// Student Name: Reziyemu Sulaiman

// Seneca Student ID: 106-153-208

// Seneca email: rsulaiman2@myseneca.ca

// Date of completion: 2021-05-15

In this project, I was able to fully learn the characteristics of object-oriented languages such as: class, object, encapsulation, inheritance, polymorphism, and abstraction. I learned how these building blocks are related to each other by implementing General Hospital Pre-Triage Application.

Below is my overview about all these concepts, later I will talk about the issues that was challenging while I was doing this project.

To begin with, I would like to start with the concept of class. Take an example, the Patient class: A patient has created by the patient’s name, OHIP insurance number, and ticket. This class performs actions such as, categorize patients as covid or triage patient, set the time that the ticket was issued to patients, and get the ticket number of the patient. In simple words, the Patient class is just a design where everything is defined.

Next, is about inheritance. Considering Patient as a class (which it has designed above), Covid patient and Triage patient are also classes, but most of the properties and functions are already included in the Patient class, hence, they can inherit everything from class Patient using the concept of inheritance. What I mean is, Covid patient and Triage patient both inherit the class Patient mean that as they have some common features which I have already implemented in the Patient class, hence, I do not have to implement them again in the Covid patient and Triage patient classes. They can simply inherit them from the class Patient. This is what inheritance means…

Next, is about object. Object is an instance of a class. For example, my name is Rezi, and I am an instance of patient class. Me, or my friends are the forms of this class while a Patient class is just a logical definition. It means, Patient is a class of which I am an object with the name Rezi.

Next, is about abstraction. Abstraction means providing only required details to the user while hiding the details. Take an example, the Patient class has private members, such as: patient name, OHIP insurance number, and ticket of patient. These private members can be accessed only from within the class. They are not allowed to be accessed by the user. The patient class has also some methods that are declared as public, such as: setting the patient arrival time and patient ticket number. These public members can be accessed from anyone whether from the derived classes or the user. The Patient class can decide which data members will be visible to the outside world and which is not. This is what abstraction means by displaying only required information and hiding the details.

Next, is about encapsulation. Encapsulation means binding together the data and the functions that manipulate them. As I stated before, since user can not directly access the private members of the class, the user can request to the class to give that particular data. Take the Ticket class as an example, this class has private member called m\_number (which is ticket number), this class has a public method, such as:

int number()const; // return ticket number

Since m\_number (ticket number) is made private, so that this variable can be accessed only by using method number() that are present within the class. Therefore, the variable m\_number and the method number() have bound together that is encapsulation.

Last but not least, is about polymorphism. It means having many forms. Here I want to talk about two types of polymorphism: one is overriding, another one is overloading.

Function overriding means redefinition of base class function in its derived class with same signature. This can only be done in the derived class, and it can be achieved at run time. In other words, function overriding means when derived class function has something to add or to do a different job than the base class function.

Take an example of Patient class and Triage patient class. Both classes have the same two functions below:

std::ostream& write(std::ostream& ostr)const;

std::istream& read(std::istream& istr);

When I implement these functions in the derived Triage patient class, in addition to the implementation of base class Patient, I added some logic to read and write the symptoms of the patient in the derived Triage patient class which means I override the functions from base class Patient.

However, Function overloading means change the part used to perform something. It means provides multiple definitions to the functions by changing signature. In other words, for the same function, take different arguments where to change the number of argument or the type of argument. For example, in the utils module, I implemented getInt() function where the first getInt() function takes only one argument (which is prompt), and the second getInt() function takes five arguments (min, max, prompt, error message, and showRangeAtError). This is what overloading is bu changing number of arguments for the same function, and this can be done in the base class as well as derived class. This can be achieved at compile time.

Lastly, I would like to talk about the parts that was challenging for me while doing this project.

While doing this project, I encountered to get unknown size string from the user. In utils.cpp, I implemented function getcstr(). This function receives an unknown size of string, allocates memory for the string, and return the dynamically allocated memory. The way I implement this function is: at first, I defined string object and used getline() function to get unknown size of string from user up to the delimiter that the user has entered. Then, I checked if the user input is in a good state or not. If it is, I allocated the memory for the size of string plus null termination and copied the string into it. If user input is not in a good state, I cleared the error flags and ignored the rest of the characters. This is how I implemented getcsr() function.

Next, is about how I used this getcstr() function in the further classes.

When I implemented the Patient class and Triage patient class, I had to extract unknown size of string to store patient’s name and the patient’s symptoms, respectively. In order to avoid the duplication of code that I’ve had implemented in the getcstr() function in the utils module, I created a local dynamically allocated cString in read() function (in both classes), then I called to getcstr() function to store the unknown size of string into local dynamically allocated cString. Then, to make a copy of this string to patient’s name or symptoms, I deleted the previously allocated memory for the name and symptoms, then allocated a new memory for them, and then copied the unknown size string to the patient’s name and symptoms. In final step, I deleted the local dynamically allocated cString. This way, I was able to achieve to make a deep copy from the user input (unknown size string) to the patient’s name and patient’s symptoms step by step.

All in all, by doing this project, I was able to improve my weakness about some concepts that I have learned in the class lecture. I reviewed many concepts while doing this project. Milestone 1 and 5 are the most toughest for me which it needs a lot of logic in some of the functions, but it has helped me to strengthen my knowledge about all the things that I’ve learned through out this semester.