Object Oriented Programming

Peter C. Chapin
CIS-3030, Vermont Technical College

Subtypes

- Recall…
  - A type is a set of values and a set of operations.
- ***Defn***
  - A is a subtype of B (written `A <: B` in Scala) means
    - A's values are a subset of B's values.
    - A's operations are a superset of B's operations.
  - Cat <: Animal
    - All Cats are Animals. Cats may have extra operations.

LSP

- **Liskov Substitution Principle**
  - *A <: B means an object of type A can be used where an object of type B is expected without changing program correctness*.
    - `def feed(a: Animal) = …`
      `feed( new Cat )`
    - `val a: Animal = new Cat`
    - `def addCreature(zoo: List[Anima]) =`
      `(new Cat) :: zoo`
    - Every Cat is an Animal. Animal operations apply to Cat

Scala Notation

- Scala uses `extends` to define a subtype.
  - ```
    class Animal
    class Cat extends Animal
    class Dog extends Animal
    class Tiger extends Cat
    ```
  - Tiger <: Cat <: Animal and Dog <: Animal
  - Subtypes can:
    - Add new fields and operations (“extends”)
    - Override existing fields and operations
    ```

Abstract Classes

- Supertype used to name a concept
  - … but it makes no sense to create an instance.
  - … declare the class `abstract`
  - `abstract class Animal`
    - `new Animal` is now an error.
    - … but references to Animal are allowed… must refer to a subtype instance (Cat, Dog, Tiger, etc).

Abstract Methods

- An abstract class can have abstract methods
  - ```
    abstract class Animal {
      def vocalize
      def getWeight = …
    }
    ```
  - No implementation for `vocalize`
    - Does not make sense in the general case.
  - Can have normal methods also
    - Operations for which general implementation ok.
    ```

Concrete Classes

- Concrete classes
  - *Must* define inherited abstract methods
  - *May* override inherited concrete methods
  - ```
    class Cat extends Animal {
      def vocalize = println("Meow")
      override def getWeight = 10.0
    }
    ```
  - Here we assume all cats weight 10 pounds.
    - What about Tigers?
    ```

`super` Calls

- Sometimes you want to "add value"
  - ```
    class Cat extends Animal {
      override def getWeight = {
        val animalWeight = super.getWeight
        animalWeight + furWeight
      }
    }
    ```
  - Cat's getWeight invokes superclass method
    - … and then does some additional things.
    ```

Constructors

- Superclass constructors called automatically
  - ```
    class Animal {
      println(“Assembling protoplasm”)
    }
    class Cat extends Animal {
      println(“Meow”)
    }
    new Cat
    ```
  - Outputs “Assembling protoplasm… Meow”
    ```

Constructors with Parameters

- Provide constructor arguments up front
  - ```
    class Animal(w: Double) {
      println("Assembling protoplasm")
    }
    class Cat(w: Double) extends Animal(w) {
      println("Meow")
    }
    new Cat(10.0)
    ```
  - Outputs "Assembling protoplasm… Meow"
    ```

Flaw in Scala?

- Consider…
  - ```
    abstract class Animal {
      println(“Assembling protoplasm”)
      vocalize
      def vocalize
    }
    class Cat extends Animal {
      val loud = 10
      def vocalize = println(s“Meow $loud”)
    }
    ```
  - What’s the problem?
    ```

C++ Fix

- C++ turns off dynamic dispatch in constructors
  - ```
    class Animal {
      Animal( ) {
        printf( "Assembling protoplasm\n" );
        vocalize( );
      }
      virtual void vocalize( ) = 0;
    };
    ```
  - C++ class with constructor attempting to call a pure virtual method.
    ```

C++ Fix continued

- The corresponding Cat class
  - ```
    class Cat : public Animal {
      int loud;
      Cat( ) {
        loud = 10;
      }
      void vocalize( ) {
        printf( "Meow %d\n", loud );
      }
    };
    ```
  - The program fails!
    ```

Compiler Reactions

- **clang++** (v3.1)
  - *Compile Time*: warning: call to pure virtual member function 'vocalize'; overrides of 'vocalize‘ in subclasses are not available in the constructor of 'Animal‘
  - *Run Time*: pure virtual method called (program aborted)
- **g++** (v4.5.2)
  - *Compile Time*: warning: abstract virtual ‘virtual void Animal::vocalize()’ called from constructor
  - *Link Time*: undefined reference to `Animal::vocalize()‘
- **MSVC++** (v11.0)
  - *Link Time*: unresolved external symbol "public: virtual void Animal::vocalize(void)"