Assignment3

Name and Surname : Muhammed Enes Yavuz

Identity Number : 21989712

Problem

In this assignment our goal is develop a code about hospitals. In our problem we have three input files we are reading patients information from this text files. After this we update two files according to input.txt and at the end of the code we are creating a new text file named output.txt. In this text we have information according to input.txt. While developing code we should use decorator pattern and data access objects. And I think our codes main task is calculating price for patients.

Algorithm

In this assignment a stored data in lists. I have two important list and my all data is stored in this lists. This lists based on two separate classes and store them as value. First I read two input files with patients information after this I stored them in my lists. According to input file I updated them and end of the code I created the output file.

Solution

My main build is based on Decorator pattern. Most important part of my code an abstract class. This abstract class name is Operation. And it is also super class. Its subclasses are Inpatient, Outpatient and Decorator. Decorator is also an abstract class and its subclasses are Imaging Measurement, Test and Visit. With this structure when I create an new object with one of the Inpatient and Outpatient and one , two or three of Imaging, Measurement, Test and Visit the object calls all classes it has. So I can calculate total price easily. In the second part of my code I use two data access object. PatientFileDao and AdmisionFileDao are Interfaces. PatientDaoImp is implemented from . PatientFileDao and AdmisionDaoImp implemented from AdmisionFileDao. When I run my program main class calls Splitter class. Splitter class calls PatientDaoImp and AdmisionDaoImp . In this classes I read values with ReadFromFile and send values to the two lists. This lists are so important our all data is in this lists. In PatientDaoImp I create list from class Patient. This list store patient objects in it. The patient class have some attributes. In PatientDaoImp I have create a new object from patient and set its attributes based on patient text file. After creation add them up to the my list. I do same things in the AdmissionDaoImp. In this class I created a list based on Examination class and add this examination objects to the my second list. In this class I do some different thing from the previous one. In the Examination class I have a operation list. In this list I store objects from operation class. But in this list I added subclasses of operation class to the list with addOperation method. This structure name is polymorphism actually. And for create wrapped

objects dynamically from string I use forName, newInstance and getConstructor methods. Now we have two lists based on two text file. When we continue at Splitter class we can see I controlled every line of input text file with a for loop. For each type of line I have got a if statement and a method in the PatientDaoImp and AdmisionDaoImp. With help of this methods I updated my lists and I call my values in the lists. I create three String in the splitter class. One of for the output and the other two is for the updated text files. At the and I added all values in my lists to the two string and add some values to output string with if/else statement according to input file. At the final I writed strings to the text files with Writer class.

UML Class Diagram

<<Java Interface>> PatientFileDao
+getByID(String):Patient
+deleteByID(String):void
+add(Patient):void
+getAll():List<Patient>

<<Java Class>> PatientDaoImp
-firstList: String[]
+PatientDaoImp()
+getByID(String):Patient
+deleteByID(String):void
+add(Patient):void
+getAll():List<Patient>

<<Java Class>> Patient
-ID: String
-name: String
-surname: String
-address: String
-phoneNumber: String
+Patient()
+getID():String
+setID(String):void
+getName():String
+setName(String):void
+getSurname():String
+setSurname(String):void
+getAddress():String
+setAddress(String):void
+getPhoneNumber():String
+setPhoneNumber(String):void

<<Java Interface>> AdmissionFileDao
+getAll():List<Examination>
+deleteByID(String):void
+createAdmission(String,String):void
+first():void
+addExamination(String,String,String[]):void
+creator(String,String[]):Operation
+calculator(String):String

<<Java Class>> AdmissionDaoImp
-firstList: String[]
-admissionList: List<Examination>
+AdmissionDaoImp()
+first():void
+createAdmission(String,String):void
+addExamination(String,String,String[]):void
+creator(String,String[]):Operation
+getAll():List<Examination>
+deleteByID(String):void
+calculator(String):String

<<Java Class>> Examination
-operationList: List<Operation>
-admissionID: String
+Examination()
+addOperation(Operation):void
+getOperation():List<Operation>
+getAdmissionID():String
+setAdmissionID(String):void

<<Java Class>> Inpatient
+Inpatient()
+printOperations():String
+cost():int

<<Java Class>> Operation
+Operation()
+printOperations():String
+cost():int

<<Java Class>> Outpatient
+Outpatient()
+printOperations():String
+cost():int

<<Java Class>> Decorator
-operation: Operation
+Decorator(Operation)
+printOperations():String
+cost():int
+setOperation(Operation):void
+getter():Operation

<<Java Class>> Imaging
+Imaging(Operation)
+printOperations():String
+cost():int

<<Java Class>> Visit
+Visit(Operation)
+printOperations():String
+cost():int

<<Java Class>> Measurement
+Measurement(Operation)
+printOperations():String
+cost():int

<<Java Class>> Test
+Test(Operation)
+printOperations():String
+cost():int

<<Java Class>> Splitter
-allPatients: String
-allExaminations: String
-total: String
+Splitter()
+split(String[]):void

<<Java Class>> ReadFromFile
+ReadFromFile()
+readFile(String):String[]

<<Java Class>> Writer
+Writer()
+writer(String,String):void

<<Java Class>> Main
+Main()
+main(String[]):void