**CMSC203 Assignment 2 Implementation (Documentation)**

Class: CMSC203 CRN 21955

 Program: Assignment #Project 2

Instructor: Gary Thai

 Summary of Description: This project aims to build a class named Patient that has fields for different kinds of data.

 Due Date: 10/04/2024

 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 - Create a class named Patient to hold information about them.

2 - Fields are First name, middle name, last name, street address, city, state, ZIP code, Phone number, and emergency contact’s name and phone number.

3 - Create no-arg constructor.

4 - Create a parametrized constructor that initializes patient’s first, middle and last name.

5 - Create a parametrized constructor that initializes all attributes of the patient to the given values.

6 - Create getters and setters for each attribute.

7 - Create a method named buildFullName that returns a String with the values of first, middle, and last name concatenated.

8 - Create a method named buildEmergencyContact that returns a String which is the concatenation of emergency name and emergency phone separated by space.

9 - Create a toString method that displays all information of the patient using the custom methods to display the information.

10 - Create a class named Procedure to hold information about the medical procedures that have been performed on a patient.

11 - Fields are name of the procedure, date, practitioner, charges.

12 - Create 3 constructors: a no arg, a parametrized that initializes procedure’s name and date, and a third one that initializes all attributes of the procedure to the given values.

13 - Create getters and setters for each attribute.

14 - Create a toString method that displays all information of the procedure.

15 - Create a PatientDriverApp Class that creates an instance of the Patient class, initialized with sample data from user input.

16 - Create three instances of the Procedure class, each instance should use one of each of the three constructors. (IF YOU USE A DIFFERENT CONSTRUCTOR THAN THE ONE THAT INITIALIZES ALL ATTRIBUTES, YOU MUST SET THE VALUES FOR ALL ATTRIBUTES)

17 - Custom methods will be displayPatient, which displays the patient’s information based on a patient object. displayProcedure will display procedure information, and calculateTotalCharges takes three procedures as parameters and returns the total charges of these procedures.

18 - Program displays: patient’s information, information about all three procedures separately, and total charges of the procedures separated by comma with two decimal points.

**UML DIAGRAM**

|  |  |  |
| --- | --- | --- |
| **Patient** | **Procedure** | **PatientDriverApp** |
| -firstName : String | -procedureName: String | -patient: String |
| -middleName: String | -date: String | -procedure1: String |
| -lastName : String | -practitioner: String | -procedure2: String |
| -streetAddress: String | -charges: double | -procedure3: String |
| -city: String | +Procedure() | +displayPatient(patient: String):void |
| -state: String | +Procedure(procedureName:  String, date : String); | +displayProcedure(procedure: String):void |
| -zip: String | +Procedure(procedureName:  String, date: String,  practitioner: String, charges: double); | +calculateTotalCharges(procedure1: |
| -phoneN: String | +setProcedureName(  procedureName:  String): void |  |
| -emergencyC: String | +setDate(date:String):void |  |
| -emergencyPhone: String | +setPractitioner(  practitioner:String):void |  |
| + Patient(); | +setCharges(charges:double):  Void |  |
| + Patient(firstName: String, middleName: String, lastName : String); | +getProcedureName():String |  |
| + Patient(firstName: String, middleName: String, lastName : String, streetAddress: String, city:String, state: String, zip: String, phoneN: String, emergencyC: String, emergencyPhone: String); | +getDate():String |  |
| +setFirstName(firstName: String): void | +getPractitioner():String |  |
| +setMiddleName(middleName: String): void | +getCharges():double |  |
| +setLastName(lastName: String) : void | +toString():String |  |
| +setStreetAddress(streetAddress: String):void |  |  |
| +setCity(city: String): void |  |  |
| +setState(state: String): void |  |  |
| +setZip(zip: String):void |  |  |
| setPhoneN(phoneN: String): void |  |  |
| +setEmergencyC(emergency: String):void |  |  |
| +setEmergencyPhone(emergencyPhone:  String): void |  |  |
| +getFirstName():String |  |  |
| +getMiddleName():String |  |  |
| +getLastName():String |  |  |
| +getStreetAddress(): String |  |  |
| +getCity(): String |  |  |
| +getState(): String |  |  |
| +getZip(): String |  |  |
| +getPhoneNumber(): String |  |  |
| +getEmergencyC(): String |  |  |
| +getEmergencyPhone():String |  |  |
| +buildFullName():String |  |  |
| +buildAddress(): String |  |  |
| +buildEmergencyContact():String |  |  |
| +toString():String |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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 - Test Patient full name with valid data | First Name: "John", Middle Name: "A.", Last Name: "Doe"  Address: 123 Red Lane Rockville MD 20906  Phone Number: 123456789  EmergencyContact: Marie 987654321 | Name: "John A. Doe"  Address: 123 Red Lane Rockville MD 20906  Phone Number: 123456789  EmergencyContact: Marie 987654321 | Patient Details:  Name: John A. Doe.  Address: 123 Red Lane Rockville MD 20906  Phone Number: 123456789  EmergencyContact: Marie 987654321 | YES |
| Case 2- Test Patient full name with empty fields | First Name: "", Middle Name: "", Last Name: "Doe" | Doe | Doe | YES |
| Case 3- Test Procedure creation with incorrect data type | Name: "Endoscopy", Date: "08/01/2024",  Practitioner: “Dr.Moody”  Charges: "One Hundred" | Exception | Exception | YES |
| Case 4- Test Procedure with out-of-range charges | Name: "Endoscopy", Date: "08/01/2024",  Practitioner: “Dr.Moody”  Charges: -500.00 | Exception | Exception | YES |

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

**Case 1**

**A close up of a number

Description automatically generated**

**Case 2**

**A white screen with black text

Description automatically generated**

**Case 3**

**A screenshot of a computer error

Description automatically generated**

**Case 4**

****

**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 that the project can be more enjoyable if the tasks are split and done separately.

What did you struggle with? Generating the Javadoc documentation and making sure the constructors were correct. I also had to meet with the class coach to understand how the relationship between classes were demonstraded in the UML diagram.

What would you do differently on your next project? Start on day 1

I would one hundred per content allocate even more time, for the project and try to meet with Prof. Thai for questions.

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

I was successful separating all of the different deliverables and getting them done a bit by bit. This allowed me to have better focus and attention to detail to the whole project.

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

The text book: Revel for Starting Out with Java from Control Structures through Objects, 8e.

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

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