Class Diagrams

The goal of this worksheet is to develop *class diagrams*, which are visual models for classes and relationships among classes within a program. The class diagrams in the GoF book make use of OMT (Object Modeling Technique) as explained in [GoF, Appendix B.1]. The modern version of OMT is called UML (Unified Modeling Language). If you're interested, you can spend a great deal of time learning about UML; feel free to look it up on your own. In what follows we will develop a simplified version of a UML class diagram, and indicate how they compare to the OMT class diagrams.

In order to complete this worksheet you will need the files located in the folder `ClassDiagrams` in CoCalc.

1. (3pts) Consider the class defined in the file `01-Account.h`.

   (a) Draw the UML class diagram and the OMT class diagram for `Account` without any type information. Label the diagrams with UML and OMT appropriately.

   (b) Draw the UML class diagram for `Account` with all the type information.

2. (5pts) Relationships:

   (a) (Is-a) Suppose `BaseClass` has a subclass `DerivedClass`. In other words, `DerivedClass` inherits behavior from `BaseClass`. Draw the corresponding class diagram with the appropriate arrow. Note that UML and OMT use the same arrow for inheritance.

   (b) (Has-a) Draw a class diagram that shows that class `A` has a data member named `b` of type `B` by drawing a solid line with a solid arrow from `A` to `B`. Label the start of the arrow with the name of the data member.

   (c) (Has-a collection) Draw a class diagram that shows that class `A` has a collection `bs` of data members of type `B` by adding the label `0..*` to the end of the has-a arrow. This labeling is consistent with UML.

   (d) Draw the class diagram for the classes found in `02-Relationships.cpp`. Your class diagrams should include names of all methods and data members, but you do not need to include the type information. Label all the has-a arrows with the name of the associated data member (these names do not need to be included in the diagram's list of data members).

3. (5pts) Differences between our class diagrams and those found in UML and OMT:

   (a) (Collections in OMT) In place of the label `0..*`, OMT draws a filled in circle at the end of their has-a arrows. Redraw your answer to 2(c) using OMT.

   (b) (Aggregation and Composition) For the sake of simplicity, in this course (at least for now) we won't be drawing diamonds on our class diagrams. Describe the use of diamonds in OMT and UML class diagrams.

   (c) (Associations) Describe how UML indicates that there is some association between two classes. Note that a solid arrow in UML may not always indicate a has-a relationship.

4. (7pts) Abstract Classes:

   (a) What does it mean to say that a method is abstract?

   (b) What does it mean to say that a class is abstract?

   (c) What do we call a class that is not abstract?

   (d) How are abstract methods and classes indicated in OMT class diagrams?

   (e) Draw the class diagram for the classes found in `04-Abstract.cpp`. Attach the label `<<abstract>>` to each abstract class, and the label `<<abs>>` to each abstract method.

5. (10pts) Draw the class diagram for all the classes in `05-Example.h`.

6. (10pts) Draw the class diagram for all the classes in `06-Employee.cpp`.

7. (10pts) Draw the class diagram for all the classes found in the folder `07-src`.