



Object Orientation Third Story

Bok, Jong Soon
javaexpert@nate.com
<https://github.com/swacademy/JavaSE>

Polymorphism

- Is a powerful feature of OO
- Means *many forms*
- Is the ability to have many different forms
- A reference variable has many forms ; it can refer to objects of different forms.
- Each of the subclasses is a form of the superclass it extended
- For example, polymorphism lets you refer to subclass objects using a reference to the superclass

Polymorphism (Cont.)

- Used in OOO to emphasize the fact that inheritance extends one class into one or more other classes.
- Use a reference variable of the superclass type to store the address of an object instantiated from one of its subclasses.
- Can be used in two ways:
 - Polymorphic parameters
 - Heterogeneous collections

Polymorphic Parameters

```
class Vet {  
    void vaccinate (Mammal m) {  
        //vaccinate m  
    }  
}  
class Mammal {  
    // Mammal members  
}  
class Dog extends Mammal {  
    // Dog members  
}  
class Cat extends Mammal {  
    // Cat members  
}
```

```
class Example {  
    public static void main(String [] args) {  
        Vet doctor = new Vet();  
        Dog myDog = new Dog();  
        Cat myCat = new Cat();  
        doctor.vaccinate(myDog);  
        doctor.vaccinate(myCat);  
    }  
}
```

The instanceof Operator

```
public class Employee extends Object
public class Manager extends Employee
public class Contractor extends Employee
-----
public void method(Employee e) {
    if (e instanceof Manager) {
        // Gets benefits and options
        // along with salary
    } else if (e instanceof Contractor) {
        // Gets hourly rates
    } else {
        // regular employee
    }
}
```

Heterogeneous Collections

- Collections with a common class are called *homogenous* collections.
- Collections with dissimilar objects are *heterogeneous* collections.
- It is not possible to mix the types of value being stored in an array using primitives.
- Heterogeneous collections are created using arrays of class types, where the array type is the super class.

```
Mammal [ ] mammalArray = new Mammal [10];  
mammalArray [0] = new Cat ();  
mammalArray [1] = new Dog ();  
mammalArray [2] = new Horse ();  
// and so on
```

Casting Objects

- Use `instanceof` to test the type of an object.
- Restore full functionality of an object by casting.
- Check for proper casting using the following guidelines:
 - Casts up hierarchy are done implicitly.
 - Downward casts must be to a subclass and checked by the compiler.
 - The object type is checked at runtime when runtime errors can occur.

Overriding Methods

- Methods in the same inheritance hierarchy can be overridden.
- If several classes inherit the same method, each subclass can override it.
- A subclass can modify behavior inherited from a parent class.
- A subclass can create a method with different functionality than the parent's method but with the same:
 - Name
 - Return type
 - Argument list

Overriding Methods (Cont.)

- Virtual method invocation:

```
Employee e = new Manager ();  
e.getDetails ();
```

- Compile-time type and runtime type

Rules About Overridden Methods

- Must have a return type that is identical to the method it overrides.
- Cannot be less accessible than the method it overrides.