OOP WS2020/21

EXERCISE 3

1. Implement Student class 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). Implement this class in Student.java. (Hints: use java.util.Date for the Date)

Student
-id: int -idGen: static int (initial value = 10000) -lastname: String -firstname: String -birthdate: Date -studycourse: Course
-Student() +Student(firstname, lastname) +Student(firstname, lastname, birthdate) +getFirstName(): String +setFirstName(String): void +getLastName(): String +setLastName(String): void +getID(): int +getBirthDate(): Date +setBirthDate(Date): void +getStudyCourse(): Course +getStudyCourseDescription(): String +setStudyCourse(Course): void +setStudyCourse(String): void

2. Design a class named **Fan** to represent a fan. The class contains:
   - Three constants named **SLOW**, **MEDIUM**, and **FAST** with the values **1**, **2**, and **3** to denote the fan speed.
   - A private **int** data field named **speed** that specifies the speed of the fan (the default is **SLOW**).
   - A private **boolean** data field named **on** that specifies whether the fan is on (the default is **false**).

- A private **double** data field named **radius** that specifies the radius of the fan (the default is **5**).
- A string data field named **color** that specifies the color of the fan (the default is **blue**).
- The accessor and mutator methods for all four data fields. Please remember for boolean accessor the method start with "is" not "get".
- A no-arg constructor that creates a default fan.
- A method named **toString()** that returns a string description for the fan. If the fan is on, the method returns the fan speed, color, and radius in one combined string, e.g. "fan speed is medium, fan color is yellow, fan radius is 5.0". If the fan is not on, the method returns the fan color and radius along with the string "fan is off" in one combined string, e.g. "fan color is blue, fan radius is 9.0, fan is off"

Draw the UML diagram for the class then implement the class. Upload the UML diagram in the UML folder. Write a test program that creates two **Fan** objects. Assign maximum speed, radius **10**, color **yellow**, and turn it on to the first object. Assign medium speed, radius **5**, color **blue**, and turn it off to the second object. Display the objects by invoking their **toString** method. Create two java files, one for the fan and the other for test program. Place the files in the same folder as Student.java in question 1.

3. 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

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

Draw the UML diagram for the class then implement the class. Upload the UML diagram in the UML folder. Write a test pro- gram that creates three **RegularPolygon** objects, created using the no-arg con- structor, using **RegularPolygon(6, 4)**, and

using **RegularPolygon(10, 4, 5.6, 7.8)**. For each object, display its perimeter and area. Create two java files, one for the polygon and the other for test program. Place the files in the same folder as Student.java in question 1.