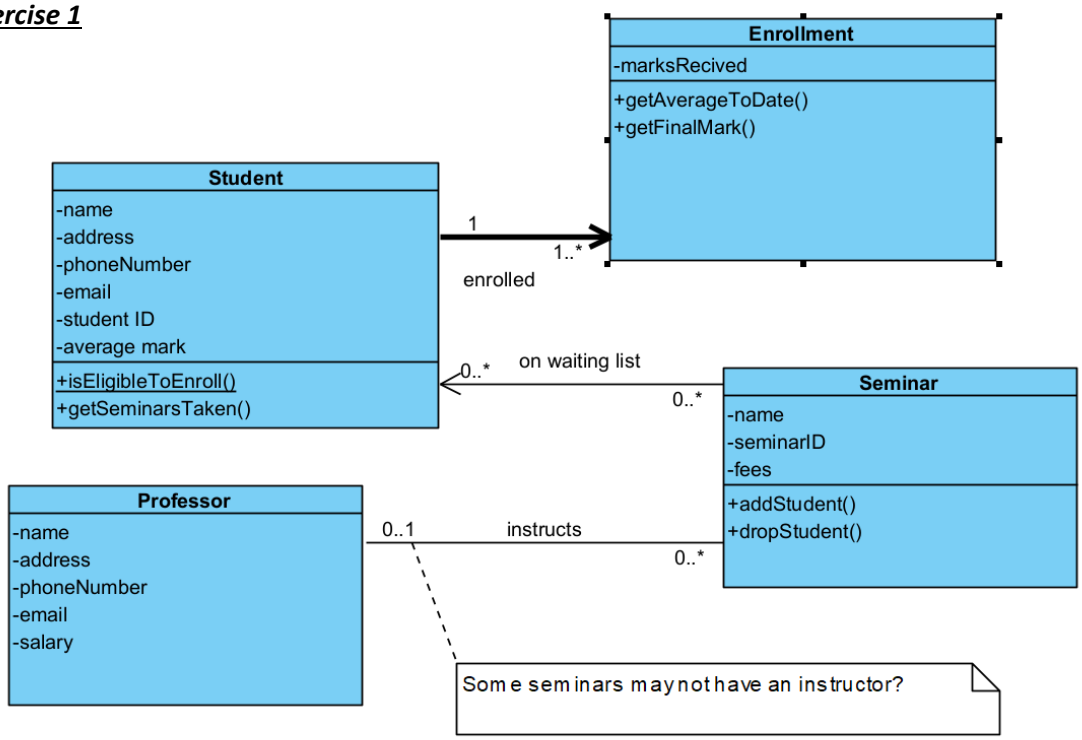
**Student :** Popa Ioan-Ciprian **Subgroup :** 5

**Homework**

**Lab 5**

***Exercise 1***




**Student**

- Attributes:name, address, phoneNumber, email, studentID, average mark
- Operations
  - +isEligibleToEnroll() – static function
  - +getSeminarsTaken() - function

**Professor**

- Attributes: name, address, phoneNumber, email, salary

**Seminar**

- Attributes: name, seminarID, fees
- Operations, addStudent(student) , dropStudent(student) -

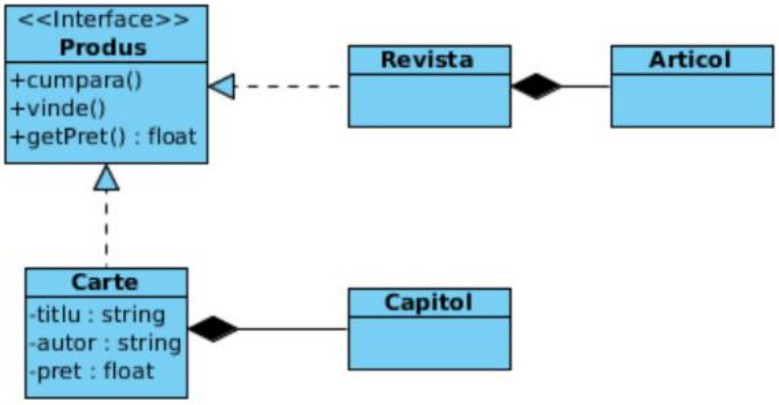
**Relationships**

- A student can enroll in zero or more seminars (0...*).
- A seminar can have zero or more students (0…*).

- A seminar can have zero or one professor (0...1) - This means some seminars may not have an instructor assigned yet.
- A professor instructs zero or more seminars (0...*) - This means a professor can teach multiple seminars.
- One or more enrollments (1…*) have only one student enrolled.

***Exercise 2***

***1.***




Check the complete and correct description of the relations of class Carte:

(a) Generalization between class Carte(subclass) and class Produs(superclass); composition between classes \Carte(composite) and Capitol(component).

(b) Generalization between class Carte subclass) and interface Produs (superclass); composition between classes Carte (composite) and Capitol(component).

(c) Realization between class Carte and class Produs; class Carte implements class Produs; between classes Carte(composite) and Capitol(component).

(d) Realization between class Carte and interface Produs ; class Carte implements interface Produs; composition between classes Carte(composite) and Capitol(component).

(e) Realization between class Carte and interface Produs ; class Carte implements interface Produs; composition between classes Carte(component) andCapitol (composite).

Why: D because it uses the correct relationship terms for class Carte and Produs and the correct type for Produs – interface and well the composition between Carte and Capitol

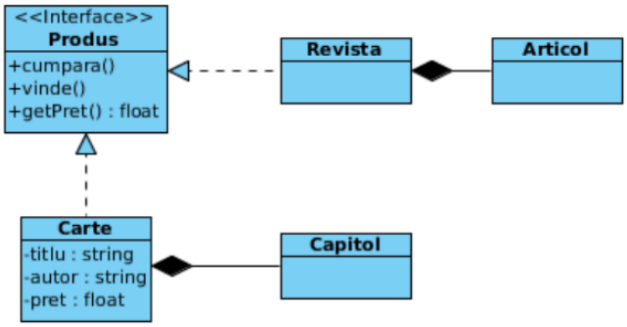
E – not correct because the types for carte and produs are switched

A - Is not correct because It does not correctly identified Produs as an interface and the correct relationships

B - a class implements an interface and not the other way around

C - there is the correct relationship but Produs is not an interface

***2.***




Which valid sequences of Java code result from the diagram for class Revista?

(a) class Revista extends Produs{...}

(b) class Produs implements Revista{...}

(c) private Collection<Articol> capitole = new Collection();

(d) public float getPret();

(e) protected float getPret();

(f) class Revista implements Produs{...}

(g) public Produs vinde(){...}

Why:

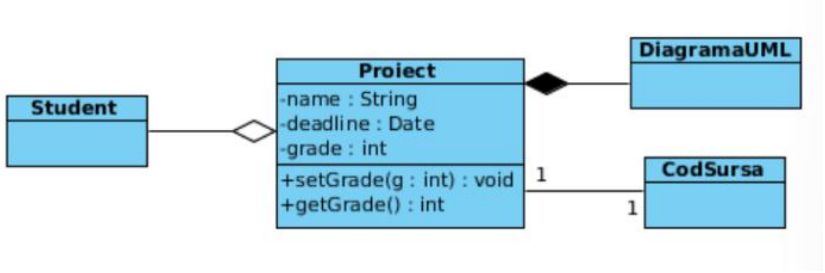
A -not good because Revista should implement Produs, as it is an interface and not an abstract class

B - a class implements an interface and not the other way around

D and E - not good cause we dint have specified an getter inside the UML

G - incorrect because vinde() is a method in the Produs interface, and we cannot instantiate an interface or return an interface type from a method without context to suggest that it is possible.

***3.***




Which sequences of Java code are valid for class Proiect?

(a) private CodSursa theCode;

(b) private grade int;

(c) private Collection<DiagramaUML> diagrams = new Collection();

(d) public Date deadline;

(e) public getGrade(){...}

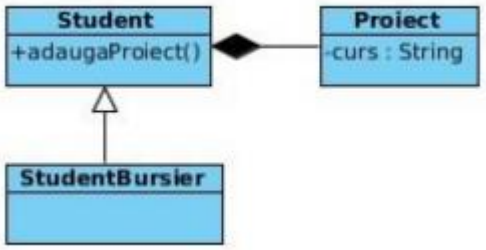
(f) class Proiect extends Student{...}

(g) public void setGrade(int g){...}

(h) private DiagramaUML diagram;

Why:

4.




Select the true statements?

(a) Class Student inherits from class StudentBursier

(b) An object of type Student contains a collection of objects of type Proiect.

(c) Class Proiect has a public attribute of type String.

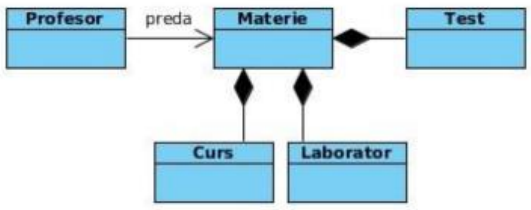
(d) Class Student has the public operation adaugaProiect.

(e) Class Student has the private operation adaugaProiect.

(f) Class Student is superclass for the class StudentBursier.

Why:

5.



Which sequence of Java code correctly describes the relationship between classes Profesor and Materie?

(a) class Profesor extends Materie{...}

(b) class Profesor {

private Materie preda; ...}

(c) class Materie {

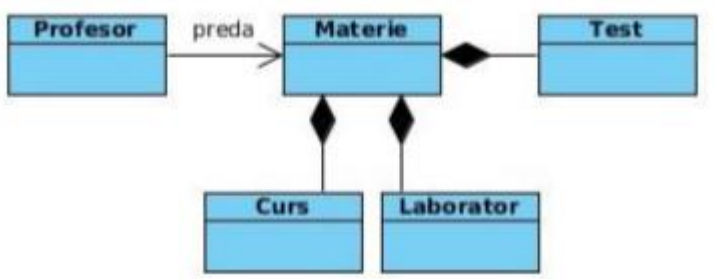
private Profesor preda; ...}

(d) class Materie {

private Collection<Materie> preda;...}

Why:

6.




Select the true statements?

(a) Class Materie defines a composition of objects of type Curs.

(b) A bidirectional association exists between class Profesor and class Materie.

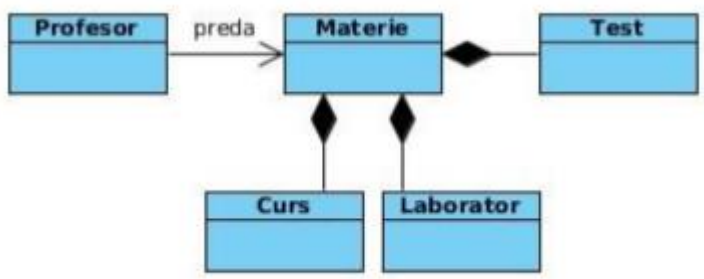
(c) Class Test inherits from class Materie.

(d) Class Materie defines an aggregate of objects of type Laborator.

(e) An object of type Materie contains a collection of objects of type Test.

Why:

7.



Which sequence of Java code correctly and completely describes the relationship between class Materie and class Laborator?

(a) class Materie extends Laborator{...}

(b) class Laborator extends Materie{...}

(c) class Materie {

private Collection <Laborator> laboratoare = new Collection();...}

class Laborator {

private Materie materie;

...}

(d) class Materie {

private Laborator laborator;...}

class Laborator {

private Collection<Materie> materie;

...}

(e) class Materie {

private Collection <Laborator> laboratoare;...}

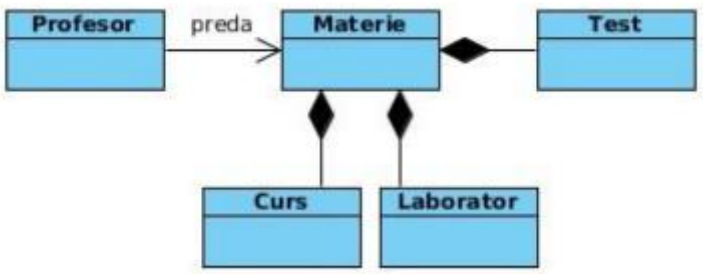
class Laborator {

private Materie materie;

...}

Why:

8.




Select the complete and correct description of the relations represented on the diagram:

(a) Association between classes Profesor and Materie; aggregation between classes

Materie(aggregate) and Test(component), Materie(aggregate) and Laborator(component), Materie(aggregate) and Curs(component).

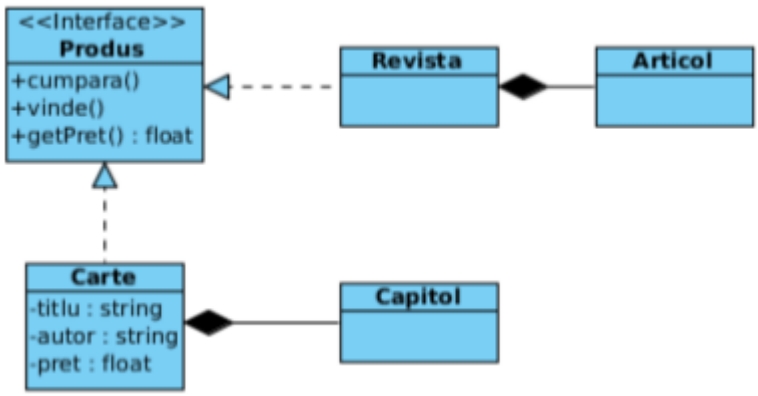
(b) Unidirectional association, named preda, from class Profesor to class Materie; aggregation between classes Materie(aggregate) and Test(component), Materie(aggregate) and Laborator(component), Materie(aggregate) and Curs(component).

(c) Unidirectional association, named preda, from class Profesor to class Materie; class Materie is superclass for classes Test, Laborator and Curs.

(d) Unidirectional association, named preda, from class Profesor to class Materie; composition between class Materie(composite) and class Curs(component); composition between class Materie(composite) and class Laborator(component); composition between class Materie(composite) and class Test(component).

(e) Unidirectional association, named preda, from class Profesor to class Materie; classes Curs, Laborator and Test inherit from class Materie.
Why:

9.




Which sequences of Java code are valid for class Carte?

(a) class Carte extends Produs{...}

(b) private float pret;

(c) private pret float;

(d) public float getPret();

(e) class Carte implements Produs{...}

(f) private float pret(){...}

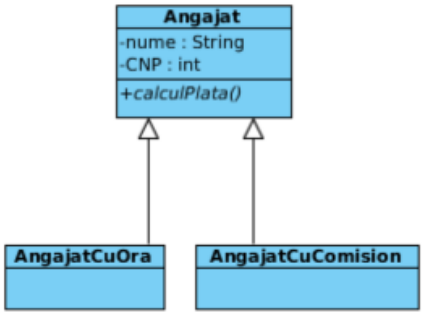
(g) public float getPret(){...}

(h) public Produs cumpara(){...}

(i) private Collection<Capitol> capitole = new Collection();

Why:

10.




Which sequence of Java code correctly and completely defines what results from the diagram for class Angajat?

(a) class Angajat {

private String nume;

private int CNP;

public abstract void calculPlata();

...}

(b) abstract class Angajat {

private String nume;

private int CNP;

public abstract void calculPlata();

...}

(c) abstract class Angajat {

private String nume;

private int CNP;

public abstract void calculPlata(){...};

...}

(d) abstract class Angajat {

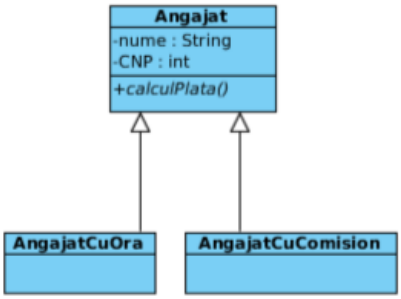
private String nume;

private int CNP;

public void calculPlata();

...}
Why:

11.




Which sequences of Java code are valid for class AngajatCuOra?

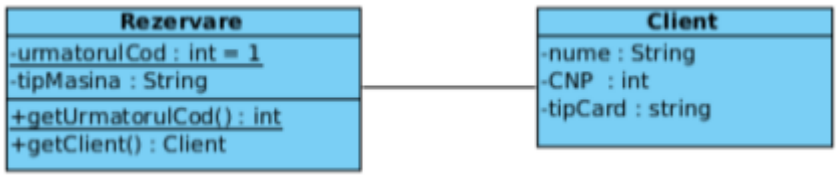
(a) class AngajatCuOra extends Angajat{...}

(b) class AngajatCuOra implements Angajat{...}

(c) public calculPlata();

(d) public calculPlata(){...};
Why:

12.




(a) class Rezervare extends Client{...}

(b) private int urmatorulCod = 1;

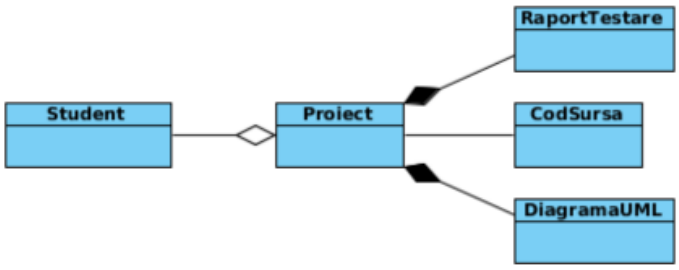
(c) private static int urmatorulCod = 1;

(d) public static int getUrmatorulCod(){...};

(e) private Client client;

(f) public static Client getClient(){...};
Why:

13.




Select the true statements.

(a) Class Student defines an aggregate of objects of type Proiect; classes RaportTestare and DiagrameUML define compositions of objects of type Proiect.

(b) Class Proiect defines compositions of objects of type Student, of objects of type DiagrameUML and of objects of type RaportTestare.

(c) Class Proiect defines an aggregate of objects of type Student and compositions of objects of type DiagrameUML and of objects of type RaportTestare.

(d) Class Proiect defines a composition of objects of type Student and aggregates of objects of type DiagrameUML and of objects of type RaportTestare.

(e) Class Proiect has an association relationship with class CodSursa.
Why: