CSC207.01 2014S, Class 16: Inheritance, Continued

Overview

- Preliminaries.
  - Admin.
  - Upcoming work.
  - Questions on the exam.
- Q and A on Inheritance and Polymorphism.
- Lab.
- Reflection - What are benefits and issues of inheritance and polymorphism?

Preliminaries

Admin

- Issues with homework partners.
- Extra credit:
  - CS Table Friday at noon: Law, Order, and Computers
  - CS Extras next week: ?
  - More?

Upcoming Work

- Reading for Monday: Java and the Command Line (Forthcoming Saturday)
- No writeup for the inheritance lab.
- Continue to work on the exam - bring questions.

Questions on the Exam

- Bring some on Monday.

Q and A on Inheritance and Polymorphism

Lab

- Start Eclipse when you come in the room. It may make things faster.

Reflection

Sam, talk to us about protection

- Sam's model: We use protection of fields and methods so that we can easily change implementation without affecting our clients.
- Four levels of protection
  - private - only other objects in the same class
  - package (default) - only other objects in the same package
    - This fits Sam's model well - We'll assume that the elements of a package work together, so that if we change fields in one place, we know what else we'll have to change
  - protected - other objects in the same package, objects in a subclass
  - public - anything

Sam, what should we have learned (in general)?

- You can use an element of a subclass in place of an element of its superclass [DecrementableCounter, initial questions]
- When you do so, you don't get to use the additional components of the subclass [DecrementableCounter, additional questions]
- However, if the subclass has overridden the methods of the superclass, you get the overridden methods. [NamedCounter]
- These characteristics also hold for sub-sub-classes (and so on)
- You cannot add to the behavior of a method [LimitedCounter]
- Changing semantics when overriding can be dangerous [DoubleCounter]
- Sometimes we want to ignore parts of the underlying implementation, and treat a superclass as an interface [DblCtr]

Sam, what annoying Java issues should we have learned?

- You need to call the constructor of the superclass.
- You need the right constructor in the superclass.
- And you have to do it first!