


Core Java Course
Interfaces and Inner Classes
(Chapter 6)


Dr. Jeff Blessing
MSOE



Agenda

- ❖ Homework
- ❖ Interfaces
- ❖ Code Examples
- ❖ Cloning
- ❖ Inner Classes
- ❖ Code Examples


2



Why Interfaces

- ❖ Problem statement: Inherit behavior of two different classes
 - ❖ Example: Need to scale up or scale down different shapes (Circle, Triangle, etc... derived from Shape).
 - ❖ Possible solution in C++: derive Circle and other shape subclasses from both Shape and Scalable


3



Superclasses

- ⊗ A class in Java can have only one superclass
- ⊗ Java does not have multiple inheritance
- ☺ The solution: Java provides **interfaces** as the way to capture behavior of more than one class


4



What is an Interface?

- ❖ **Interface** - a promise (a contract) of a class to implement certain methods with specified return type and a list of arguments.
- ❖ Another way to implement polymorphism (using interface type instead of superclass type) *without requiring the use of inheritance!*
- ❖ Compiler **requires** implementing interfaces however it is concerned about signatures only. The actual implementation does not matter
- ❖ Interface is a preferred way of implementing callback methods

5



Example

```

class Circle extends Shape
{ Point center;
  Integer radius;
}
public interface Scalable
{
    public void changeScale(int percent);
}
class ScalableCircle extends Circle implements Scalable
{
    // changeScale() method must be implemented here
    public void changeScale(int percent){. . .};
}
    
```

6

Interfaces vs. Classes (similarities)

- ❖ A variable can be declared using *either* its **class** type *or* its **interface** type:

```
Scalable c = new ScalableCircle();
```
- ❖ Both can be extended:

```
public interface Readable
{
    public String read();
}
public interface Searchable extends Readable
{
    public int findPosition();
}
```

7

Interfaces vs. Classes (differences)

- ❖ Interfaces cannot specify static methods
- ❖ Interfaces cannot have instance variables
- ❖ Interfaces cannot be instantiated using `new`
- ❖ One class may implement multiple interfaces

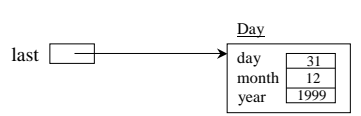
```
public class Word implements Searchable,
    Replaceable
{
    public int findPosition(Searchable s) {...}
    public void replace(Replaceable w1,
        Replaceable w2) {...}
}
```

8

Cloning

- ❖ Also referred to as “deep copy”
- ❖ By default, Java does “shallow copy”

```
Day last = new Day(1999, 12, 31);
```

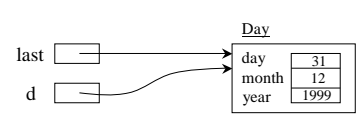


9

Cloning

- ❖ Also referred to as “deep copy”
- ❖ By default, Java does “shallow copy”

```
Day last = new Day(1999, 12, 31);
Day d = last;
```

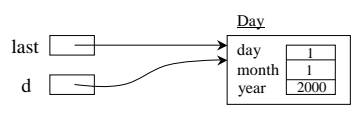


10

Cloning

- ❖ Also referred to as “deep copy”
- ❖ By default, Java does “shallow copy”

```
Day last = new Day(1999, 12, 31);
Day d = last;
d.advance(1); //also changed last!
```



11

Correct Solution

```
public class Day implements Cloneable
{
    . . .
    public Object clone()
    {
        return new Day(day, month, year);
    }
    private int day;
    private int month;
    private int year;
}
```

12

Cloning (deep copy)

```
Day last = new Day(1999, 12, 31);
```

13

Cloning (deep copy)

```
Day last = new Day(1999, 12, 31);
Day d = (Day) last.clone();
```

14

Cloning (deep copy)

```
Day last = new Day(1999, 12, 31);
Day d = (Day) last.clone();
d.advance(1); //OK, last is unaffected!
```

15

Problems with = (shallow copy)

```
Employee al = new Employee("Al", 50000,
    new Day(1988, 8, 18));
Employee bob = (Employee) al.clone();
bob.setName("Bob");
bob.setSalary(75000);
bob.setHireDay(1999, 9, 19);
```

❖ Is there a problem with the above code?!
 ❖ We just changed Al's hire date, as well as Bob's!

16

Al and Bob

17

The Cloneable Interface

❖ Solution: implement the Cloneable interface
 ❖ and override the clone() method

```
public class Employee implements Cloneable
{
    ...
    public Object clone()
    {
        try
        {
            Employee e = (Employee) super.clone();
            e.hireDay = (Day) hireDay.clone();
            return e;
        }
        catch (CloneNotSupportedException e)
        {
            return null;
        }
    }
}
```

18

The Cloneable Interface

- ❖ User code cannot call the `clone()` method unless the class implements the `Cloneable` interface (else an exception is thrown)
 - ❖ Thus `Cloneable` is called a *tagging interface*
- ❖ The `Object` class does not implement the `Cloneable` interface
 - ❖ It merely provides a `protected clone()` method as a convenience to implementors (subclasses)

19

Implementing Callbacks in Java

- ❖ **Callback** - a method that gets called when a certain event or action occurs.
- ❖ Examples:
 - ❖ Mouse click - `mouseClicked()`
 - ❖ Timer expiration - `timerExpired()`
 - ❖ Window closing - `windowClosed()`
- ❖ C++ uses function pointers to implement callbacks
- ❖ Java does not have function pointers. Interfaces are used to specify the callback function signature without specifying the class it belongs to.

20

Timer Code Examples

```
interface TimerListener
{ public void timeElapsed(Timer t); }

class AlarmClock implements TimerListener
{ AlarmClock()
  { t = new Timer(this);
    t.setDelay(100); // 100 millisc.
  }
  public void timeElapsed(Timer t)
  { if(t.getTime() >= wakeUpTime)
    wakeUp.play();
  }
  Timer t; ...
}
```

21

Timer Implementation

```
class Timer extends Thread
{ Timer(TimerListener t1)
  { listener = t1; }
  ...
  public void run()
  { while (true)
    { sleep(interval);
      listener.timeElapsed(this);
    }
  }
  private TimerListener listener;
}
```

22

Inner classes (Java 1.1)

- ❖ **Inner class** - a class defined inside another class
- ❖ Invisible to other classes
- ❖ Has an implicit reference to the “outer” class (not available to other code, even in same package!)
- ❖ Can access private data of the outer class
- ❖ Inner classes are always private
- ❖ Translated into a separate file with the name `OuterClass$InnerClass.class`
- ❖ Can be anonymous (useful for callbacks, etc.)

23

Property Editor Code Example

- ❖ Review the code in the example file:
 - `PropertyTest.java`

24



Homework Assignment

- ❖ Improve the ShapeBox example:
 - ❖ implement a Movable interface for all shapes
 - ❖ implement a Scalable interface for all shapes
 - ❖ test it using “draw”, “move”, “scaleUp”, and “scaleDown” commands

25



26