Inheritance and Subtyping

1-21-2011

Opening Discussion

- Minute Essay Comments
  - Passing mechanism in Scala.
  - There is no ++ in Scala. Minor sacrifice for flexibility.
  - Demonstration of Any.

Inheritance in Drawer

- In the in-class project we are going to have multiple types of things that we can draw.
- We will have a Drawable type at the top of a hierarchy. Anything that can be drawn will come below that.
- There is also a command supertype and different commands will come under it.

Abstract

- Abstract types are types you can't instantiate. You have to make subtypes and instantiate those.
- It is quite common that supertypes know something should be possible, but have no idea how to do it.
- They might also know that a value is needed, but not know what the value should be.
- These members are called abstract. In Scala simply don't initialize them. Class must be labeled abstract.

Protected

- The protected visibility allows subtypes to see the members.
- This visibility isn't used all that commonly. Only when you have methods or members that subtypes need to deal with but which really isn't important to any other code.

Anonymous Classes

- You can make a subtype of a given type that doesn't have its own name. We did so last semester.
- The syntax is like this.
  - new SuperType(args) { ... }
  - The args are optional if they aren't needed.
- If the supertype is abstract then implementations of the abstract members must appear in the {}.

Writing Code

- I want to spend the rest of the day working a bit on the drawing application.

Minute Essay

- Can you think of anything that needs to be a type hierarchy in the project you are planning to do? If so, what is it?
- Quiz next class.
- If you use a local copy of the book, remember to pull it down occasionally so you get updates.