Lab 2

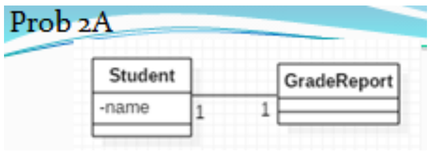
Reading: Beginning Java Objects, pp. 368—385 and the lectured Slides.

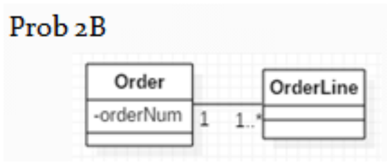
Note: You have three problem to solve.

1. Extend your work on the project management system from Problem 3 of Lab 1 so that your class diagram includes associations and dependencies, with names (optionally, roles), and multiplicities shown. Also include in your classes any operations that seem to be suggested by the associations you have added to your diagram. Below is a reproduction of the problem statement. **[5 Points]**

> ***Problem Description:***
>
> A project, before final release, is required to have a specified feature set. Associated with a project are multiple releases. A release is a functional piece of the project being developed that includes a subset of the feature set for the project and which is to be delivered on a specified date (the feature set and release date are determined by the Project Manager). When the last release is delivered, the project is considered completed.
>
> Associated with each feature for a project is a developer who is responsible for developing this feature for inclusion in the project. A developer has an id and provides, for each feature he is responsible for, the estimated time remaining to complete work on that feature. The Project Manager assigns features to developers to work on.

2. For each of the following small class diagrams, write corresponding Java code and create a main method that creates instances of the classes in the diagram in a way that agrees with the requirements of the diagram. For example, in a 1-1 two-way association between A and B, whenever an instance of A is created, an instance of B must also be created and each must contain an instance of the other. **[Each 10 Points→20 points]**





Prob2A: In Grade Report class, include an attribute grade as String.

Prob2B: In Order add an attribute orderDate.

In OrderLine class, add atributes orderlinenum,price and quatity.

Name your java packages for these as follows:

`prob2A, prob2B`

Problem: 3

Objective: Develop a one-to-one unidirectional association example.

Steps:

1. **Create an Example:** Reflect on the scenario discussed in class (refer to Slide 30) and come up with your own example illustrating a One to Zero..One unidirectional association.
2. **Diagram:** Draw a diagram representing the association, ensuring it aligns with your example.
3. **Implementation:** Implement the association in Java. You can either use the straightforward method discussed on Slide 31 or the singleton approach from Slide 32.
4. **Main Method:** Develop a main method that instantiates the classes according to the diagram, ensuring all relationships and requirements are correctly followed.