Lab Assignment: Inheritance and Polymorphism

Learning Objectives

You should be able to:

1. Read and understand UML diagrams involving class inheritance
2. Implement classes from a given specification
3. Create inheritance relationships
4. Understand the difference between *is-a* and *has-a* relationships
5. Develop testing procedures for your own code

Preparation

1. Once you create the new project, it may not initially know where to find the standard Java libraries (it varies depending on your configuration).

   (a) *Project* menu: Select *properties*

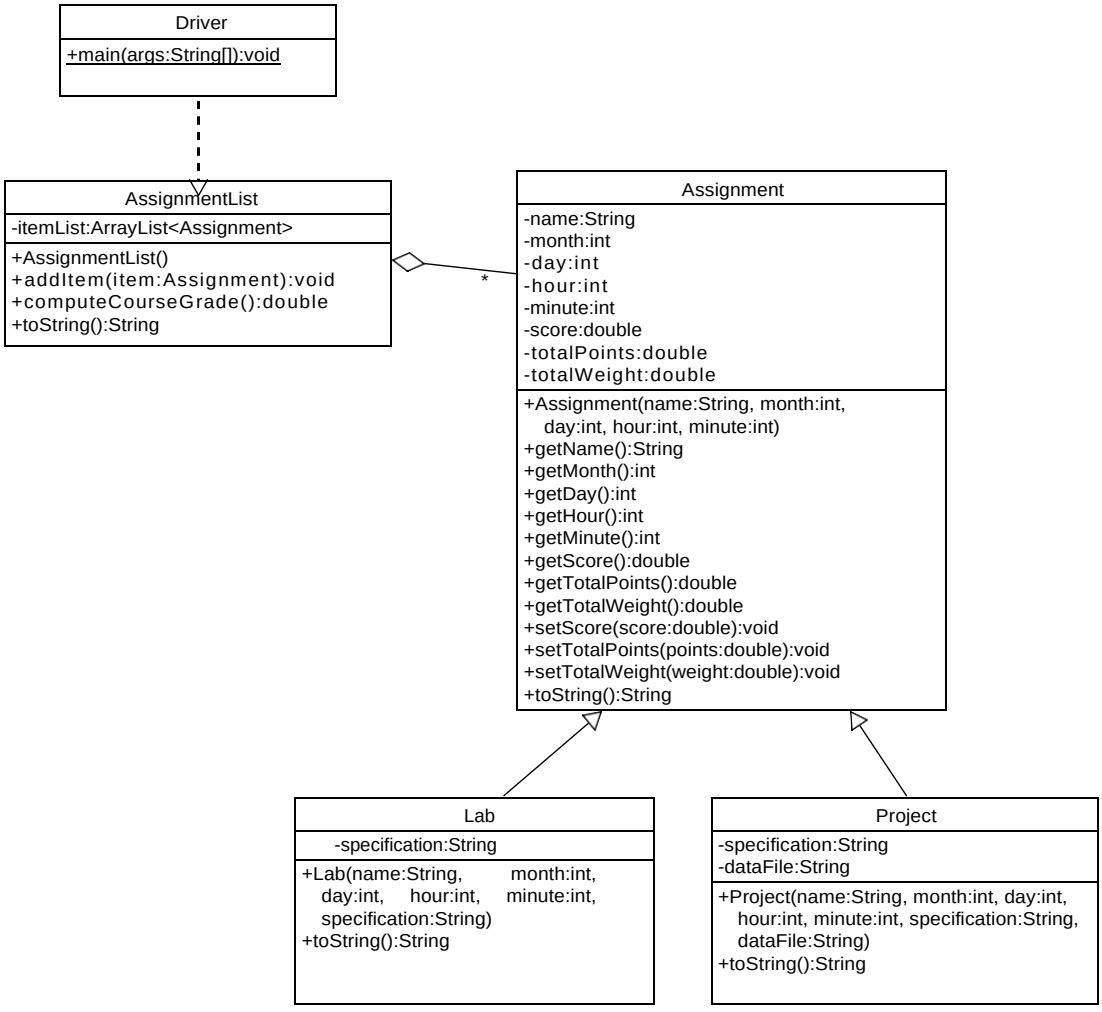
   (b) Select *Java Build Path / Libraries*

   (c) If the *JRE System Library* is not listed, then select *Add library: JRE System Library* and click *Next*. Select *Workspace default*. Click *OK*

   (d) Click *Apply* and then *OK*

Representing Assignments

Below is the UML representation of a set of classes that represent items that are used to compute a course grade. Your task is to implement this set of classes.



The line from Lab to Assignment indicates that a Lab *is-a* Assignment. You should use inheritance to capture this relationship. Likewise, for the Project and Assignment classes.

The line from AssignmentList to Assignment indicates that AssignmentList *has-a* Assignment. Observe that there is an itemList attribute (instance variable) in the middle section of the AssignmentList box. Other than including this attribute, there is nothing special to be done for a has-a relationship. The "*" on this relationship indicates that the AssignmentList can have any number of Assignment objects. This fact is captured in the AssignmentList through the use of the ArrayList of Assignment (this is the meaning of the ArrayList<Assignment> notation in the type definition).

How much a particular Assignment contributes to a course grade is determined by three properties:

1. **score** is the number of points that you have received on the assignment,
2. **totalPoints** is the number of points that are possible for the assignment, and
3. **totalWeight** is the weight of this Assignment relative to other assignments.

AssignmentList.computeCourseGrade() returns a score for the class based on the list of assignments. In particular:

$$return = \frac{\sum_{i=0}^{n-1}(totalWeight_i \times score_i / totalPoints_i)}{\sum_{i=0}^{n-1} totalWeight_i}$$

where n is the total number of Assignments in the list, and $totalWeight_i$ refers to the *totalWeight* of the i^{th} Assignment. If $n = 0$, then the returned course grade must be zero.

Specific Instructions

1. Create a Java class for each of the *Assignment*, *Lab* and *Project,* AssignmentList, and Driver classes as described in the UML diagram.
   - Be sure that the class name is exactly as shown
   - You must use the default package, meaning that the package field must be left blank when you create the class
2. Implement the designated attributes and methods for each class:

- Use the same spelling and visibility for instance variables and method names as shown in the UML
- Do not add functionality to the classes beyond what has been specified
- Don't forget to document as you go!

Submission instructions

- All required components are due on Thursday, March 21 2019