OOP WS2020/21

EXERCISE 4

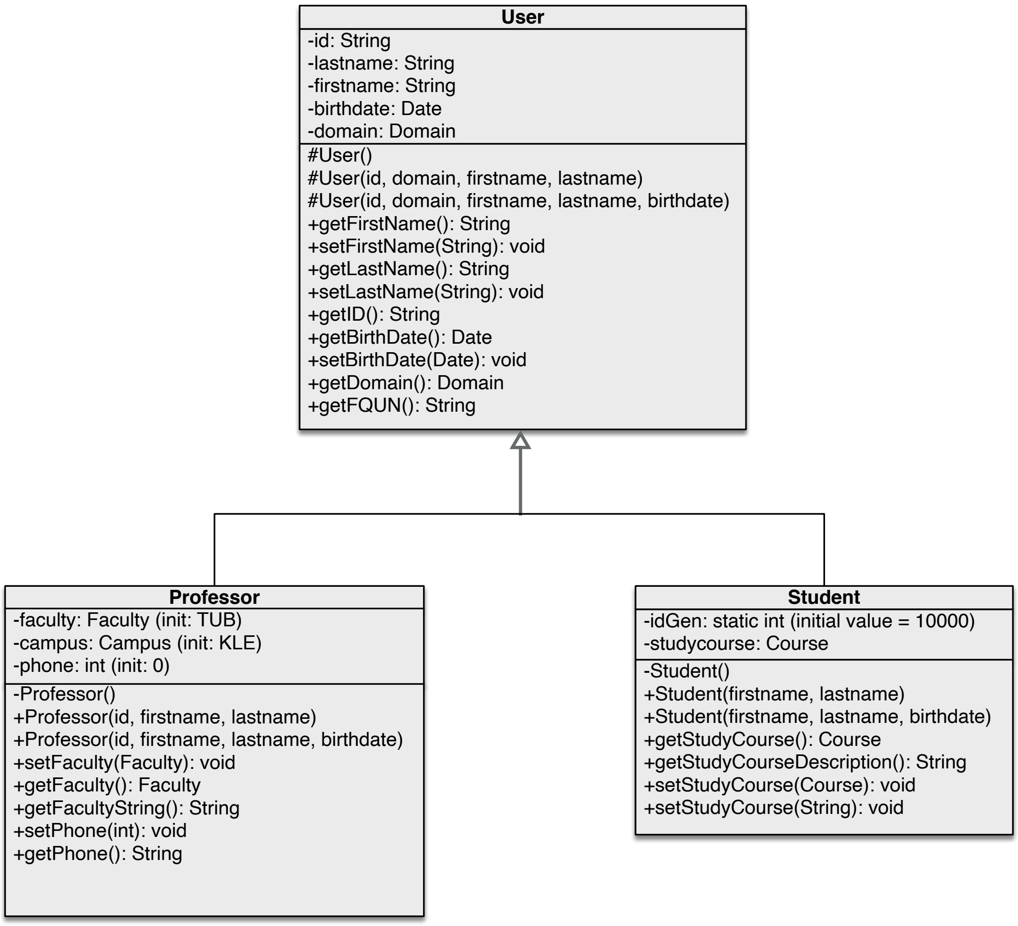
1. Extend the Student class implementation from exercise 3 with the Professor and User classes using generalization as depicted in the UML-Diagram below.

   The **idGen** has value 10000 initially and keep incrementing for every new student object instantiated. The student's id is assigned during instantiation using current **idGen** value before increment. The Course is data type enumeration with the following content: ME, MSE, EL, IE, BMS, SCB, MME, MBB. This datatype already provided in the project folder. It is also possible to set the studycourse using a string value like "IE" or "EL", etc. The study course description will return a complete human readable study course, i.e., ME = "*Mechanical Engineering, B.Sc.*", MME="*Mechanical Engineering, M.Sc.*", MBB="*Bionics, M.Sc.*", (plese refer to actual study program in our faculty: https://www.hochschule-rhein-waal.de/en/faculties/technology-and-bionics/degree-programmes).

   The Domain is data type enumeration with the following content: STAFF, STUDENT. The value domain cannot be set from anywhere, but it will be derived from the class where the instance belongs to, i.e., Professor will have a domain value of STAFF and Student will have a domain value of STUDENT.

   The Faculty is data type enumeration with the following content: TUB, LS, GO, KU. The Campus is data type enumeration with the following content: KLE, KL. Similar like domain, campus value cannot be set manually but its value will be derived from the faculty value, i.e., (KLE for TUB, LS, and GO) and (KL for KU).
   The getFQUN() method will return a full qualified user name that is a concatenation of the "id@staff.hsrw" (for STAFF) and "id@student.hsrw" (for STUDENT).

   The method setPhone() in the Professor class accepts only the extension of the phone number (three or four digits)
   The method getFacultyString() will return the name of the faculty in string, i.e., TUB is "Technology and Bionics", LS = "Life Sciences", GO = "Society and Economics" and "KU" = "Communication and Environment".
   The method getPhone() will return a full phone number in a string format, this value depends on the campus location, i.e., KLE will return "+49 2821 806 73 XXX" and KL will return "+49 2842 908 XXX", where XXX is the three or four digits number from the phone variable.
   Implement the student class in Student.java and create all necessary java files by yourself. (Hints: use java.util.Date for the Date)


User
-id: String
-lastname: String
-firstname: String
-birthdate: Date
-domain: Domain
#User()
#User(id, domain, firstname, lastname)
#User(id, domain, firstname, lastname, birthdate)
+getFirstName(): String
+setFirstName(String): void
+getLastName(): String
+setLastName(String): void
+getID(): String
+getBirthDate(): Date
+setBirthDate(Date): void
+getDomain(): Domain
+getFQUN(): String
Professor
-faculty: Faculty (init: TUB)
-campus: Campus (init: KLE)
-phone: int (init: 0)
-Professor()
+Professor(id, firstname, lastname)
+Professor(id, firstname, lastname, birthdate)
+setFaculty(Faculty): void
+getFaculty(): Faculty
+getFacultyString(): String
+setPhone(int): void
+getPhone(): String
Student
-idGen: static int (initial value = 10000)
-studycourse: Course
-Student()
+Student(firstname, lastname)
+Student(firstname, lastname, birthdate)
+getStudyCourse(): Course
+getStudyCourseDescription(): String
+setStudyCourse(Course): void
+setStudyCourse(String): void

2. Design a class named **Triangle** that extends **GeometricObject**. (see lecture slide for the complete description of the GeometricObject class. The class contains:
   - Three **double** data fields named **side1**, **side2**, and **side3** with default values **1.0** to denote three sides of a triangle.
   - A no-arg constructor that creates a default triangle.
   - A constructor that creates a triangle with the specified **side1**, **side2**, and **side3**.
   - The accessor and mutator methods for all three data fields.
   - A method named **getArea()** that returns the area of this triangle.
   - A method named **getPerimeter()** that returns the perimeter of this triangle.
   - A method named **toString()** that returns a string description for the triangle.

   For the formula to compute the area of a triangle, refer to your school math knowledge. The **toString()** method is implemented as follows:
   return "Triangle: side1 = " + side1 + **" side2 = "** + side2 + **" side3 = "** + side3;

   Implement the **Triangle** and **GeometricObject** classes, place the files on the same folder as question no 1.
   Write a test program that prompts the user to enter three sides of the triangle, a color, and a Boolean value to indicate whether the triangle is filled. The program should create a **Triangle** object with these sides and set the **color** and **filled**

properties using the input. The program should display the area, perimeter, color, and true or false to indicate whether it is filled or not.

3. Refactor the **RegularPolygon** class from exercise 3 to be a subclass of **GeometricObject**. The following is the previous description of this class.
   In an *n*-sided regular polygon, all sides have the same length and all angles have the same degree (i.e., the polygon is both equilateral and equiangular). Design a class named **RegularPolygon** that contains:
   - A private **int** data field named **n** that defines the number of sides in the polygon with default value **3**.
   - A private **double** data field named **side** that stores the length of the side with default value **1**.
   - A private **double** data field named **x** that defines the *x*-coordinate of the polygon's center with default value **0**.
   - A private **double** data field named **y** that defines the *y*-coordinate of the polygon's center with default value **0**.
   - A no-arg constructor that creates a regular polygon with default values.
   - A constructor that creates a regular polygon with the specified number of sides and length of side, centered at (**0**, **0**).
   - A constructor that creates a regular polygon with the specified number of sides, length of side, and *x*- and *y*-coordinates.
   - The accessor and mutator methods for all data fields.
   - The method **getPerimeter()** that returns the perimeter of the polygon.
   - The method **getArea()** that returns the area of the polygon. The formula for computing the area of a regular polygon is

$$Area = \frac{n \times s^2}{4 \times tan(\frac{\pi}{n})}$$

   The **toString()** method is implemented as follows **return "RegularPolygon: number of sides = "** + n + **" side length = "** + side;

   Draw a complete UML diagram that includes this class, **Triangle** class and **GeometricObject** class. Implement this class in the same folder as previous questions. Upload the UML diagram in the UML folder.

   Write a test program that prompts the user to enter number of sides and their length of the polygon, a color, and a Boolean value to indicate whether the polygon is filled. The program should create a **RegularPolygon** object with these sides properties and set the **color** and **filled** properties using the input. The program should display the area, perimeter, color, and true or false to indicate whether it is filled or not.