Hacettepe University

Department Of Computer Engineering

BBM104 Introduction to Programming II Laboratory

Programming Assignment III

Name and Surname: Abdullah Palaz

Student Number: 21993016

Subject: Inheritance and Polymorphism

Submission Date: 01.05.2020

Due Date: 22.05.2020 (23:59)

Advisors: R.A. Bahar GEZICI, Nebi YILMAZ

Problem Definition:

It is expected to develop a Hospital Management System with given rules using inheritance and polymorphism concepts of object oriented programming. There is one input file named “input.txt” which contains instructions and there is two other data files named “admission.txt” and “patient.txt” which contains admission data and patient data respectively.

Solution:

To store the data in the data files, an ArrayList is created. After reading all the data, the program reads instructions and manipulates data through them. Then writes new data to data files. As a basic structure of hospital: patients has admissions, admissions has examinations and examinations are decorated by operations.

Algorithm:

1- Creating an ArrayList named “patients” to store all the data in the data files(.txt) and manipulate the data through given instructions.

2- Reading data from “patient.txt” and store it in “patients”.

2.1- Program creates a Patient object with information given in that line.

2.2- Program adds this patient object to “patients” ArrayList.

3- Reading data from “admission.txt” and store it in “patients”.

3.1- Program checks if this line stars with a number.

3.2- If line stars with a number:

3.2.1- Program creates a Admission object with given admission id.

3.2.2- Program gets the patient from “patients” ArrayList using patient id provided from data file.

3.2.3 Program assigns admission object to patient’s admission field.

3.3- If line doesn’t start with a number:

3.3.1- Program creates an Examination with given inputs and decorates examination with operations using decorator pattern.

3.3.2- Program adds this examination to previous patient’s admission’s “examinations” ArrayList.

4- Reads instructions from “input.txt” and manipulates data.

4.1- Program reads line and with using switch feature on line’s first word, does necessary work.

5- Program updates “patient.txt” with new data.

5.1- Program sorts “patients” ArrayList according to patient IDs.

5.2- For each patient in “patients” ArrayList, program writes a line using patient’s attributes.

6- Program updates “admission.txt” with new data.

6.1- Program creates an ArrayList called “compareArray”.

6.2- Program adds patient from “patients” ArrayList to “compareArray” ArrayList if this patient has an admission.

6.3- Program sorts “compareArray” according to admission IDs of patients.

6.4- Program writes a line using patient’s and patient’s admission’s attributes.

Important Methods:

There are 2 Data Access Classes (named “AdmissionData” and “PatientData”) which contains almost same methods but because of the difference of data files’ format, they differ at some points.

These methods that’s stated as “static” are static because I wanted to use them without creating an object which could be unnecessary sometimes.

1- **static fromFileToArray(String dataFileName, ArrayList<Patient> patients):**

   This method does works that mentioned in Algorithm header’s number 2 and 3. As I mentioned method’s way of doing that differs a little since different formats needs different approaches. Basically this method opens the file, reads line by line and for each line does the necessary job. Both Data Access Class.

   First, this method opens the file named dataFileName. Then to read information from file it creates a BufferedReader with file that has been created.

2- **static fromArrayToFile(String dataFileName, ArrayList<Patient> patients):**

   This method does works that mentioned in Algorithm header’s number 5 and 6. It opens the file named dataFileName and creates a FileWriter to write to file. It clears the text file first since fromFileToArray method does not deletes texts in the text file after reading them.

3- **static readInputWriteOutput(String inputFileName, String outputFileName, ArrayList<Patient> patients):**

   This method belongs to InputOutput Class which is created to deal with inputs. This method opens both input and output files and creates one BufferedReader with input file to read the inputs, one FileWriter with output file to write outputs to “output.txt”. “output.txt” file contains information such as “Patient 33 created”. “output.txt” file does not contains information read from “patient.txt” or “admission.txt”, only contains what “input.txt” file instructs.

4- **cost():**

   This method belongs every class that implements Interface Examination. It returns the cost of the examination. Cost of every examination is integer. Calculating the cost is performed using Decorator Pattern. Each decorator adds its cost to return value of that function. StandartExamination is the basic class of Decorator Pattern which has only examinationType(String) as a field. StandartExamination’s cost method returns different integer values depending on examinatonType. Returns 10 if examinationType is “Inpatient” and returns 15 if examinationType is “Outpatient”.

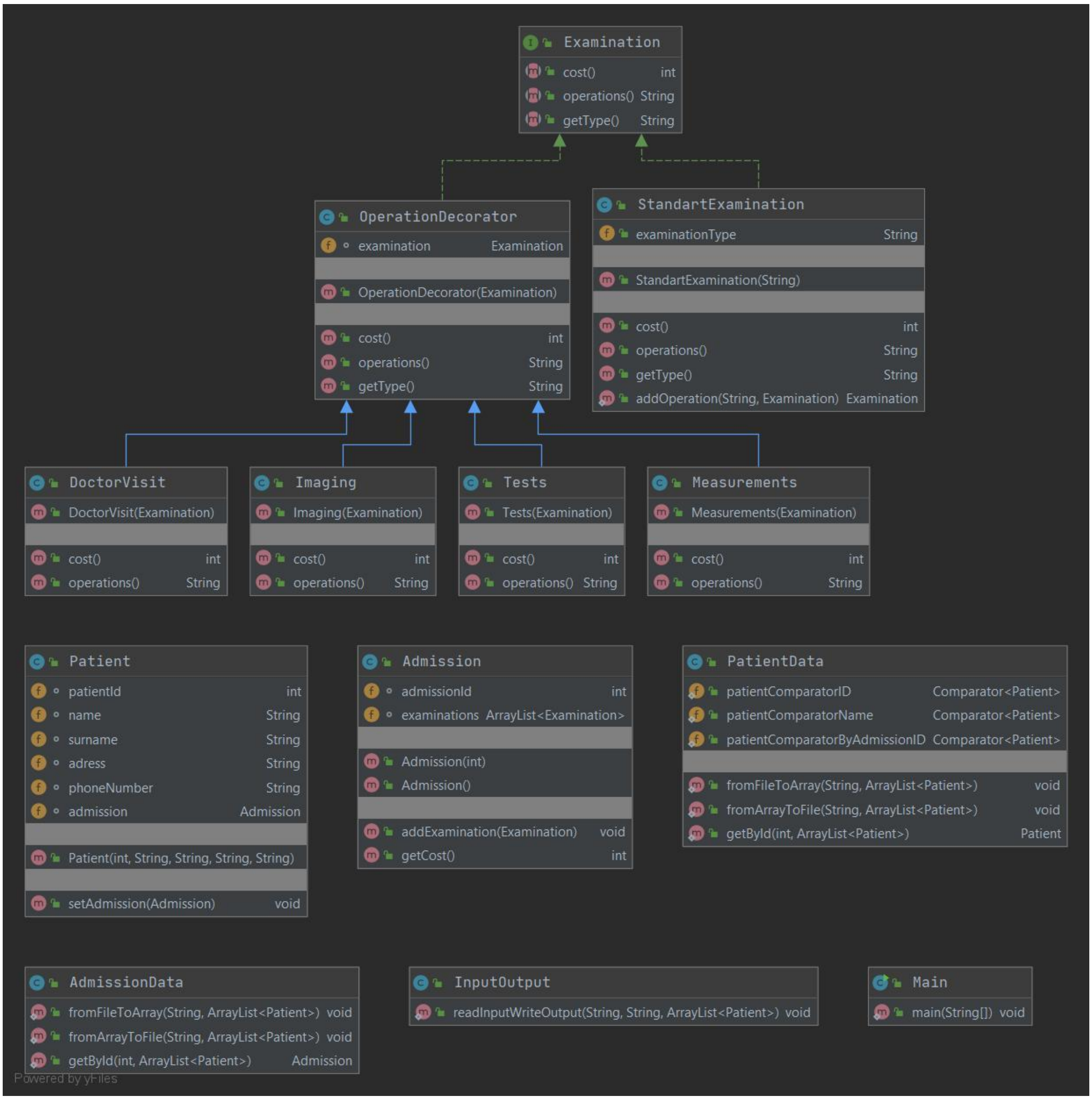
5- operations():

This method returns a string which contains the operation names that has been added to an examination using Decorator Pattern.

6- static addOperation(String operationName, Examination ex):

This method decorates examination ex with an operation using operationName. It uses switch feature to determine the operation.

UML Diagram of the Program:


Examination
cost() int
operations() String
getType() String
OperationDecorator
examination Examination
OperationDecorator(Examination)
cost() int
operations() String
getType() String
StandartExamination
examinationType String
StandartExamination(String)
cost() int
operations() String
getType() String
addOperation(String, Examination) Examination
DoctorVisit
DoctorVisit(Examination)
cost() int
operations() String
Imaging
Imaging(Examination)
cost() int
operations() String
Tests
Tests(Examination)
cost() int
operations() String
Measurements
Measurements(Examination)
cost() int
operations() String
Patient
patientId int
name String
surname String
adress String
phoneNumber String
admission Admission
Patient(int, String, String, String, String)
setAdmission(Admission) void
Admission
admissionId int
examinations ArrayList<Examination>
Admission(int)
Admission()
addExamination(Examination) void
getCost() int
PatientData
patientComparatorID Comparator<Patient>
patientComparatorName Comparator<Patient>
patientComparatorByAdmissionID Comparator<Patient>
fromFileToArray(String, ArrayList<Patient>) void
fromArrayToFile(String, ArrayList<Patient>) void
getById(int, ArrayList<Patient>) Patient
AdmissionData
fromFileToArray(String, ArrayList<Patient>) void
fromArrayToFile(String, ArrayList<Patient>) void
getById(int, ArrayList<Patient>) Admission
InputOutput
readInputWriteOutput(String, String, ArrayList<Patient>) void
Main
main(String[]) void
Powered by yFiles