11

Reusing Code using Inheritance

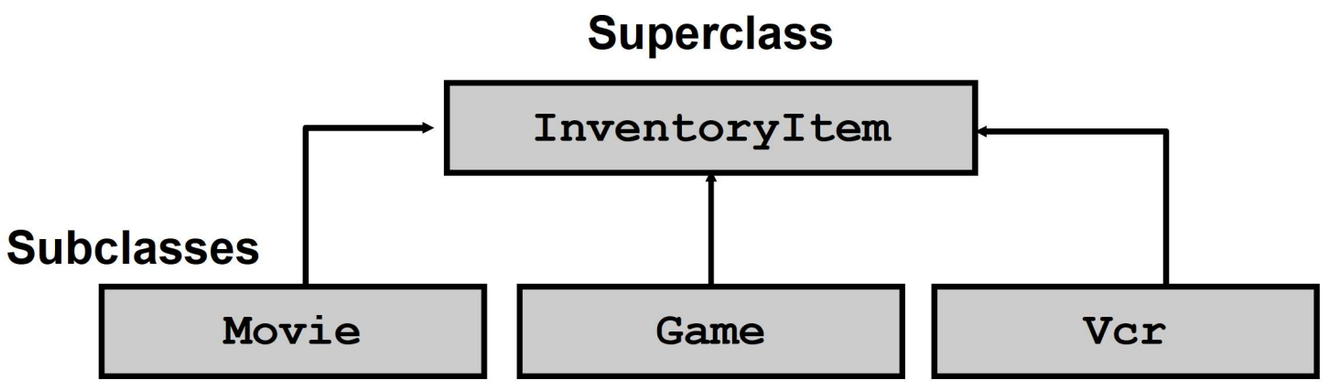
Objectives

After completing this lesson, you should be able to do the following:

- Define inheritance
- Use inheritance to define new classes
- Provide suitable constructors
- Override methods in the superclass
- Describe polymorphism
- Use polymorphism effectively

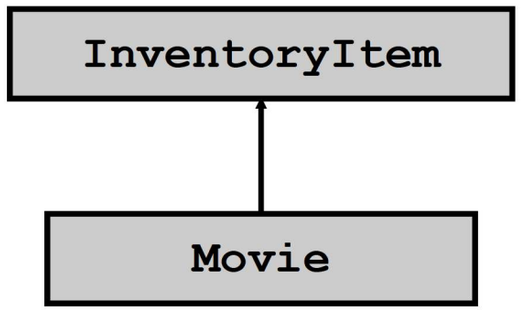
Key Object-Oriented Components

- Inheritance
- Constructors referenced by subclass
- Polymorphism
- Inheritance as a fundamental in object-oriented programming



Example of Inheritance

- The `InventoryItem` class defines methods and variables.



- `Movie` extends `InventoryItem` and can:
  - Add new variables
  - Add new methods
  - Override methods in the `InventoryItem` class

Specifying Inheritance in Java

- Inheritance is achieved by specifying which superclass the subclass extends.

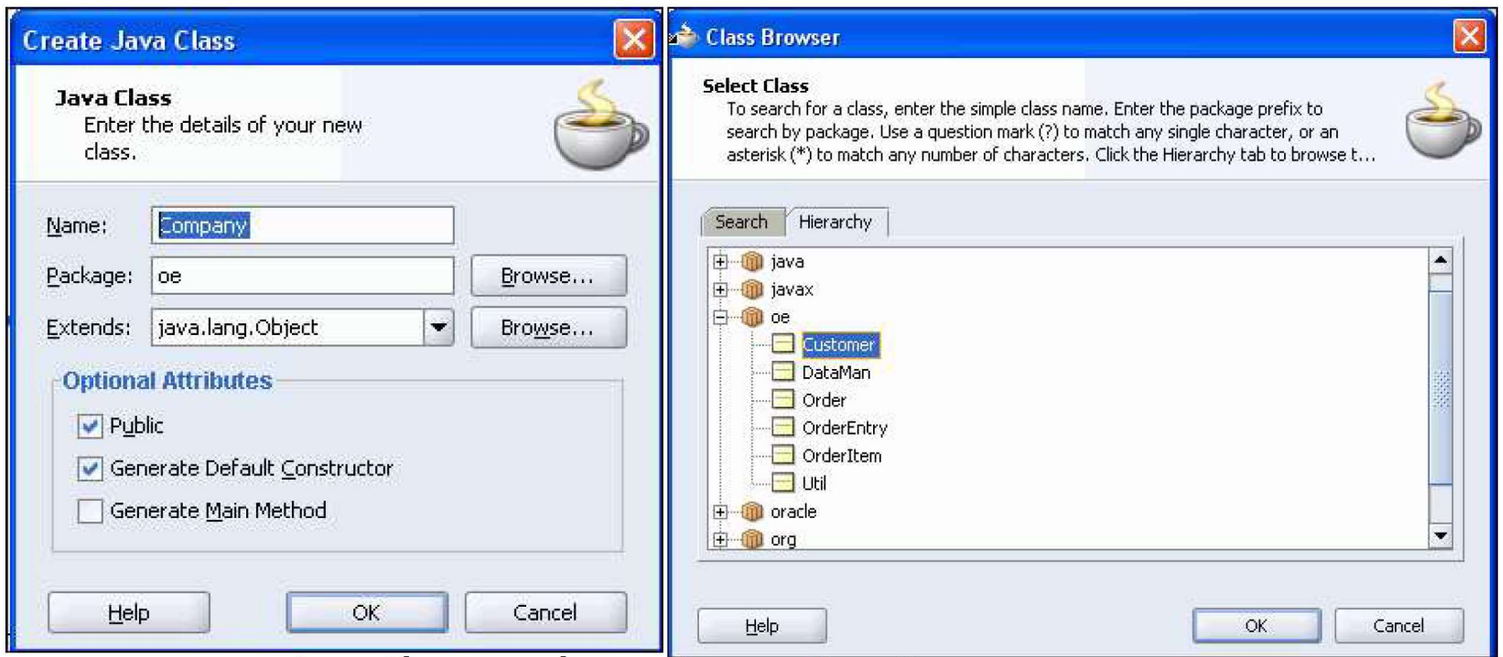
```
public class InventoryItem {
   …
}
```

```
public class Movie extends InventoryItem {
   …
}
```

- `Movie` inherits all the variables and methods of `InventoryItem`.
- If the `extends` keyword is missing, `java.lang.Object` is the implicit superclass.

Defining Inheritance

- When creating a new class, JDeveloper asks for its superclass:



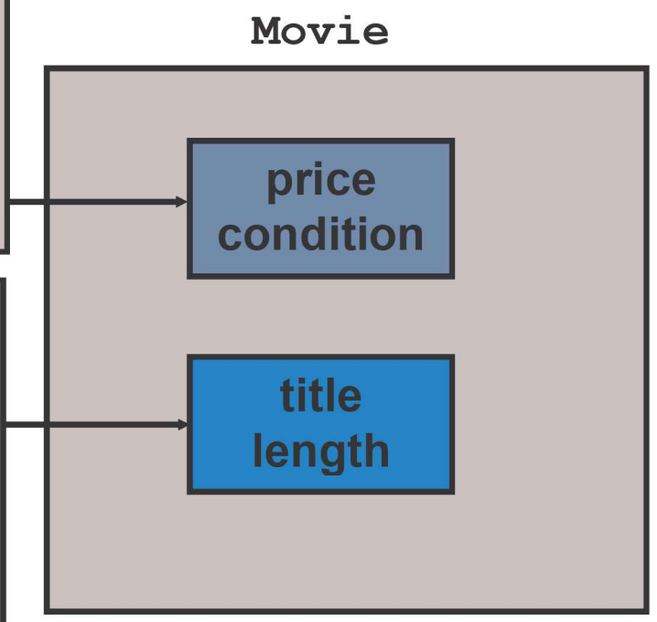
- JDeveloper generates the code automatically.

Subclass and Superclass Variables

A subclass inherits all the instance variables of its superclass.

```
public class InventoryItem {
  private float price;
  private String condition; …
}
```

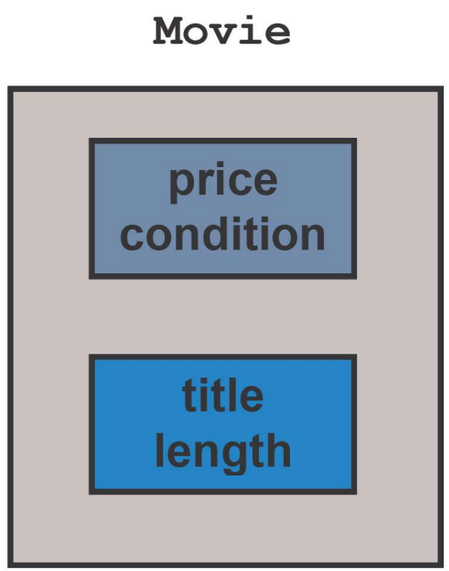
```
public class
 Movie extends InventoryItem {
  private String title;
  private int length; …
}
```



Default Initialization

- What happens when a subclass object is created?

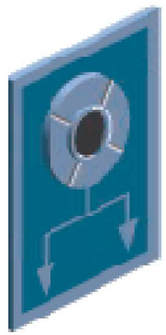
```
Movie movie1 = new Movie();
```



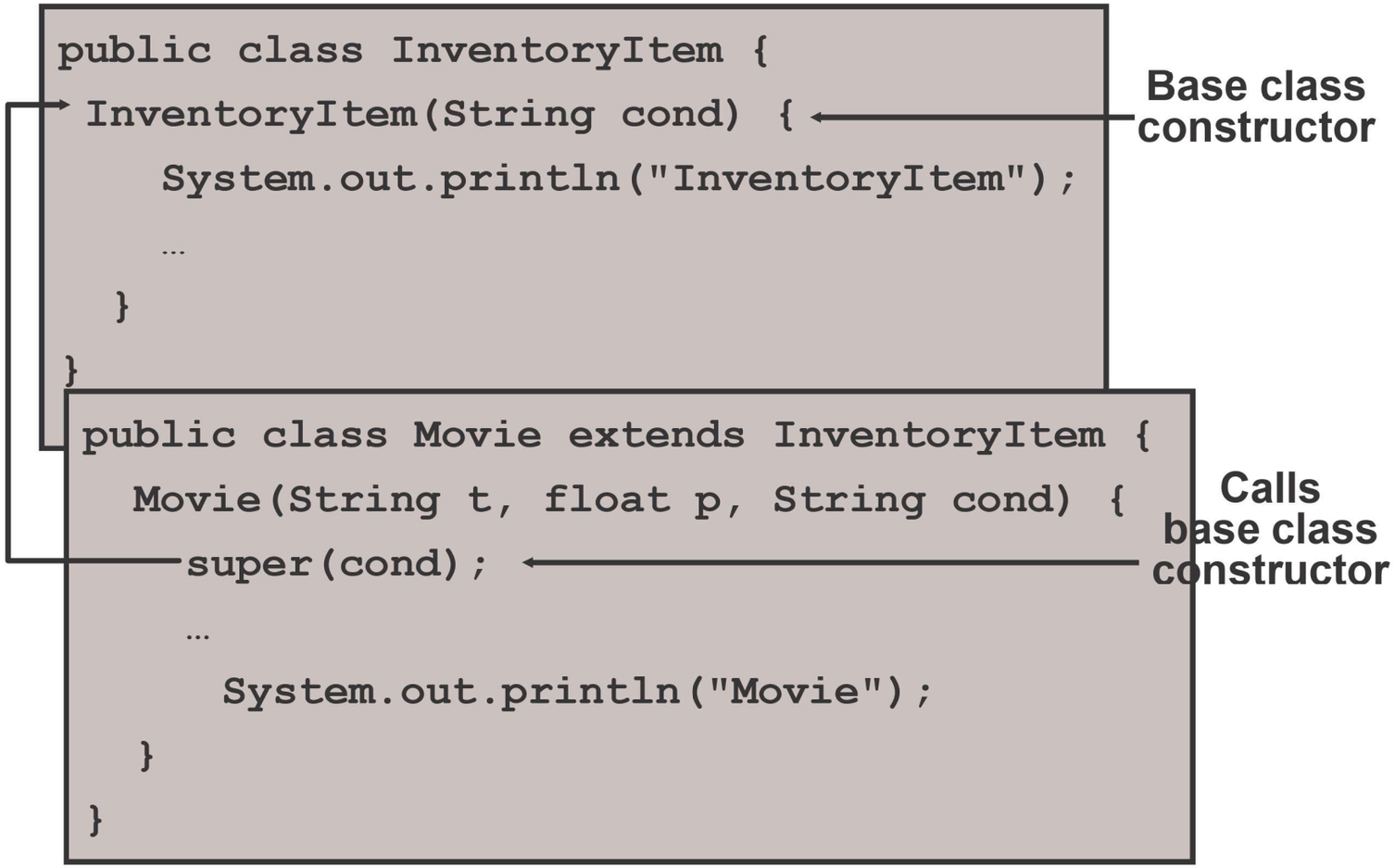
- If no constructors are defined:
  1. The default `no-arg` constructor is called in the superclass.
  2. The default `no-arg` constructor is then called in the subclass.

Super() Reference

- Refers to the base superclass
- Is useful for calling base class constructors
- Must be the first line in the derived class constructor

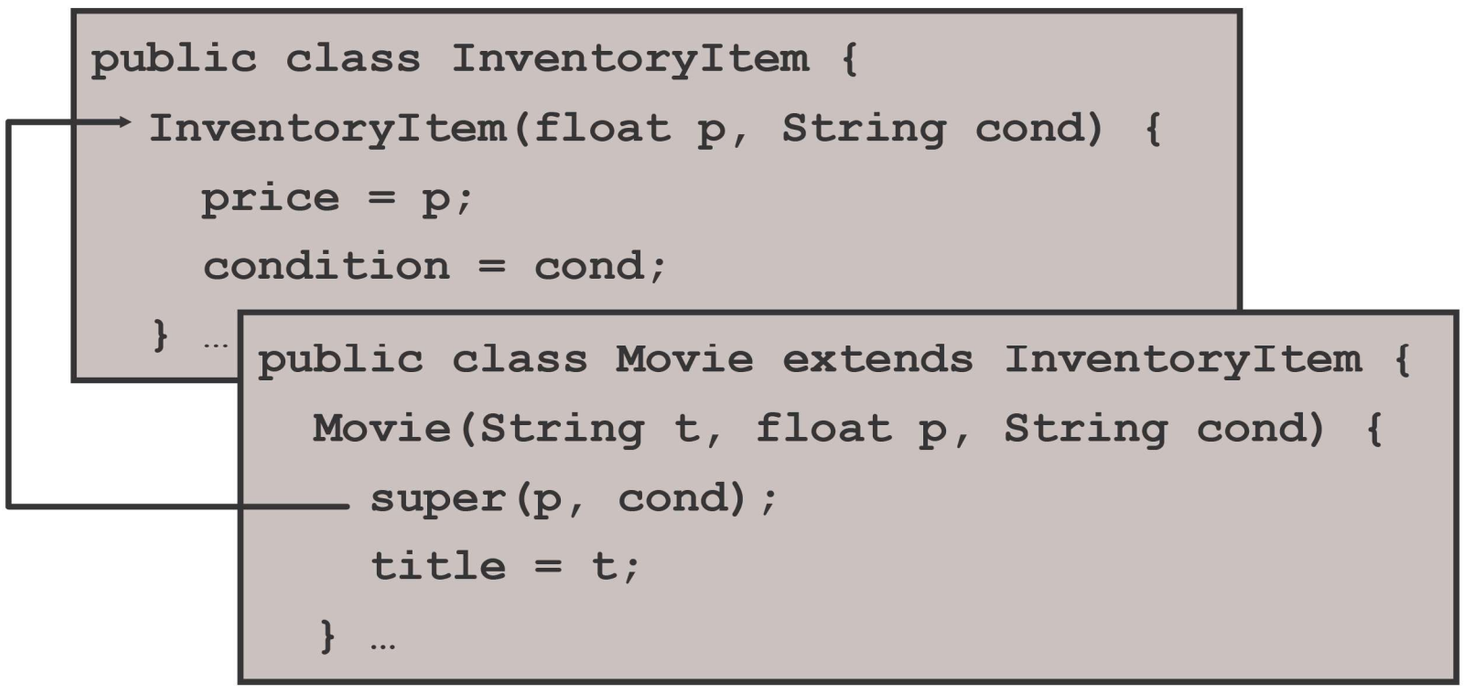


Super() Reference: Example



Using Superclass Constructors

Use `super()` to call a superclass constructor:



Specifying Additional Methods

- The superclass defines methods for all types of `InventoryItem`.
- The subclass can specify additional methods that are specific to `Movie`.

```
public class InventoryItem {
  public float calcDeposit()…
  public String calcDateDue()…
  …
```

```
public class Movie extends InventoryItem {
  public void getTitle()…
  public String getLength()…
```

Overriding Superclass Methods

- A subclass inherits all the methods of its superclass.
- The subclass can override a method with its own specialized version.
  - The subclass method must have the same signature and semantics as the superclass method.

```
public class InventoryItem {
  public float calcDeposit(int custId) {
  if …
    retur
  }
```

```
public class Vcr extends InventoryItem {
  public float calcDeposit(int custId) {
   if …
     return itemDeposit;
  }
```

Invoking Superclass Methods

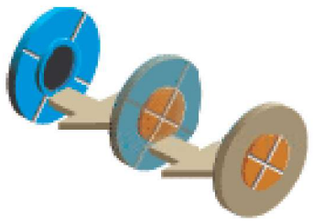
- If a subclass overrides a method, it can still call the original superclass method.
- Use `super.method()` to call a superclass method from the subclass.

```
public class InventoryItem {
  public float calcDeposit(int custId) {
    if
    ret
  }
```

```
public class Vcr extends InventoryItem {
 public float calcDeposit(int custId) {
     itemDeposit = super.calcDeposit(custId);
     return (itemDeposit + vcrDeposit);
}
```

Example of Polymorphism in Java

Recall that the `java.lang.Object` class is the root class for all Java classes.



- Methods in the `Object` class are inherited by its subclasses.
- The `toString()` method is most commonly overridden to achieve polymorphic behavior.
- Example:

```
public class InventoryItem {
 public String toString() {
   return "InventoryItem value";
 }
```

```
InventoryItem item = new InventoryItem();
System.out.println(item); // toString() called
```

Treating a Subclass As Its Superclass

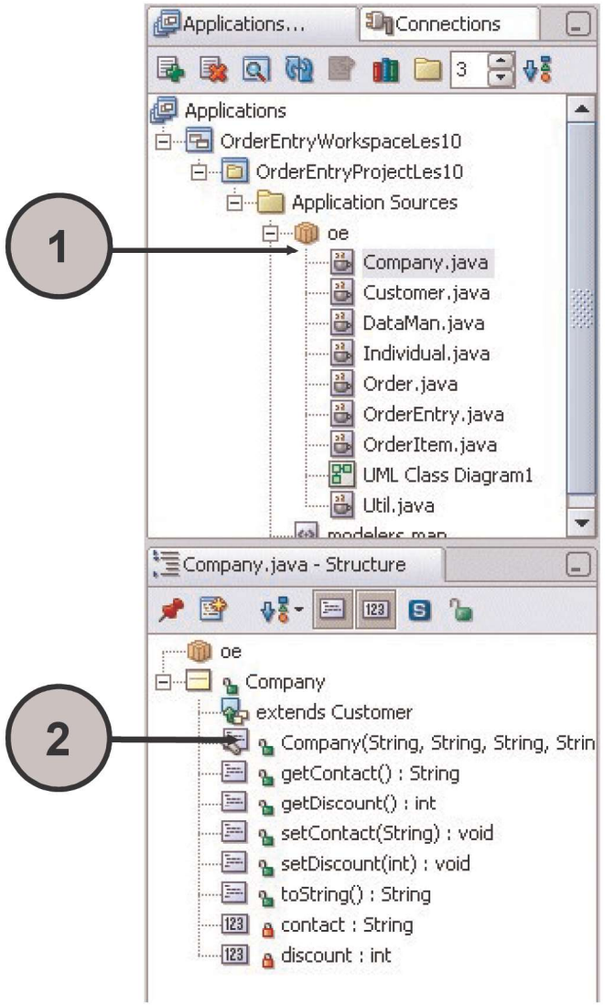
A Java object instance of a subclass is assignable to its superclass definition.

- You can assign a subclass object to a reference that is declared with the superclass.

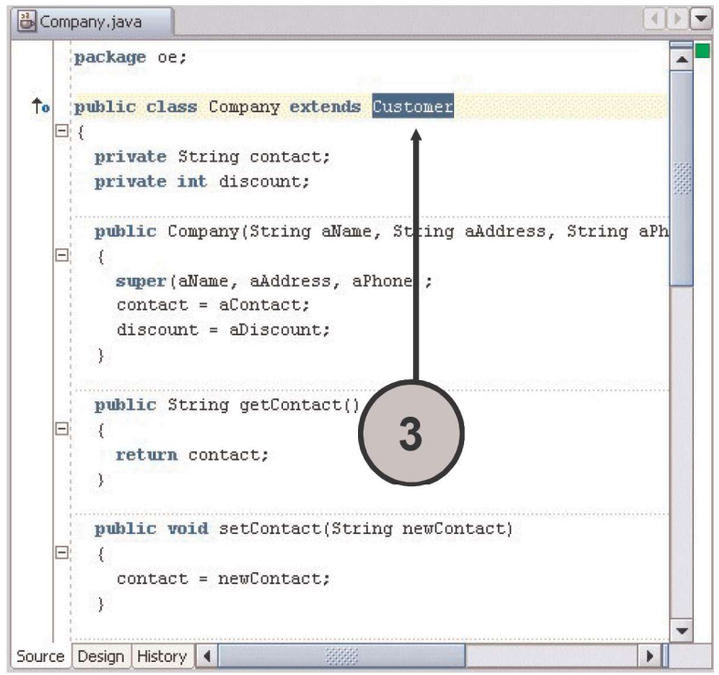
```
public static void main(String[] args) {
  InventoryItem item = new Vcr();
  double deposit = item.calcDeposit();
}
```

- The compiler treats the object via its reference (that is, in terms of its superclass definition).
- The JVM run-time environment creates a subclass object, executing subclass methods, if overridden.

Browsing Superclass References

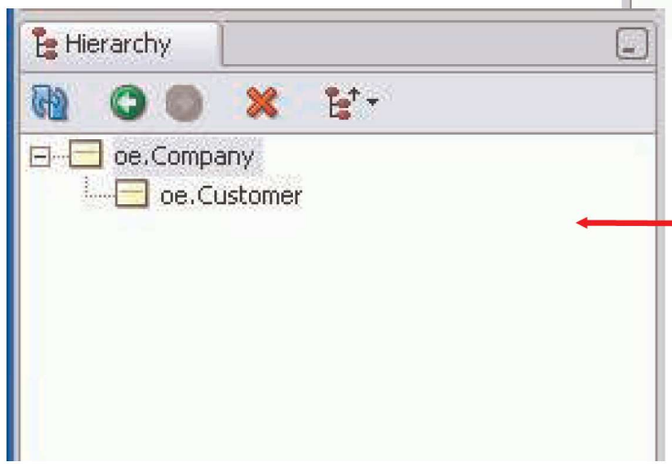


Oracle JDeveloper makes it easy to browse the contents of your superclass.



Hierarchy Browser

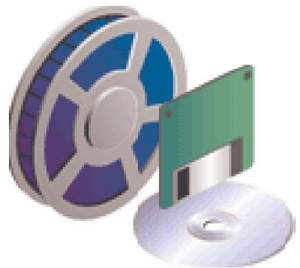
- Select the class and choose View Type Hierarchy from the shortcut menu.



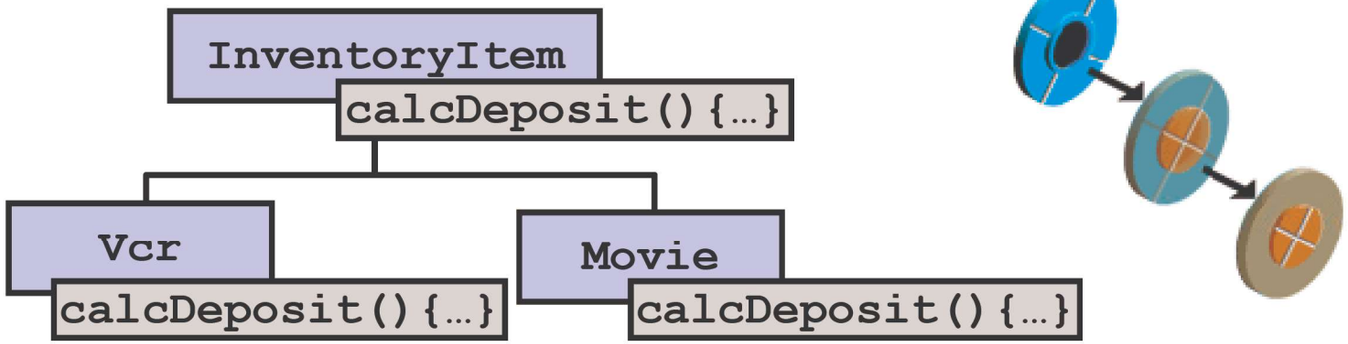
- The Hierarchy window displays the class hierarchy.

Acme Video and Polymorphism

- Acme Video started renting only videos.
- Acme Video added games and VCRs.
- What is next?
- Polymorphism solves the problem.



Using Polymorphism for Acme Video



ShoppingBasket

```
void addItem(InventoryItem item) {
    // this method is called each time
    // the clerk scans in a new item
    float deposit = item.calcDeposit();
    …
}
```

instanceof Operator

- You can determine the true type of an object by using an `instanceof` operator.
- An object reference can be downcast to the correct type if necessary.

```
public void aMethod(InventoryItem i) {
  …
  if (i instanceof Vcr)
    ((Vcr)i).playTestTape();
}
```

Limiting Methods and Classes with `final`

- You can mark a method as `final` to prevent it from being overridden.

```
public final boolean checkPassword(String p) {
   …
}
```

- You can mark a whole class as `final` to prevent it from being extended.

```
public final class Color {
   …
}
```

Ensuring Genuine Inheritance

- Inheritance must be used only for genuine “is a kind of” relationships:
  - It must always be possible to substitute a subclass object for a superclass object.
  - All methods in the superclass must make sense in the subclass.
- Inheritance for short-term convenience leads to problems in the future.

Summary

In this lesson, you should have learned the following:

- Inheritance and polymorphism are object-oriented programming techniques that involve the reuse of existing code.
- A subclass inherits all the variables and methods of its superclass.
- You can specify additional variables and methods and override methods.
- A subclass can call an overridden superclass method by using `super`.
- Polymorphism ensures that the correct version of a generic method is called at run time.

Practice : Overview

This practice covers the following topics:

- Defining subclasses of `Customer`
- Providing subclass constructors
- Adding new methods in the subclasses
- Overriding existing superclass methods