6.178: Introduction to Software Engineering in Java

Lecture 6: Abstract Classes & Interfaces

Abstract Classes

- May not be instantiated
- Methods may or may not be implemented
- Fields don’t have to be static and final
- May have public, protected, and private concrete methods
- Each class may extend only one abstract class

Why bother?

- I have to write a concrete class anyway. Why abstract class as well?
  - Closely related subclasses (Don’t repeat yourself)
  - Many shared fields & methods
  - Access modifiers other than public
  - Declare non-static and non-final fields
    - Allows for the changing of state

Examples:

- Abstract Class: AbstractMap
  - HashMap extends AbstractMap
  - TreeMap extends AbstractMap
  - ConcurrentHashMap extends AbstractMap

Lecture Exercise:

- Create an abstract class for a 2D Shape:
  - Shapes have a center at x and y.
- Create fields for x, y
- Create a constructor that sets all of these fields
- Create concrete getter methods getCenter
- Create abstract methods getPerimeter and getArea

Lecture Exercise:

- Create concrete Rectangle class extending Shape
  - Create length and width fields
  - Modify the constructor to accept fields indicating width and length
  - Override getPerimeter and getArea

- Create your own shape that extends Shape

Interfaces

- Also may not be instantiated
- Methods are **NOT** implemented
- Fields must be static and final
- Only abstract methods may be declared
- A class may implement many interfaces

Why bother?

- Completely unrelated classes may implement your interface
- You only care about specification not implementation
- You want to take advantage of the fact that a class may implement multiple interfaces

Examples

- HashMap implements several interfaces:
  - Serializable (May be converted to byte stream)
  - Cloneable
  - Map<K, V>

Lecture Exercise:

- Create Interface Resizable:
  - Has abstract method scaleArea(double factor)
  - That method scales the area of the Shape by the given factor

- Change your Rectangle shape and other shape so that they implement Resizable

Colorable Interface & Implementation

Rectangle.java

```java
package lecture6_exercises;

public class Rectangle extends Shape implements Resizable, Colorable {

    private String color = "white";

    @Override
    public void setColor(String color) {
        this.color = color;
    }

    @Override
    public String getColor() {
        return color;
    }
```

Colorable.java

```java
package lecture6_exercises;

public interface Colorable {

    public abstract void setColor(String color);

    public abstract String getColor();

}
```