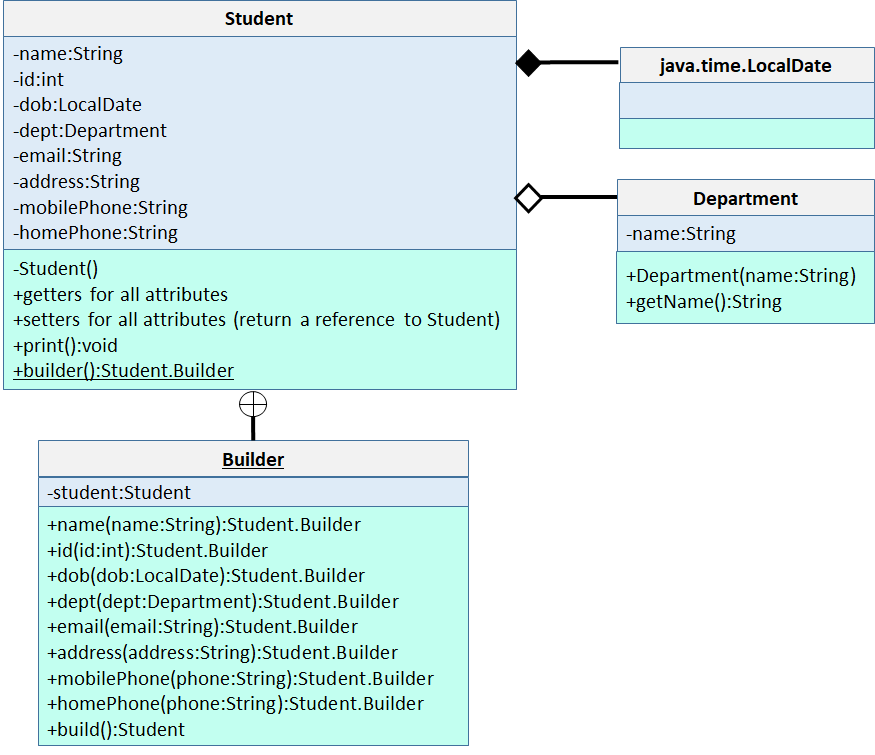
Lab 15

**Rules:**

* You are asked to implement two files: **Department.java** and **Student.java** as described below.
* Do not forget to take your work with you when you leave the lab by either copying your work files to your own USB flash disk, or by e-mailing them to yourself.

Here are the UML class diagrams of the classes you will be implementing in this Lab and how they relate to each other:



We have a **Student** class with many attributes. Any Student object **must** have a name, an id > 0, and a date of birth. All the other fields are optional. Because the Student class has many attributes and most of them are optional, we will be using the **Builder design pattern** to create a Student as shown in the UML diagram. The Student class must have a **public static Builder** class with the given attributes. The circle with a cross that connects Student to the Builder class indicates that Builder is an inner class of Student. In order to create a Student, we will first call Student.builder(), which is a public static method, to create a Builder object. We use the Builder object to set the fields of the students we want to set. Finally, we call the **build()** method of the Builder object to get the Student object initialized so far. Notice that all setter methods of the Builder class return a reference to itself so that we can use method chaining. Look at Test.java to see how it works.

The Student class must have the usual getter and setter methods for ALL attributes. Make sure that the setter methods return a reference to the current Student object for method chaining. Look at Test.java to see how method chaining works with Student objects.

Here are the validations you have to perform:

1. When id() is called to set the id of the student, make sure that the user-supplied id >=1. Otherwise, throw an IllegalArgumentException.
2. When name() is called to set the name of the student, make sure that name is null and has at least 2 characters.
3. When build() is called to get the final student, make sure that the initialized student has its name, id and department fields set. Otherwise, return a “null” object.

**Department** is an **enum** with the following **enum constants** and their values. First, implement this enum class with the given attributes and values.

|  |  |
| --- | --- |
| **Enum Constant** | **Value** |
| NONE | No Department |
| CSCI | Computer Science |
| CHEM | Chemistry |
| PHYSICS | Physics |
| BIO | Biology |
| GEO | Geography |

When the Student object is first created via the no-args constructor, set the id to 0, name to null, and dept to NONE. When build() of the Builder is called, you can check if these attributes have been initialized properly. If not, you can simply return a “null” object. If they are initialized properly, return the created object.

The **print()** method of the **Student** class is supposed to print the valid attributes of the Student in a nice table. Look at the output of our Test code to see how your output should look like.

To test your classes, we are giving you a driver code (**Test.java**) that creates 3 students. Here is the expected output of your code when run against **Test.java**:

student1.name: John Doe

student1.id: 1

student1.dob: 1999-02-15

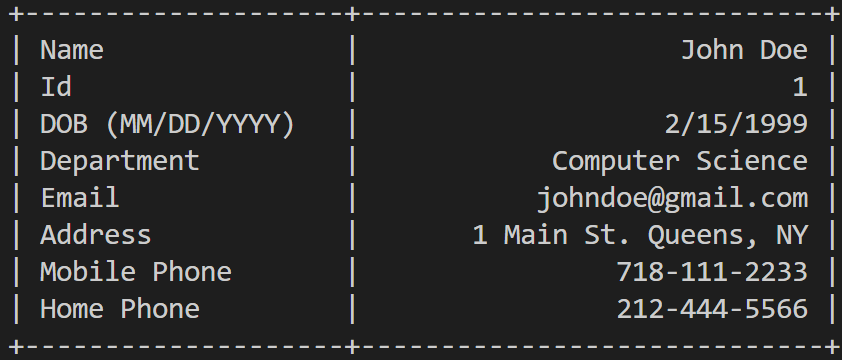
student1.department: Computer Science

student1.email: johndoe@gmail.com

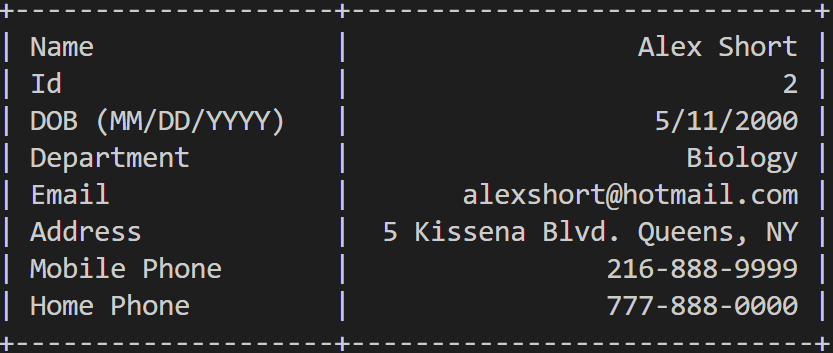
student1.address: 1 Main St. Queens, NY

student1.mobilePhone: 718-111-2233

student1.homePhone: 212-444-5566



Changing student1's information

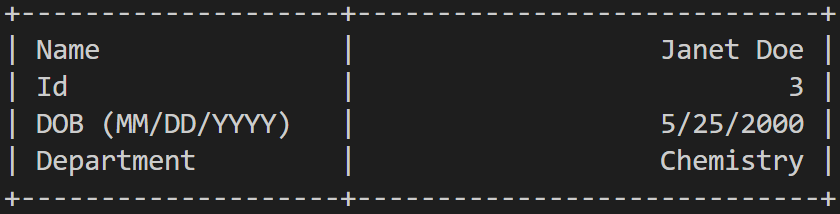


student2.name: Janet Doe

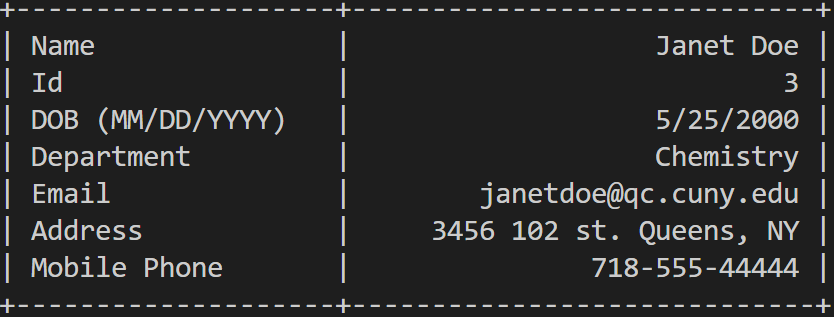
student2.id: 3

student2.dob: 2000-05-25

student2.department: Chemistry



Adding an email, address & mobile phone to student2



OK, you passed the test! student3 is null.

Exception in thread "main" java.lang.IllegalArgumentException

at Student.setId(Student.java:41)

at Test.main(Test.java:73)

You are advised to implement your own test code. When grading, we may use a different Test. Make sure that your code works under all circumstances.

Lab Work Submission:

* You can continue to work on this lab after our lab class, on your own, at home.
* Submit your lab work via Blackboard on or before: **Wednesday, October 25, 2023, 11:59pm**.
* The only accepted submission method!
* Once you submit your assignment you will not be able to resubmit it!
* Make absolutely sure the Java files you want to submit are the Java files you want graded.
* You will not be able to submit your lab work under any circumstances once **Lab15** disappears at **12:00 a.m.** on **Thursday, October 26, 2023**.
* There will be **NO** exceptions to these rules!
* To submit your lab work, upload **Department.java** & **Student.java** files (**with .java extension**) you did for this lab to the **Lab15** assignment in the **Labs** tab in your Lab section’s presence in Blackboard.
* Then, make sure you click the **Submit** button to submit your lab work.
* This lab is worth **6 points**.

|  |  |
| --- | --- |
| Department.java | 1 point |
| Student.java | 5 points |