Introduction to Computer Science II

CS112-2012S-10

Inheritance

David Galles

Department of Computer Science
University of San Francisco
10-0: Classes

- Want to create a bunch of classes for a real estate program
  - Apartment Building:
    - Square Footage
    - Number of floors
    - Number of units
10-1: Classes

- Want to create a bunch of classes for a real estate program
  - Private House
    - Square Footage
    - Number of floors
    - Number of bedrooms
10-2: Classes

- Want to create a bunch of classes for a real estate program
  - Office Building
    - Square Footage
    - Number of floors
    - Number of Elevators
Several data members (and methods) that are shared across all building types:

- Square footage
- floors

We could duplicate these variables in all of our classes (so that the House class and Office class and Apartment class would all contain a definition of an instance variable for Square Footage, for example)

- What are potential problems with this approach?
It would be nice if we didn’t need have so much repetition:

- Class that describes a building
  - Contains Square footage, number of floors, etc
  - (Features common to all buildings)
- Class that describes an apartment building
  - Automatically has everything in a generic building
  - Extra instance variables / methods that are particular to apartments
Add "extends <classname>" to class definition
  - class Apartment extends Building { ... }

Defines an "is-a" relationship
  - Apartment is a building

Defines a superclass/subclass relationship
  - Building is the superclass
  - Apartment is the subclass
• Add “extends <classname>” to class definition
  • class Apartment extends Building { ... }
• Subclass inherits all of the methods / data from the superclass.
  • Examples from code
10-7: Adding Constructors

- Superclasses and subclasses can have constructors
- Subclass can call the constructor of the superclass with the “super” keyword
  - We’ll find more uses for the super keyword later
- If you call the constructor of a subclass from a superclass, you need to do it first
- Show examples
  - How would you do the same thing without using “super”?
If you do not call the constructor of the superclass explicitly, then the default (no parameter) version of the constructor is called automatically at the beginning of the constructor.

Each class takes care of itself:
- Don’t need to worry (too much) about the inner workings of a superclass when writing a subclass.
- Just concentrate on the new stuff.

(examples)
10-9: Access control

- public: Anyone use it
- protected: Only subclasses can use it
- private: No one but the original class can use it.

Examples
class SuperClass {
    void print() {
        System.out.println("Message from SuperClass");
    }
}

class SubClass extends SuperClass {
    void print() {
        System.out.println("Message from SubClass");
    }
}

SuperClass sup = new SuperClass();
SubClass sub = new SubClass();

sup.print();
sub.print();
class SuperClass {
    void print() {
        System.out.println("Message from SuperClass");
    }
}

class SubClass extends SuperClass {
    void print() {
        System.out.println("Message from SubClass");
        super.print();
    }
}
Define a Class Vehicle, which has the protected integer instance variable `numWheels`. Also, add public accessor (get/set) methods for `numWheels`, and a constructor that takes as an input parameter the number of wheels.

Define a Class Bicycle, subclass of Vehicle, which has the protected integer instance variable `gears`. Add public accessor (get/set) methods for `gears`, and a constructor that takes as an input parameter the number of gears. The constructor should set the number of wheels to 2 (by calling super).

Define a Class Car, subclass of Vehicle, which has the protected double instance variable `horsepower`. Add public accessor (get/set) methods for `horsepower`, and a constructor which takes as an input parameter the horsepower, and set the number of wheels to 4 (by calling super).

Define a Class Truck, subclass of Car, which has the protected double instance variable `payloadVolume`. Add public accessor for `payloadVolume`, and a constructor that takes 2 parameters, horsepower and payload volume.

Create a main program that instantiates one of each class, and then uses set methods to make the truck a 3 wheeled, 0.5 horsepower truck with a payload volume of 0.1.