HACETTEPE UNIVERSITY

DEPARTMENT OF
COMPUTER ENGINEERING

BBM104 PROGRAMMING LAB.
Assigment 3
Inheritance and Polimorphism

Author
Kayla AKYÜZ
21726914
b21726914@cs.hacettepe.edu.tr

Advisor
Bahar GEZICI
bahargezici@cs.hacettepe.edu.tr

Main.java

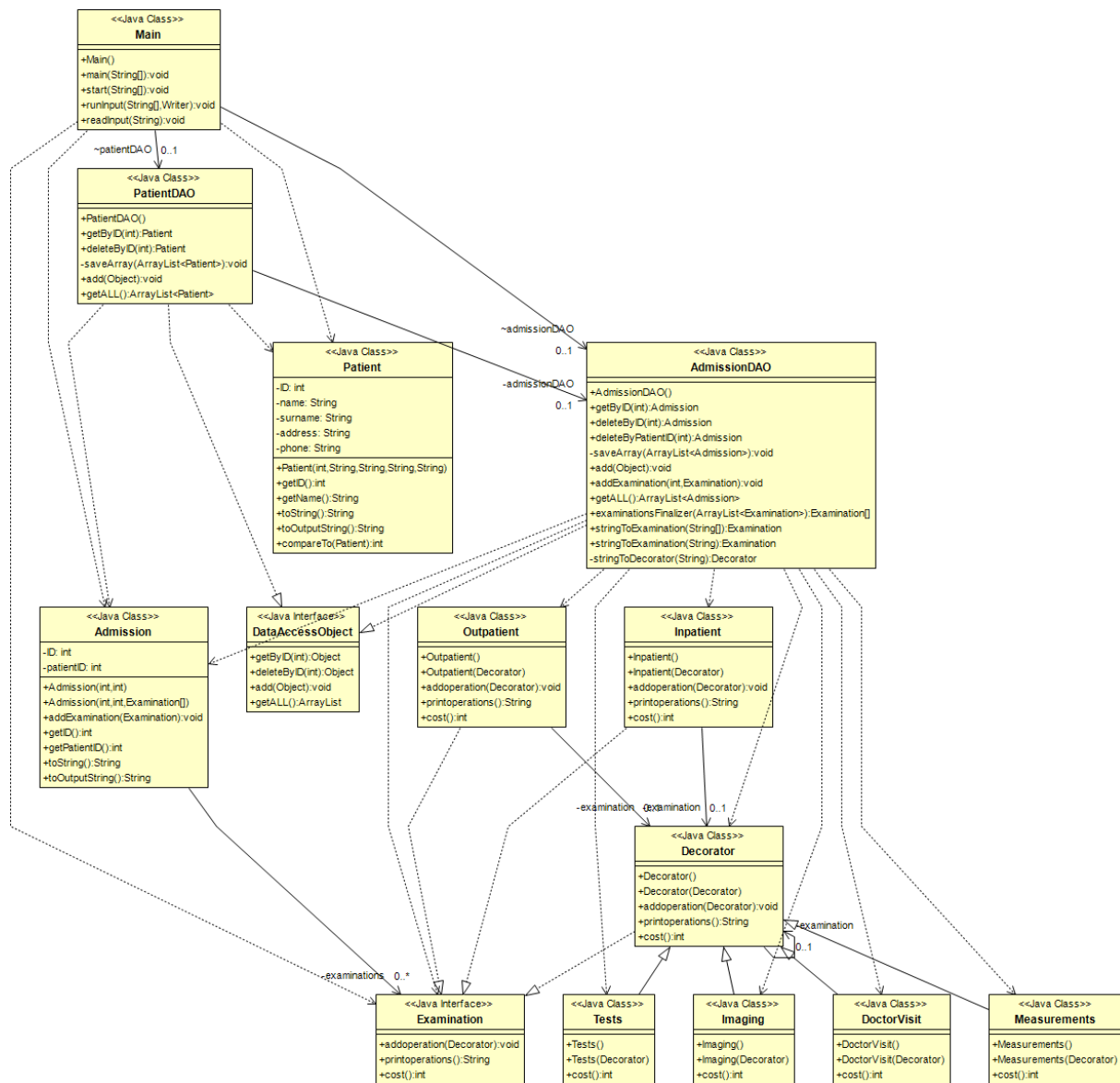May 22,2020

Problem Definition

With the usage of inheritance and polymorphism in java, we are asked to create a hospital patient system with different types of operations and costs. The database will be the txt files and dynamically be accessed by DAO. All the operation will be performed via DAO on txt files. Also when we are reading the database with DAO to create examination structure for calculations and printing we must use decorator pattern.

Algorithm Steps

1. Make initialization:
   1.1 Read input file.
2. Execute runInput() method for every line:
   2.1 Check first word of line to determine action to take.
   2.2 According to action perform one of the below.
      2.2.1.1 Create new patient object.
      2.2.1.2 Using DAO add patient object to database.
      2.2.1.3 Print to output.
      2.2.2.1 Using DAO remove patient object with given ID.
      2.2.2.2 Print to output.
      2.2.3.1 Create new admission object.
      2.2.3.2 Using DAO add admission object to database.
      2.2.3.3 Print to output.
      2.2.4.1 Using DAO generate a examination object.
      2.2.4.2 Using DAO add examination object to admission object with given ID.
      2.2.4.3 Print to output.
      2.2.5.1 Print to output by getting object with DAO and calling .toOutputString method.
      2.2.6.1 Using DAO get an arraylist of patient objects.
      2.2.6.2 Sort the arraylist.
      2.2.6.3 Print to output by calling .toOutputString method on each patient object.
3. Concluding:
   3.1 Close output file.

In order to understand how DAO algorithms work please check the java files. There is no way to summarize them, there are comments in java files.

UML Diagram

<<Java Class>>
Main

+Main()
+main(String[]):void
+start(String[]):void
+runInput(String[],Writer):void
+readInput(String):void

~patientDAO   0..1

<<Java Class>>
PatientDAO

+PatientDAO()
+getByID(int):Patient
+deleteByID(int):Patient
-saveArray(ArrayList<Patient>):void
+add(Object):void
+getALL():ArrayList<Patient>

~admissionDAO
0..1

-admissionDAO
0..1

<<Java Class>>
Patient

-ID: int
-name: String
-surname: String
-address: String
-phone: String

+Patient(int,String,String,String,String)
+getID():int
+getName():String
+toString():String
+toOutputString():String
+compareTo(Patient):int

<<Java Class>>
AdmissionDAO

+AdmissionDAO()
+getByID(int):Admission
+deleteByID(int):Admission
+deleteByPatientID(int):Admission
-saveArray(ArrayList<Admission>):void
+add(Object):void
+addExamination(int,Examination):void
+getALL():ArrayList<Admission>
+examinationsFinalizer(ArrayList<Examination>):Examination[]
+stringToExamination(String[]):Examination
+stringToExamination(String):Examination
-stringToDecorator(String):Decorator

<<Java Class>>
Admission

-ID: int
-patientID: int

+Admission(int,int)
+Admission(int,int,Examination[])
+addExamination(Examination):void
+getID():int
+getPatientID():int
+toString():String
+toOutputString():String

<<Java Interface>>
DataAccessObject

+getByID(int):Object
+deleteByID(int):Object
+add(Object):void
+getALL():ArrayList

<<Java Class>>
Outpatient

+Outpatient()
+Outpatient(Decorator)
+addoperation(Decorator):void
+printoperations():String
+cost():int

<<Java Class>>
Inpatient

+Inpatient()
+Inpatient(Decorator)
+addoperation(Decorator):void
+printoperations():String
+cost():int

-examination   -examination   0..1

<<Java Class>>
Decorator

+Decorator()
+Decorator(Decorator)
+addoperation(Decorator):void
+printoperations():String
+cost():int

-examination
0..1

-examinations   0..*

<<Java Interface>>
Examination

+addoperation(Decorator):void
+printoperations():String
+cost():int

<<Java Class>>
Tests

+Tests()
+Tests(Decorator)
+cost():int

<<Java Class>>
Imaging

+Imaging()
+Imaging(Decorator)
+cost():int

<<Java Class>>
DoctorVisit

+DoctorVisit()
+DoctorVisit(Decorator)
+cost():int

<<Java Class>>
Measurements

+Measurements()
+Measurements(Decorator)
+cost():int

REFERENCES

Assignment Paper
LaTex Tutorials