Midterm Exam

Question 1:

1. An instance variable belongs to the class in which it is declared.
A. True because all the fields and methods are called Class members.

2. A constructor is a method that never returns anything, in fact it has void return type.
A. False. A constructor doesn't have a return type. The main purpose of the constructor is to instantiate the instance variables of a class.

3. A constructor is called the "default constructor", if it has no parameters.
A. True. A constructor has two types. Default(non-parameterized) and Parameterized.

4. A constructor is a method and therefore can be used through the dot operator, applied to an instance of the class where it was declared.
A. False. A constructor is a special type of method in a class that is invoked only once during the instantiation of the object. That is when using the new keyword with class name.

5. The following instance variable x is correctly used:

```
Point p1 = new Point();
Point.x = 10;
```

A. False. As x is an instance variable, it should be used by the object p1 and not the class name. Class name is used for static variables.

6. If a method is declared static, it cannot be called outside of its package.
A. False. Static is not an access modifier. Hence, we cannot determine whether it can be called outside its package or not.

7. If we don't prefix modifiers to a method, the method is only accessible within the same package.
A. True. The method will be set as default which provides the access in the same package.

8. An instance initializer is invoked before constructors.
A. True. Instance Initialization blocks are executed before the constructors in a top-down approach.

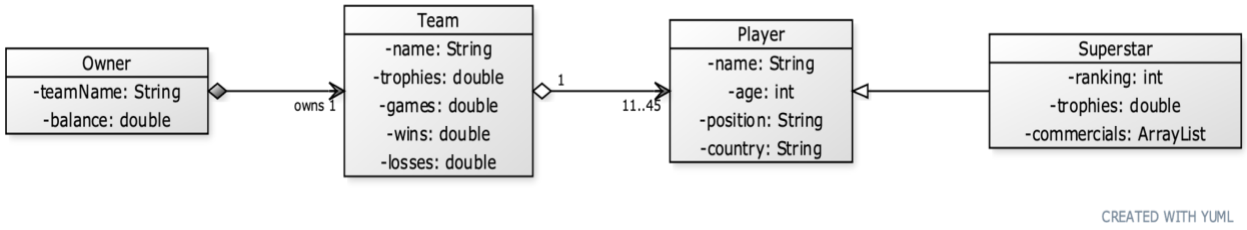
9. The keyword this allows you to reference the members of an object that will be created only at runtime within the object itself.
A. True because objects are created during the execution, and "this" is used to refer the instance members (mainly when there is a namespace collision). It cannot be used for static members.

10. The keyword this is always optional.
A. False. this keyword is required when there is a namespace collision.

Question 2:

Aggregation vs Composition:

- ➔ Aggregation is represented by a straight line with the null-diamonded arrow mark whereas Composition is represented by a straight line with the full diamond arrow mark at one end.
- ➔ Aggregation is a subset of the Association whereas Composition is the subset of the Aggregation
- ➔ In aggregation, the connection between the two classes is subtle, but it is strong in composition.
- ➔ In aggregation, the related objects are present independently with in the entire system whereas the objects are dependent on each other in composition.
- ➔ Deletion of one element doesn't impact the other elements in aggregation whereas the deletion of one object impacts the presence of others in composition.
- ➔ Aggregation can be derived by considering "has-a", "part of". In other hand, Composition can be derived from "has-a", "part-of" and "belongs-to".



Question 3:

[1] Second2

[2] Third2

[3] Fourth2

[4] Third2

[5] Second2

[6] Third2

[7] Fourth2

[8] Third2

[9] error

[10] Third1/Second2

Question 4:

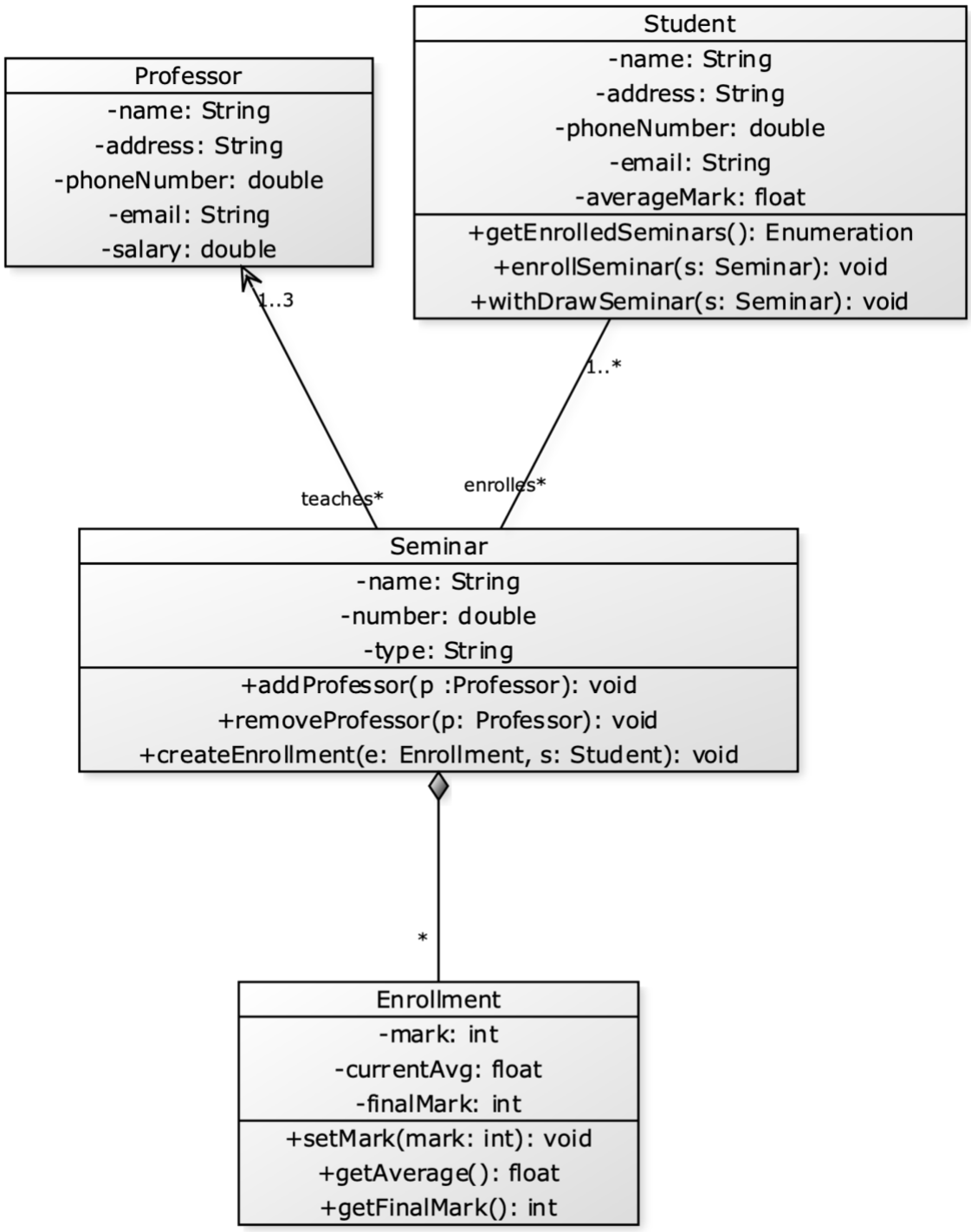
Source files

- ➔ City.java: src/question4/City.java
- ➔ Nation.java: src/question4/City.java
- ➔ Question2main: src/question4/Question2Main.java

Question 5:

Motivation:

- ➔ As stated in the question, we must construct a professor class with the following attributes: name, address, phone number, email address, and salary. I set the access modifier for all attributes in the class to private.
- ➔ Like the Professor class, I have created a student class with similar attributes, expect salary, and added an additional attribute, average, that holds the average marks of all his or her final marks in the seminars.
- ➔ A seminar class is defined by its name, number, and type property. The "type" property specifies if the seminar is a bachelor's or a master's.
- ➔ Moreover, an enrollment class is created that contains the mark, current average, and final mark of the seminar for each student.
- ➔ There is a bidirectional relationship between student and seminar since the student can enroll in 0 or many seminars, and the seminars are made up of 1 or more enrolled students.
- ➔ Depending on the type of seminar, a student may potentially withdraw from the registered seminar. A student cannot withdraw from the bachelor's seminar; however, a student may withdraw from the master's seminar. We can implement the logic in the withDrawSeminar() method since every seminar has the type attribute.
- ➔ In the seminar, we need to make an enrollment for each student, and the seminar has a whole part of ownership with enrollment, which results in composition between them. Because when the seminar object is destroyed (cancelled), the enrollment in the seminar should also be deleted.
- ➔ Finally, a professor can teach multiple seminars, but it is not necessary to have a seminar in its implementation. Likewise, each seminar has at least one and at most 3 professors which corresponds to the cardinality for Seminar.


Professor
-name: String
-address: String
-phoneNumber: double
-email: String
-salary: double
Student
-name: String
-address: String
-phoneNumber: double
-email: String
-averageMark: float
+getEnrolledSeminars(): Enumeration
+enrollSeminar(s: Seminar): void
+withDrawSeminar(s: Seminar): void
1..3
1..*
teaches*
enrolles*
Seminar
-name: String
-number: double
-type: String
+addProfessor(p :Professor): void
+removeProfessor(p: Professor): void
+createEnrollment(e: Enrollment, s: Student): void
*
Enrollment
-mark: int
-currentAvg: float
-finalMark: int
+setMark(mark: int): void
+getAverage(): float
+getFinalMark(): int
CREATED WITH YUML

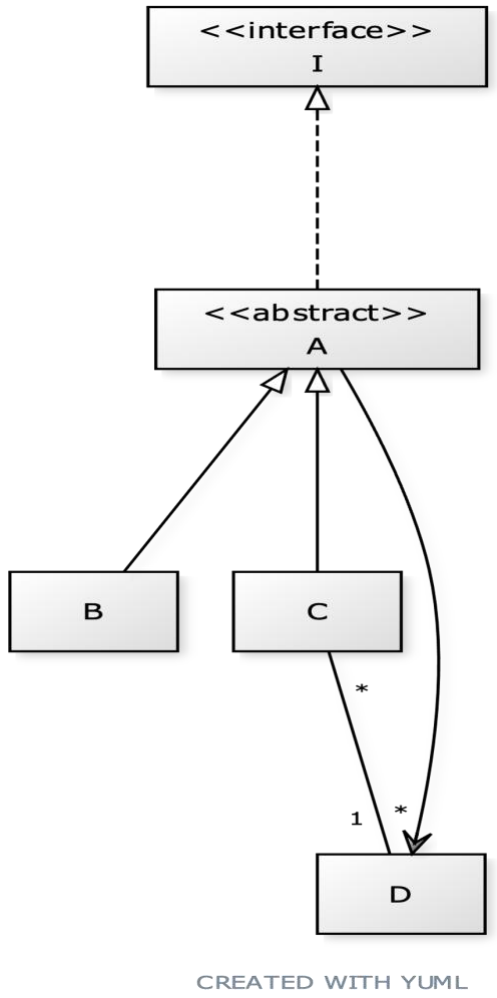
Question 6:

Part A:

Source files

- ➔ Account.java: src/question6/Account.java
- ➔ AccountHolder.java: src/question6/AccountHolder.java
- ➔ CorporateHolder.java: src/question6/CorporateHolder.java
- ➔ IndividualHolder.java: src/question6/IndividualHolder.java

Part B:



Question 7:

Source files

- ➔ GradStudent.java: src/question7/GradStudent.java
- ➔ Student.java: src/question7/Student.java