Introduction to Object Oriented Programming
Timothy A. Budd

Chapter 20

A Second Look at Classes

© Timothy A. Budd 1996 (Modifications © H. C. Cunningham 1997)

Topics in this Chapter

- What is a class?
- Class data values
- Object-oriented programming without classes

© Timothy A. Budd 1996 (Modifications © H. C. Cunningham 1997)

What is a Class?

- A "cookie cutter" used for making clones of some object
- A generalization of a record
- A mechanism for variable scoping and encapsulation
- A special kind of type
- A special kind of object

© Timothy A. Budd 1996 (Modifications © H. C. Cunningham 1997)

Classes as Types

What is a type?

application programmer's view: set of values with common attributes and operations

object-oriented view: behavior specifications

compiler view: set of syntactic constraints on expressions (uses)

verification view: set of invariants that instances of the type must satisfy

systems programming view: protective mechanism wrapped around the bits

implementation view: description of how the bits are stored in memory

© Timothy A. Budd 1996 (Modifications © H. C. Cunningham 1997)

Implications of Classes for Types

The *is-a* relation produces some subtle issues

- Programmer analyzing a program may not know the final type of objects (which may be a subclass)
- Inheritance only ensures that *interface* is maintained, not other properties of execution
- Subclasses not guaranteed preserve all aspects of parent (may take longer, perform actions in different sequence, etc).
- Testing of program, even proof of correctness, may be invalidated by replacement of methods

© Timothy A. Budd 1996 (Modifications © H. C. Cunningham 1997)

Classes as Objects

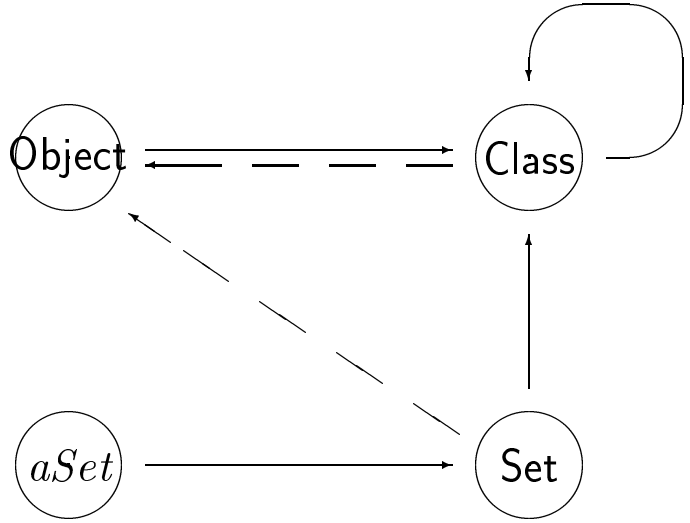
If a class is an object it must have responsibilities.

What are these responsibilities?

- Maintain information about the class (instance size, methods)
- Generate new copies of the class
- But, if a class is an object and every object is an instance of some class, what is the class of a class ?

© Timothy A. Budd 1996 (Modifications © H. C. Cunningham 1997)

Instance and Subclass Relationships



© Timothy A. Budd 1996 (Modifications © H. C. Cunningham 1997)

Classes as Objects in Java

- Root class of the inheritance hierarchy is class Object
- Parent class of all classes is Class
- One instance of class Class for each class loaded onto the machine
- Method Object.getClass returns the class for the object – instance of type Class
- If X is a class, them X.class is constant that refers to its Class object

© Timothy A. Budd 1996 (Modifications © H. C. Cunningham 1997)

Classes as Objects in Java (Continued)

- Class Class has several useful methods:
  - getName returns the name of the class as a string
  - forName loads the class with the fully qualified (including package) class name string and returns the class
  - newInstance creates a new instance of the class (applies the no-argument constructor)
  - getSuperclass returns the superclass of the class
  - other methods to return the interfaces, constructors, methods, fields, etc.
- Reflection API (java.lang.reflect package) provides additional capabilities
- Java Beans API (java.beans package) gives additional support for reusable components

© Timothy A. Budd 1996 (Modifications © H. C. Cunningham 1997)

Class Data

Often it is useful to have a single data area *shared* by all instances of a class.

- Known as a class variable
  - `static` variable in Java
- Important problem: who initializes this value?
  - avoid failing to initialize
  - avoid multiple initialization
- Mechanism differs among languages
  - Java uses standard variable initializers for simple initialization
  - Java also uses `static` blocks for more complex initialization

© Timothy A. Budd 1996 (Modifications © H. C. Cunningham 1997)

Are Classes Necessary?

Can an object oriented language be constructed without classes?

YES, Instead of classes, use *delegation*

© Timothy A. Budd 1996 (Modifications © H. C. Cunningham 1997)

Delegation

Basic structuring mechanism: construct a new object using an existing object as a delegate

Intuition: new object is exactly like old except for things that are overridden

Mechanism: any data references or function references go through original value. If not overridden, they are passed on the original delegate

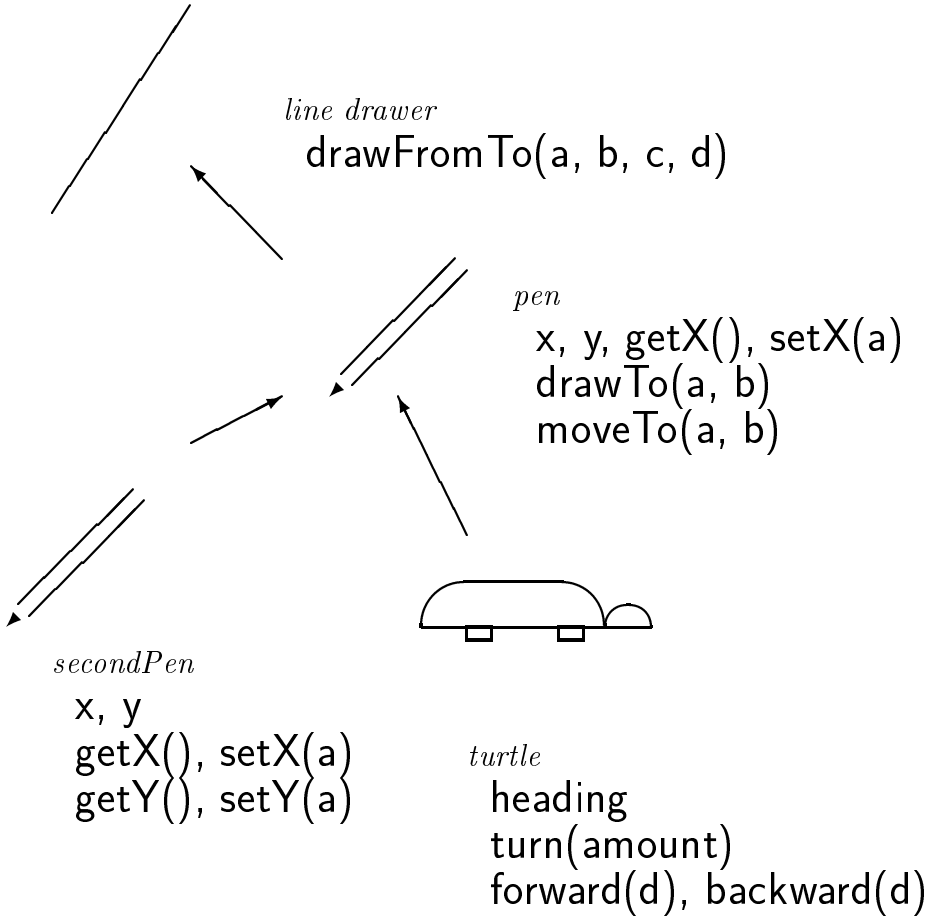
© Timothy A. Budd 1996 (Modifications © H. C. Cunningham 1997)

Example – Lines and Pens

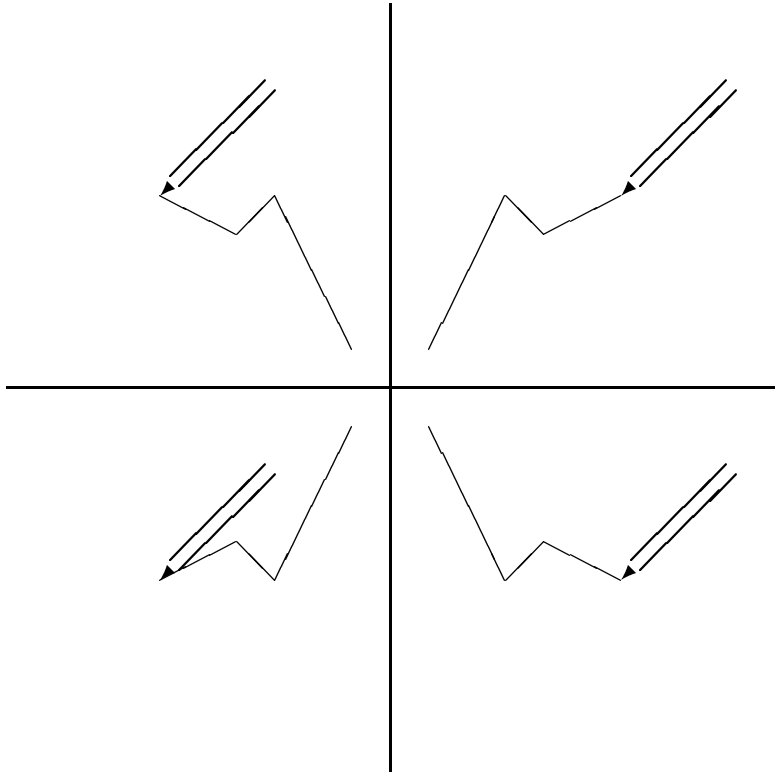
- Begin with an object that can draw a line
- From it build an object that remembers it's location and can be moved on the screen – a Pen
- From this build a different pen that remembers different locations, but with same drawing code
- Alternatively, create an object that remembers its location and its orientation – a Turtle
- Alternatively, create an object that draws four copies of any drawing command in reflections

© Timothy A. Budd 1996 (Modifications © H. C. Cunningham 1997)

Patterns of Delegation



Kaleidoscope Pen



© Timothy A. Budd 1996 (Modifications © H. C. Cunningham 1997)