Classes, Objects, and Encapsulation

8-26-2011
Opening Discussion

- Have you come up with any questions about the class or topics from last time?
- Programs outside the terminal?
- Web apps?
Getting the Balance

- Let's write just a bit more on our bank account example.
- We have to have enough to make sure things work.
- We also want to protect from things going wrong.
Special Methods

- Whether you are writing a class, an object, or a trait (for later), there are certain methods that are interpreted in special ways.
- Scala also has a broader naming scheme than most other languages.
- Let's make a 3-D vector class we can demonstrate things on.
Symbolic Methods

- These aren't really special, it is just the allowed names.
- “Normal” names start with a letter or underscore followed by letters, numbers, and underscores.
- You can also have names that are composed of operators symbols.
- Lastly, you can have a normal name followed by an underscore and operator symbols.
- Abusing this leads to unreadable code.
Unary Operators

- The operators -, ~, and ! can be used in a unary, prefix notation.
- If you want one of these defined for your type declare a method with unary_ followed by the symbol you want.
  - def unary_! = ...

Methods that take no arguments in Scala don't require parentheses.

- `obj.value`

Could be a method, a val, or a var.

To preserve transparency you can make assignment methods.

- `def value_=(_:Type) { … }`

This allows

- `obj.value = something`
apply and update

- If you want to be able to use an object like a function, simply define an apply method.
- To be able to do assignments to an “index” provide update.
- This is how Arrays and other collections work.
  - \( \text{arr}(5) \rightarrow \text{arr}.\text{apply}(5) \)
  - \( \text{arr}(5)=1.3 \rightarrow \text{arr}.\text{update}(5,1.3) \)
You can make multiple methods with the same name as long as they take different arguments.

Which type is called depends on the static type of the argument.

Be careful with overloading. It can lead to confusion.
Companion Objects

- An object declaration with the same name as a class that is in the same file is called a companion object. It has access to private elements of the class.

- It is common to construct objects using apply methods in the companion object. That is how you get this syntax:
  - Array(1,2,3)
  - List(7,5,3)
You are supposed to write code that models a car. What are some of the classes you might create. List a few along with methods and member data.