6.3 Designing Methods
Top-Down Design

- In pseudocode, **write a list of subtasks** that the method must do.
- If you can easily write Java statements for a subtask, you are finished with that subtask.
- If you cannot easily write Java statements for a subtask, treat it as a new problem and break it up into a list of subtasks.
- Eventually, all of the subtasks will be small enough to easily design and code.
- Solutions to subtasks might be implemented as [redacted] methods.
- Top-down design is also known as [redacted] or ***stepwise refinement.***

Programming Tips for Writing Methods

- Apply the principle of encapsulation and detail hiding by using the **`public` and `private`** modifiers judiciously
  - » If the user will need the method, make it part of the interface by declaring it [redacted]
  - » If the method is used only within the class definition (a *helper* method, then declare it [redacted]

- Create **a [redacted] method with diagnostic (test) code** within a class's definition
  - » run just the class to execute the diagnostic program
  - » when the class is used by another **program the class's `main` is** [redacted]

Testing a Method

- Test programs are sometimes called [redacted] **programs**
- Keep it simple: test [redacted]
  - » driver program should have **only one untested method**
- If method A uses method B, there are two approaches:
- *Bottom up*
  - » test method B fully before testing A
- *Top down*
  - » test method A and use [redacted] for method B
  - » A *stub* is a method that stands in for the final version and does little actual work. It usually does something as trivial as printing a message or returning a fixed value. The idea is to have it so simple you are nearly certain it will work.

아무일을 하지 X

Listing 6.12 The DollarsFirstTry CLass -DollarFormatFirstTry.java

```
// Listing 6.12 The DollarFormatFirstTry CLass

public class DollarFormatFirstTry
{
   /**
    Outputs amount in dollars and cents notation.
    Rounds after two decimal points.
    Does not advance to the next line after output.
   */
   public static void write(double amount)
   {
      int allCents = (int)(Math.round(amount*100));  //-120
      int dollars = allCents/100;   // -1
      int cents = allCents%100;  //-20
```

```
        System.out.print('$');
        System.out.print(dollars);
        System.out.print('.');    // print("$-1.
        if (cents < 10)    // -20 < 10
        {
            System.out.print('0');  //  print("0
            System.out.print(cents);  // print("-20
        }
        else
            System.out.print(cents);
    }
    /**
     Outputs amount in dollars and cents notation.
     Rounds after two decimal points.
     Advances to the next line after output.
    */
    public static void writeln(double amount)
    {
        write(amount);
        System.out.println( );
    }
}
```

```
// Lising 6.13 A Driver that Tests DollarFormatFirstTry
// This kind of testing program is often called a driver program.
import java.util.Scanner;
public class DollarFormatFirstTryDriver
{
   public static void main(String[] args)
   {
      double amount;
      String response;
      Scanner keyboard = new Scanner(System.in);
      System.out.println("Testing DollarFormatFirstTry.write:");
      do
      {
         System.out.println("Enter a value of type double:");
         amount = keyboard.nextDouble( );
         DollarFormatFirstTry.write(amount);
         System.out.println( );
         System.out.println("Test again?");
         response = keyboard.next( );
      } while (response.equalsIgnoreCase("yes"));
      System.out.println("End of test.");
   }
}
```

C:₩WINDOWS₩system32₩cmd.exe

```
Testing DollarFormatFirstTry.write:
Enter a value of type double:
1.2345
$1.23
Test again?
yes
Enter a value of type double:
1.235
$1.24
Test again?
yes
Enter a value of type double:
9.02
$9.02
Test again?
yes
Enter a value of type double:
-1.20
$-1.0-20
Test again?
no
End of test.
계속하려면 아무 키나 누르십시오 . . .
```

Listing 6.12 The DollarsFirstTry CLass -DollarFormatFirstTry.java

```
// Listing 6.12 The DollarFormatFirstTry CLass

public class DollarFormatFirstTry
{
   /**
    Outputs amount in dollars and cents notation.
    Rounds after two decimal points.
    Does not advance to the next line after output.
   */
   public static void write(double amount)
   {
      int allCents = (int)(Math.round(amount*100));  //-120
      int dollars = allCents/100;   // -1
      int cents = allCents%100;  //-20
```

```
        System.out.print('$');
        System.out.print(dollars);
        System.out.print('.');    // print("$-1.
        if (cents < 10)    // -20 < 10
        {
            System.out.print('0');  //  print("0
            System.out.print(cents);  // print("-20
        }
        else
            System.out.print(cents);
    }
    /**
     Outputs amount in dollars and cents notation.
     Rounds after two decimal points.
     Advances to the next line after output.
    */
    public static void writeln(double amount)
    {
        write(amount);
        System.out.println( );
    }
}
```

Lising 6.14 The Corrected Dollars b - DollarFormat.java

```
// Listing 6.14. The Corrected Class DollarFormat

public class DollarFormat
{
        /**
         Outputs amount in dollars and cents notation.
         Rounds after two decimal points.
         Advances to the next line after output.
        */
        public static void writeln(double amount)
        {
            write(amount);
            System.out.println( );
        }
```

```
/**
   Outputs amount in dollars and cents notation.
   Rounds after two decimal points.
   Does not advance to the next line after output.
  */
  public static void write(double amount)
  {
     if (amount >= 0)
     {
        System.out.print('$');
        writePositive(amount);
     }
     else
     {
                                  // the case of negative amounts of money
        double positiveAmount = -amount;   //1.20
        System.out.print('$');             //$
        System.out.print('-');             //$-
        writePositive(positiveAmount);
     }
  }
```

```
//Precondition: amount >= 0;
   //Outputs amount in dollars and cents notation. Rounds
   //after two decimal points. Omits the dollar sign.
   private static void writePositive(double amount)
   {
      int allCents = (int)(Math.round(amount*100));   //120
      int dollars = allCents/100;                     //1
      int cents = allCents%100;                       //20

      System.out.print(dollars);                      //1
      System.out.print('.');                          //1.
      if (cents < 10)
      {
         System.out.print('0');
         System.out.print(cents);
      }
      else
         System.out.print(cents);                     //20
   }
}
```

```
//  . The Corrected Class Dollars
import java.util.Scanner;

public class DollarFormatDriver
{
   public static void main(String[] args)
   {
      double amount;
      String ans;
      Scanner keyboard = new Scanner(System.in);
      System.out.println("Testing DollarFormat.write:");
      do        {
         System.out.println("Enter a value of type double:");
         amount = keyboard.nextDouble( );
         DollarFormat.write(amount);
         System.out.println( );
         System.out.println("Test again?");
         ans = keyboard.next( );
      } while (ans.equalsIgnoreCase("yes"));
      System.out.println("End of test.");
   }
}
```

C:₩WINDOWS₩system32₩cmd.exe

```
Testing DollarFormat.write:
Enter a value of type double:
1.2345
$1.23
Test again?
yes
Enter a value of type double:
1.235
$1.24
Test again?
yes
Enter a value of type double:
9.02
$9.02
Test again?
yes
Enter a value of type double:
-1.20
$-1.20
Test again?
no
End of test.
계속하려면 아무 키나 누르십시오 . . .
```

끝