Polymorphism, Dynamic Binding and Interface

2 – 4 pm Thursday 7/31/2008 @JD2211

1

Announcement

- Next week is off
- The class will continue on Tuesday, 12th August

2

Agenda

- Review
  - Inheritance
  - Abstract
  - Array
- Polymorphism
- Dynamic binding
- Interface

3

Review - Inheritance

```
public class SalariedEmployee extends
  Employee {
  ……
}
```

4

Review - Abstract

```
public abstract class Employee{
     ……
      public abstract double  earnings();
}
```

5

Review - Array

- `Employee[6]` refers to the seventh Employee reference in the array Employee
- In Java, an array is an object that must be instantiated

```
Employee employees[] = new Employee[4];
```

6

Polymorphism

- The term *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 in time.

7

Polymorphism (cont'd)

- For example, if the reference `employee` is polymorphic, it can refer to different types of `employee` at different time.
- If the `employee` reference is in a loop, that line of code could call a different version of earnings method each time it is invoked

8

Binding

- At some point, the commitment is made to execute certain code to carry out a method invocation.
- This commitment is referred to as binding a method invocation to a method definition
- In many situations, the binding of a method invocation to a method definition can occur at compile time

9

Static Binding

- In many situations, the binding of a method invocation to a method definition can occur at compile time

10

Dynamic Binding

- For polymorphic references, however, the decision cannot be made until run time.
- The method definition that is used is based on the object that is being referred to by the reference variable at that moment
- This deferred commitment is called dynamic binding

11

Static and Dynamic Binding

- Dynamic binding is less efficient than binding at compile time because the decision must be made during the execution of the program
- This overhead is generally acceptable in light of flexibility that a polymorphic reference provides

12

instanceof

- It checks an object reference if that defined by a class or its subclasses and returns a boolean value;

```
<object-reference> instanceof
ClassName
```

13

getClass()

- `public final Class getClass()`
- Returns the runtime class of an object
- http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Object.html

getName()

- `String getName()`
- Returns the fully-qualified name of the entity represented by this Class object, as a String.

```
void printClassName(Object obj) {
System.out.println("The class of " + obj
+ " is " + obj.getClass().getName()); }
```

- http://java.sun.com/j2se/1.3/docs/api/java/lang/Class.html

15

Exercise 1

- Fill the blanks in each of the following statements:
  - Polymorphism helps eliminates _____ logic.
  - If a class contains at least one abstract method, it is a(n) _____ abstract.
  - Classes from which objects can be instantiated are called _____ classes.

16

Exercise 2

- State whether each of the statements that follows is true of false. If false, explain why.
  - It is possible to treat superclass objects and subclass objects similarly.
  - All methods in an abstract class must be declared as abstract methods
  - Referring to a subclass object with a superclass variable is dangerous

17

Exercise 3

- State whether each of the statements that follows is true of false. If false, explain why.
  - A class is made abstract by declaring it abstract
  - If a superclass declares an abstract method, a subclass must implement that method to become a concrete class

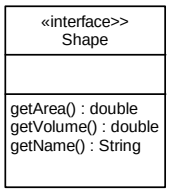
18

Interface

- A Java interface is a collection class constants and abstract methods.
- An interface is typically used in place of an abstract class when there is NO default implementation to inherit – i.e. no instance variables and no default method implementation

19

Interface – UML Notation



20

Interface Class in Java

```
public interface Shape {
      public double getArea();
      public double getVolume();
      public String getName();
}
```

21

Interface Class in Java

```
public interface Constants {
      public static fianl int ONE = 1;
      public static fianl int TWO = 2;
      public static fianl int THREE = 3;
}
```

22

Lab Assignment

- http://www.csun.edu/~twang/Java/Lab/7-31-Lab.pdf

23