The Learning House

427 S 4th Street #300

Louisville KY 40202

Java Object-Oriented Concepts

Lesson 8 - Specialization and Inheritance

SOFTWARE GUILD

Objectives

- Understand the benefits of inheritance
- Reuse code through inheritance
- Understand the protected keyword
- Use polymorphism in classes
- Understand the difference between derived and base types
- Use polymorphic methods
- Create abstract base classes

Overview Example

- Employee
  - Can doWork(), createObjectives(), etc.
- Manager
  - Can do same things as Employee plus:
    - hire(), fire(), givePerformanceReview()
    - Implementation of createObjectives() might be different
- SummerIntern
  - Can do same things as Employee plus:
    - requestPerformanceReview()

Specialization Through Inheritance

- A dog is a special kind of mammal
- A rose is a special kind of plant
- What do we mean when we say a dog **is-a** mammal?
- The specialization relationship is implemented in Java via **inheritance**
- We use the **extends** keyword

Terminology

- Base class is sometimes called superclass
- Derived class is sometimes called subclass
- The derived class extends the base class
- Employee is the base class, Manager and SummerIntern are derived classes

Code Reuse with Inheritance

- Let’s write Employee, Manager, and SummerIntern
- We’ll add new methods to derived classes and reuse methods on base class

Overriding Methods

- We've talked about **overloading**
  - Same method name, different signature
- We can also **override**
  - This means *replacing* the base class implementation of a method with our own
- Let's do this with Manager.createObjectives()

Protected Access

- So far we've seen **public** and **private** access, now we'll look at **protected**
  - o Public access: accessible to all, both inside and outside of the class
  - o Private Access: accessible only within the class. Derived classes do not have access.
  - o Protected Access: accessible only within the class and any derived classes.

Constructors

- You can call the base constructor from the derived constructor according to these rules:
  - You invoke the base class constructor using **super**.
  - You can only call **super** from within the constructor of the derived class, nowhere else.
  - The call to **super** must be the first statement in the constructor.
  - The call to **super** must match the signature of a valid constructor in the base class.
  - If you do not explicitly call **super** in the constructor of a derived class, the compiler will automatically call the base class default constructor. If one doesn't exists, a compile error will occur.
  - If your derived class does not define a constructor, the compiler provides the derived class with a default constructor that does nothing but call **super**, invoking the default constructor of the base class

Constructor Example

- ConstructorFun:
  - Dog (base class)
  - Retriever (derived from Dog)

Polymorphism

- Means “many-formed”
- Key idea: an object can take many forms
- Pillar of object-oriented design
- Method overloading is also known as **functional polymorphism**
- Now we add the concept of **object polymorphism**

Object Polymorphism

- We can treat specialized objects as if they were instances of a more general type
- Derived types *are* base types
- Inheritance creates an **is-a** relationship

Example

- Back to the Employee hierarchy
- Manager and SummerIntern are Employees
- We can treat them as employees
- We can call methods polymorphically

Calling Methods Polymorphically

- Let’s look at Employee and Manager
- Manager **is-an** Employee but has overridden the createObjectives method
- Can refer to a Manager object with an Employee reference
- Which version of createObjectives will be called?

Abstract Base Classes

- Represent an abstraction that will never have an implementation
- Can be used to define an object (methods and properties) and then force derived classes to provide implementations for one or more of the methods

Another Example

- Let’s implement the Employee hierarchy with Employee as an abstract base class