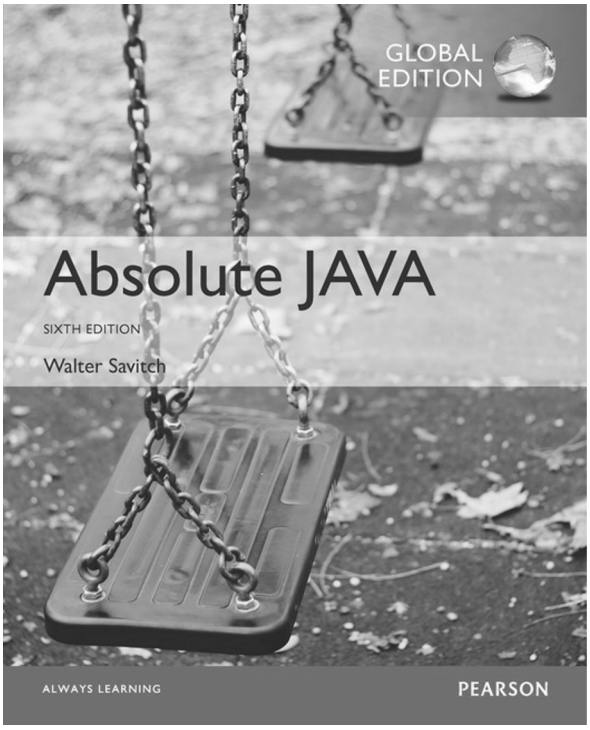

Chapter 13

Inner Classes

Slides prepared by Rose Williams, *Binghamton University*

Kenrick Mock, *University of Alaska Anchorage*

Simple Uses of Inner Classes

- Inner classes are classes defined within other classes
  - The class that includes the inner class is called the outer class
  - There is no particular location where the definition of the inner class (or classes) must be place within the outer class
  - Placing it first or last, however, will guarantee that it is easy to find

Simple Uses of Inner Classes

- An inner class definition is a member of the outer class in the same way that the instance variables and methods of the outer class are members
  - An inner class is local to the outer class definition
  - The name of an inner class may be reused for something else outside the outer class definition
  - If the inner class is private, then the inner class cannot be accessed by name outside the definition of the outer class

Simple Uses of Inner Classes

- There are two main advantages to inner classes
  - They can make the outer class more self-contained since they are defined inside a class
  - Both of their methods have access to each other's private methods and instance variables
- Using an inner class as a helping class is one of the most useful applications of inner classes
  - If used as a helping class, an inner class should be marked private

Tip: Inner and Outer Classes Have Access to Each Other's Private Members

- Within the definition of a method of an inner class:
  - It is legal to reference a private instance variable of the outer class
  - It is legal to invoke a private method of the outer class
- Within the definition of a method of the outer class
  - It is legal to reference a private instance variable of the inner class on an object of the inner class
  - It is legal to invoke a (nonstatic) method of the inner class as long as an object of the inner class is used as a calling object
- Within the definition of the inner or outer classes, the modifiers **`public`** and **`private`** are equivalent


```
public class BankAccount
{
    private class Money
    {
        private long dollars;
        private int cents;

        public Money(String stringAmount)
        {
            abortOnNull(stringAmount);
            int length = stringAmount.length();
            dollars = Long.parseLong(
                        stringAmount.substring(0, length - 3));
            cents = Integer.parseInt(
                        stringAmount.substring(length - 2, length));
        }

        public String getAmount()
        {
            if (cents > 9)
               return (dollars + "." + cents);
            else
               return (dollars + ".0" + cents);
        }
```

The modifier `private` in this line should not be changed to `public`. However, the modifiers `public` and `private` inside the inner class `Money` can be changed to anything else and it would have no effect on the class `BankAccount`.

13.9 Class with an Inner Class *(Part 1 of 2)* (continued)

```
    public void addIn(Money secondAmount)
    {
        abortOnNull(secondAmount);
        int newCents = (cents + secondAmount.cents)%100;
        long carry = (cents + secondAmount.cents)/100;
        cents = newCents;
        dollars = dollars + secondAmount.dollars + carry;
    }

   private void abortOnNull(Object o)
   {
       if (o == null)
       {
            System.out.println("Unexpected null argument.");
            System.exit(0);
       }
    }
}
```

The definition of the inner class ends here, but the definition of the outer class continues in Part 2 of this display.

```
    private Money balance;

    public BankAccount()
    {
        balance = new Money("0.00");
    }

    public String getBalance()
    {
        return balance.getAmount();
    }

    public void makeDeposit(String depositAmount)
    {
        balance.addIn(new Money(depositAmount));
    }

    public void closeAccount()
    {
        balance.dollars = 0;
        balance.cents = 0;
    }
}
```

To invoke a nonstatic method of the inner class outside of the inner class, you need to create an object of the inner class.

This invocation of the inner class method `getAmount()` would be allowed even if the method `getAmount()` were marked as `private`.

Notice that the outer class has access to the private instance variables of the inner class.

This class would normally have more methods, but we have only included the methods we need to illustrate the points covered here.

The `.class` File for an Inner Class

- Compiling any class in Java produces a **`.class`** file named ***`ClassName`*`.class`**
- Compiling a class with one (or more) inner classes causes both (or more) classes to be compiled and produces two (or more) .class files
  - Such as ***`ClassName`*`.class` and ***`ClassName$InnerClassName`*`.class`**

Anonymous Classes

- If an object is to be created, but there is no need to name the object's class, then an *anonymous class* definition can be used
  - The class definition is embedded inside the expression with the **`new`** operator
- Anonymous classes are sometimes used when they are to be assigned to a variable of another type
  - The other type must be such that an object of the anonymous class is also an object of the other type
  - The other type is usually a Java interface

ıy 13.11 Anonymous Classes *(Part 1 of 2)*

This is just a toy example to demon
the Java syntax for anonymous clas

```
public class AnonymousClassDemo
{
    public static void main(String[] args)
    {
        NumberCarrier anObject =
                new NumberCarrier()
                {
                    private int number;
                    public void setNumber(int value)
                    {
                        number = value;
                    }
                    public int getNumber()
                    {
                       return number;
                    }
                };
```

```
        NumberCarrier anotherObject =
                new NumberCarrier()
                {
                    private int number;
                    public void setNumber(int value)
                    {
                        number = 2*value;
                    }
                    public int getNumber()
                    {
                        return number;
                    }
                };

        anObject.setNumber(42);
        anotherObject.setNumber(42);
        showNumber(anObject);
        showNumber(anotherObject);
        System.out.println("End of program.");
    }

    public static void showNumber(NumberCarrier o)
    {
        System.out.println(o.getNumber());
    }
}
```

This is still the file AnonymousClassDemo.java.

Anonymous Classes

Display 13.11 Anonymous Classes *(Part 2 of 2)*

SAMPLE DIALOGUE

```
42
84
End of program.
```

This is the file NumberCarrier.java.

```
1  public interface NumberCarrier
2  {
3      public void setNumber(int value);
4      public int getNumber();
5  }
```