Member Accessibility and Object Initialization

Objectives

- Learn about access level modifiers.
- Be able to make effective use of the java keywords this and super.
Outline

- Access level modifiers.
- Object initialization sequence.
- Using \textit{this} and \textit{super} in constructors.
- Using \textit{this} and \textit{super} in methods.

Method and Field Access Levels

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Class</th>
<th>Package</th>
<th>Subclass</th>
<th>World</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>protected</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>no modifier</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>private</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

- To reduce maintenance costs, access should be as restricted as practical. (Why?)
Inheritance of private fields

- When extending an existing class, all of the attributes (fields) and operations (methods) are inherited – even if they are private.
- However, if they are private, they cannot be accessed by code written in the subclass.
  - Solution: The subclass can access private fields using the public accessor and mutator methods that it inherits.

Example: Inheritance of private field

```java
public class A {
    private int n;
    public void setN(int i) {
        n = i;
    }
}

class B extends A {
    public B () {
        n = 5;  // Error: n is private
        this.setN(5);  // OK
    }
}
```
**Object Initialization Sequence**

- The *new* operator invokes a constructor for the object being created with a matching signature.
- The constructor:
  1. Invokes a constructor from its superclass.
  2. Performs field initializations specified in the field definitions.
  3. Executes the code in the body of the constructor.
- In step 1, the superclass constructor behaves the same way, invoking a constructor from *its* superclass, etc.

**Object Initialization Sequence (2)**

- The superclass constructor may be invoked explicitly in the first line of code of a constructor. Otherwise, the default constructor from the superclass will be used.
- If the superclass does not have a default constructor, the superclass constructor *must* be invoked explicitly.
- A constructor can explicitly invoke a constructor from its superclass using the keyword *super*. 
Example: Accounts

```java
public class Account {
    private int accountNumber;
    private double balance;

    public Account(int num, double bal) {
        accountNumber = num;
        balance = bal;
    }
}
```

```java
public class OverdraftAccount extends Account {
    private double overdraftLimit;

    public OverdraftAccount(int num, double bal, double limit) {
        super(num, bal);
        overdraftLimit = limit;
    }
}
```

Invoking Superclass Methods

- Just as constructors can invoke a superclass constructor, methods can invoke methods defined in their superclass – even if they have been overridden.
- This is most common if we want to override a method to do more than the inherited method.
  - The overriding method may invoke the superclass method, and then do extra work.
- To invoke an inherited method that is not overridden, or the new version of an overridden method, an object sends itself a message using this.
- To invoke the original version of an overridden method, an object sends itself a message using super.
Rectangle class with private fields and no default constructor

```java
public class Rectangle {
    private int width;
    private int height;

    public Rectangle(int w, int h) {
        width = w;
        height = h;
    }

    public void setWidth(int w) {
        width = w;
    }

    public void setHeight(int h) {
        height = h;
    }

    public int getArea() {
        return width * height;
    }

    public int getPerimeter() {
        return 2*(width + height);
    }
}
```

Square class

```java
public class Square extends Rectangle {
    public Square(int length) {
        super(length, length); // invoke superclass constructor
    }

    public void setWidth(int w) {
        super.setWidth(w);
        super.setHeight(w);
    }

    public void setHeight(int h) {
        super.setHeight(h);
        super.setWidth(h);
    }
}
```
Invoking Constructors with this

- A constructor may use the keyword this to invoke other constructors of the same class. For example, there are 3 FlowLayout constructors:

```java
public class FlowLayout {
    public FlowLayout(int align, int hgap, int vgap) {
        // Do all the real work here!
    }

    public FlowLayout() {
        this(FlowLayout.CENTER, 5, 5);
    }

    public FlowLayout(int align) {
        this(align, 5, 5);
    }
}
```