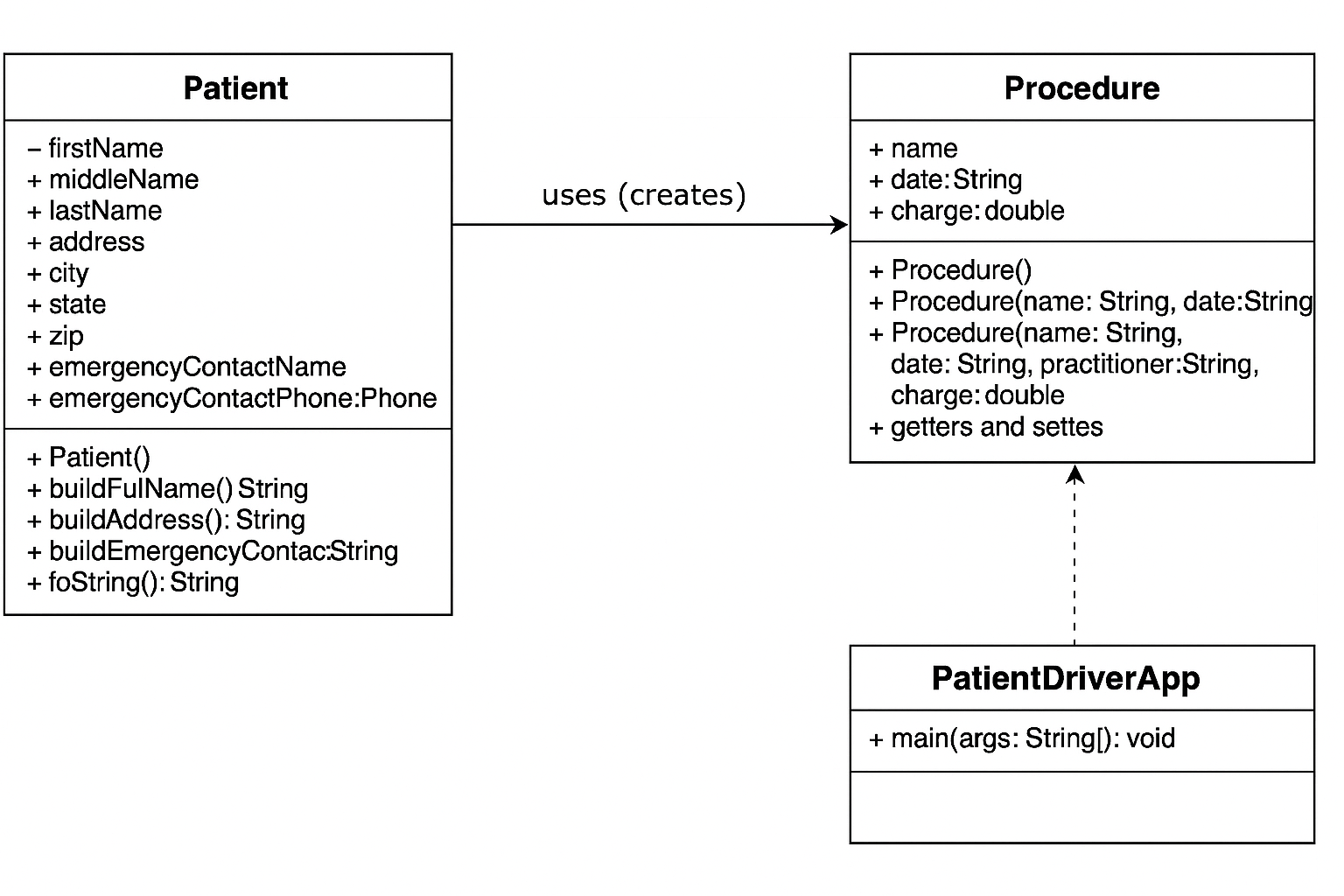
Description automatically generated**

**Terminal/Command Line Run:**  (highlighted line is where output begins)

**A screenshot of a computer

Description automatically generated**

**UML Diagram:**



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

**What have you learned?**From working on this assignment, I learned how to create and use multiple classes in a Java program. I became more comfortable with constructors (including overloaded ones), accessor and mutator methods (getters/setters), and using objects to model real-world ideas like patients and medical procedures. I also practiced formatting output, and learned how to work with user input using the Scanner class.

**What did you struggle with?**  
I initially struggled with managing files across Eclipse and the Terminal, especially when dealing with default packages and understanding how Java handles class locations. There was also some confusion when attempting to run the program from the command line, particularly due to the lack of package declarations and making sure all .java files were in the correct place. Accidentally deleting files and resolving Eclipse’s behavior with default packages was also challenging.

**What would you do differently on your next project?**  
Next time, I’ll be more organized with my file management from the beginning — including consistently backing up important files and confirming that my directory structure aligns with Java’s expectations. I’d also document my steps earlier in the process to avoid forgetting anything later when assembling deliverables like screenshots and test cases.

**What parts of this assignment were you successful with, and what parts (if any) were you not successful with?**  
I was successful in writing clean, functioning classes (Patient, Procedure, and the driver class). I also completed thorough test cases and matched the rubric requirements. I was not initially successful at running the program from the terminal due to Java version conflicts and misconfigured file locations, but I resolved those by reinstalling Java 17 and properly compiling the files in the right directory.

**Provide any additional resources/links/videos you used to while working on this assignment/project.**

**Oracle Java 17 Documentation**[**https://docs.oracle.com/en/java/javase/17/docs/api/**](https://docs.oracle.com/en/java/javase/17/docs/api/)

**StackOverflow - General Java Questions**[**https://stackoverflow.com/questions/tagged/java**](https://stackoverflow.com/questions/tagged/java)

**macOS: Setting the Default Java Version (Terminal)**[**https://developer.apple.com/library/archive/qa/qa1170/\_index.html**](https://developer.apple.com/library/archive/qa/qa1170/_index.html)

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

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