Chapter 9: Inheritance

Programming with Alice and Java
First Edition

by
John Lewis
and
Peter DePasquale

Objectives

- Derive new classes from existing ones.
- Explore the design of class hierarchies.
- Learn the concept and purpose of method overriding.
- Use abstract classes to enhance program design.
- Explore the protected visibility modifier.
- Examine polymorphism and its benefits.
- Explore processing threads and their creation.

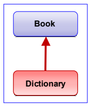
Parent and Child Classes

- *Inheritance* is the process of deriving a new class from an existing one.
- It is one of the main characteristics of object-oriented programming.
- The derived class automatically contains the variables and methods of the original class.
- One purpose of inheritance is to reuse existing software.
- The original class that is used to derive a new one is called the *parent class*, *superclass*, or *base class*.
- The derived class is called a *child class*, or *subclass*.

Creating Subclasses

- The process of inheritance should establish an *is-a relationship* between the parent and child classes.
- Example:

```
public class Dictionary extends Book
{
      // contents of Dictionary
}
```



- extends clause causes the Dictionary class to automatically inherit the definitions of methods and variables declared in the Book class.
- Although the Book class is needed to create the definition of Dictionary, a Book object is not needed in order to create a Dictionary object.

The protected Modifier

- Visibility modifiers are used to control access to the members of a class, and it is also important in the process of inheritance.
- Any public method or variable in the parent class can be explicitly referenced by name in the child class.
- Private methods and variables of the parent class cannot be referenced in the child class.
- A third visibility modifier: protected.
- Protected visibility allows the class to retain some encapsulation properties.

The super Reference

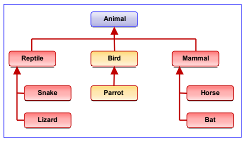
- Constructors are not inherited.
- super reference is used to invoke a parent’s constructor.
- Example: Constructor of Dictionary contained the following call: super( );
  This call explicitly calls the Book constructor.
- The child’s constructor is responsible for calling its parent’s constructor.
- If the constructor accepts parameters, they can be passed in the super call.
- The super reference also can be used to reference other variables and methods defined in the parent’s class.

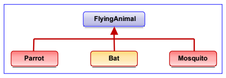
Method Overriding

- A child class can override (redefine) the parent's definition of an inherited method.
- A method can be defined with a final modifier.
- A child class cannot override the final method.
- Method overriding is a key element in object-oriented design.
- Two objects related by inheritance can use the same naming conventions for methods that accomplish the same general task in different ways.

Class Hierarchies

- The child of one class can be the parent of one or more other classes, creating a *class hierarchy*.
- Multiple classes can be derived from a single parent.
- Two children of the same class are called *siblings*, but siblings are not related by inheritance.
- Common features should be located as high in a class hierarchy as is reasonably possible.
- The inheritance mechanism is transitive.
- There is no single best hierarchy for all situations.





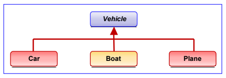
The Object Class

- All classes in Java are derived, directly or indirectly from the Object class.
- All public methods of Object are inherited by every Java class.
- The Object class is defined in the java.lang package of the Java standard class library.
- Some methods of the Object class:

```
boolean equals (Object obj)
    Returns true if this object is an alias of the specified object.
String toString ()
    Returns a string representation of this object.
Object clone ()
    Creates and returns a copy of this object.
```

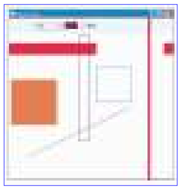
Abstract Classes

- An abstract class represents a generic concept in the class hierarchy.
- An abstract class cannot be instantiated.
- Other classes can build their definitions based on the abstract class concept.
- An abstract class usually contains abstract methods, which have no definitions.
- A class is declared as abstract by including the abstract modifier in the class header.
- The Vehicle class may be implemented as an abstract class.
- A class derived from an abstract parent must override all of its parent's abstract methods, or the derived class will also be considered abstract.

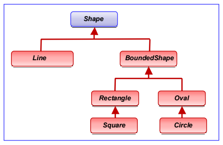


An Example: ShapeMaker

- Program allows the user to draw various shapes, filled or unfilled, using the mouse

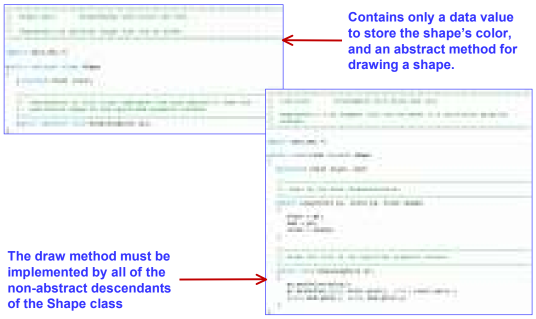


A screen shot of the program



A hierarchy of shape classes

Abstract class *Shape* and Non Abstract class Line



```
[illegible]
```

Contains only a data value to store the shape's color, and an abstract method for drawing a shape.

The draw method must be implemented by all of the non-abstract descendants of the Shape class

Polymorphism

- *Polymorphism* can be defined as “having many forms”.
- A *polymorphic reference* is a reference variable that can refer to different types of objects at different points of time.
- obj.DoIt( );
  If the reference obj is polymorphic, it can refer to different types of objects, so that line of code can call a different version of the DoIt method each time it is invoked.
- The binding of a method invocation to its definition is performed at run time for a polymorphic reference.
- It is called *late binding* or *dynamic binding*.

Polymorphism via Inheritance

- A reference variable can refer to any object created from any class related to it by inheritance.
- The type of object, not the type of the reference, determines which version of a method is invoked.
- Example:

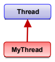
```
Mammal pet;
Horse secretariat = new Horse( );
pet = secretariat;
```

- If the Mammal class were derived from a class called Animal, then the following is valid:

```
Animal creature = new Horse( );
creature.move( );
```

Threads

- In Java, concurrency is accomplished using multiple execution *threads*.
- Class can be defined so that it runs its own thread.
- Multiple threads of execution can be running at the same time.
- A thread can be created using inheritance. Thread class is part of the java.lang package.
- If the Thread class is a parent of a new class, the child class is a thread.



Summary

- Inheritance is the process of deriving a new class from an existing one.
- Inheritance creates an “is-a” relationship between the parent and child classes.
- Protected visibility provides the best possible encapsulation that permits inheritance.
- A parent’s constructor can be invoked using the super reference.
- A child class can override (redefine) the parent’s definition of an inherited method.
- A child class can be a parent of other classes, creating a class hierarchy.
- All Java classes are derived, directly or indirectly, from the Object class.
- An abstract class cannot be instantiated. It represents a concept on which other classes can build their definitions.
- A polymorphic reference can refer to different types of objects over time.
- The binding of a method invocation to its definition is performed at run time for a polymorphic reference.
- A reference variable can refer to any object created from any class related to it by inheritance.
- The type of the object, not the type of reference, determines which version of a method is invoked.
- A thread can be created using inheritance.