Inheritance In Practice

COMP 401, Fall 2018

Lecture 10

Inheritance Recap

- Subinterfacing
  - Adding methods to create a new "contract"
- Subclassing
  - Inherits fields and methods from parent class
    - Visibility controlled by access modifier
  - Adds subclass specific fields and methods
  - Constructor relationship
  - Overloading parent methods
    - Version of a method provided in subclass with different signatures than the version(s) in the parent class.
  - Overriding parent methods
    - Subclass methods with same signature as in parent class.
    - Always virtual
      - super keyword provides a mechanism to call parent class version of a method

Using inheritance

- Example starts with several related classes that are implemented independently.
  - BlackBear
  - PolarBear
  - Python
  - Robin

  - lec10.v1

Common behavior

- Refactor common behaviors into a common parent interface.
  - Animal interface
  - lec10.v2

Common object state

- Refactor common object fields into a common parent class
  - AnimalImpl
    - Pull common fields *id* and *location* to here
    - Need to make protected in order to allow subclass access.

  - lec10.v3

Common implementation

- Refactor common method implementations into parent class.
  - Use overriding if subclasses need to do something special or different.
  - AnimalImpl
    - Constructor in parent class used by subclass constructor to initialize common fields at this level.
    - getID() and getLocation() moved here
    - Common portion of move() put here
      - Subclass-specific override of move
      - Calls common portion through *super* keyword
  - lec10.v4

Common behavior with uncommon implementation

- Notice speak() is a common behavior.
  - So we want it to be part of Animal interface
    - That way if we can have a reference to an Animal object and ask it to speak() without having to know what subclass it is.
- But no common implementation.
  - Each subclass of animal will have a different way of speaking.
  - No good default or commonalities in way of speaking that can be specified at parent class.

Abstract Classes and Methods

- Parent class has no meaningful implementation of a method.
  - But method is part of an interface implemented by parent class
  - Expect subclass to provide it.
  - In these situations, we never expect (or want) the parent class to be instantiated directly.
    - We always make new objects using a subclass.
- Syntax
  - Use “abstract” modifier when declaring parent class
  - Declare any methods that must be provided by subclass in parent
    - Add “abstract” modifier to method signature.
    - Follow signature with semicolon instead of method definition

Example Revisited

- AnimalImpl
  - Declare implementation of Animal here where it belongs since AnimalImpl is matching implementation.
  - Declare AnimalImpl as abstract
    - Prevents direct instantiation
  - Declare speak() method as abstract
    - No common implementation, but needs to be declared here as part of common interface.
    - Declaring as abstract forces subclass to override.
- lec10.v5

Repeat Exercise with Bears

- lec10.v6
  - Draws out common bear behavior into Bear interface and common bear implementation in BearImpl
    - Bear extends Animal
      - Adds getColor()
    - BearImpl extends AnimalImpl
      - Provides lumber(), trek(), move(), and speak().
      - Declared abstract because subclass must provide getColor()