More about
Predefined Classes and Objects

Chapter 3

Sections 3.3, 3.5

Objectives

You will be able to:

- Use methods of the Math class to do mathematical calculations that are not defined directly in the Java language.
- Format floating point values for display.

Packages

Section 3.3

Class Libraries

- A *class library* is a collection of class definitions that we can use when developing programs.

- The *Java standard class library* is part of any Java development environment.
  - Its classes are not part of the Java language, but are always available to us.

- Various classes we've already used (**`System`** , **`Scanner`**, **`String`**) are part of the Java standard class library.

Packages

- The classes of the Java standard class library are organized into *packages*
- Some of the packages in the standard class library are:

<u>Package</u>	<u>Purpose</u>
java.lang	**General support**
java.util	**Utilities**
java.awt	Graphics and graphical user interfaces
javax.swing	Additional graphics capabilities
java.net	Network communication

The import Declaration

- When you want to use a class from a package, you can *import* the class, and then use the class name as if you had written the class as a part of your program.

```
import java.util.Scanner;
```

- To import *all* classes in a particular package, you can use the * wildcard character.

```
import java.util.*;
```

The import Declaration

- All classes of the **`java.lang`** package are imported automatically into all programs
- It's as if all programs contain the following line:

```
import java.lang.*;
```

- That's why we didn't have to import the **`System`** or **`String`** classes explicitly in earlier programs

The Math Class

Section 3.5

The Math Class

- The **`Math`** class is part of the **`java.lang`** package
  - Doesn't have to be imported.
- The **`Math`** class contains methods that perform various mathematical functions.
- These include:
  - absolute value, square root, exponentiation
  - trigonometric functions

Static Methods

Key Concept

- The methods of the **Math** class are *static methods* (also called *class methods*).
- Static methods can be invoked through the *class name*.
  - No *object* of the **Math** class is needed.

```
value = Math.cos(90) + Math.sqrt(delta);
```

Some Methods of the Math Class

- static int abs (int num)
  - Returns the absolute value of num.
- static double sin (double angle)
  - Returns the sine of angle measured in radians.
- static double ceil (double num)
  - Returns the ceiling of num, which is the smallest whole number greater than or equal to num.

Some Methods of the Math Class

- static double pow (double num, double power)
  - Returns the value num raised to the specified power.
- static double sqrt (double num)
  - Returns the square root of num, which must be positive.

- Note that the method names are all lower case single words.

```java
//********************************************************************
//  Quadratic.java       Author: Lewis/Loftus
//
//  Demonstrates the use of the Math class to perform a calculation
//  based on user input.
//********************************************************************

import java.util.Scanner;

public class Quadratic
{
   //-----------------------------------------------------------------
   //  Determines the roots of a quadratic equation.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      int a, b, c;  // ax^2 + bx + c
      double discriminant, root1, root2;

      Scanner scan = new Scanner (System.in);

      System.out.print ("Enter the coefficient of x squared: ");
      a = scan.nextInt();
```

continued

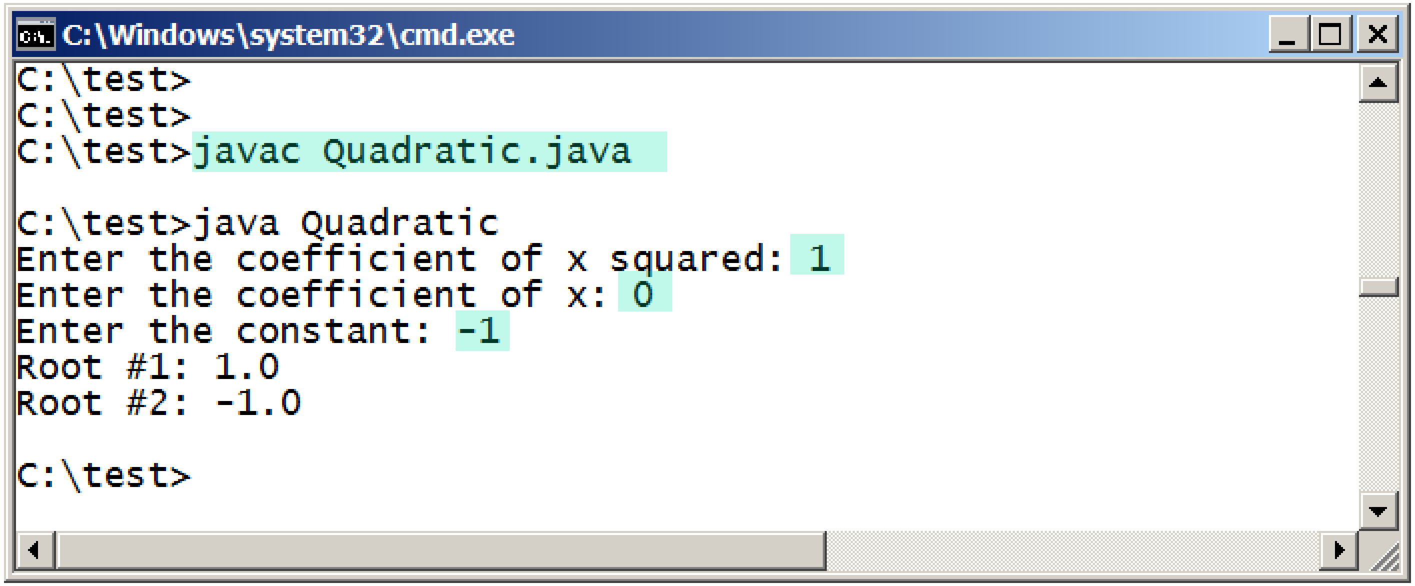
continued

```
      System.out.print ("Enter the coefficient of x: ");
      b = scan.nextInt();

      System.out.print ("Enter the constant: ");
      c = scan.nextInt();

      // Use the quadratic formula to compute the roots.
      // Assumes a positive discriminant.

      discriminant = Math.pow(b, 2) - (4 * a * c);
      root1 = ((-1 * b) + Math.sqrt(discriminant)) / (2 * a);
      root2 = ((-1 * b) - Math.sqrt(discriminant)) / (2 * a);

      System.out.println ("Root #1: " + root1);
      System.out.println ("Root #2: " + root2);
   }
}
```

Quadratic.java

- Let's use program Quadratic to solve for the roots of the equation

$$y = x^2 - 1$$



```
C:\test>
C:\test>
C:\test>javac Quadratic.java

C:\test>java Quadratic
Enter the coefficient of x squared: 1
Enter the coefficient of x: 0
Enter the constant: -1
Root #1: 1.0
Root #2: -1.0

C:\test>
```

End of Section

Floating-point Representation Error

- Not all real numbers can be represented exactly by a computer's floating point types.
  - There is a finite number of floating point values.
  - There is an infinite number of real numbers, and most of them cannot be represented exactly.

Floating-point Representation Error

- There are maximum and minimum values they can represent.
- Most importantly, they have limited, though large, precision.
- Most real numbers cannot be represented exactly in binary floating point representation regardless of how many bits we use.
  - Think about representing 1/3 as a decimal.

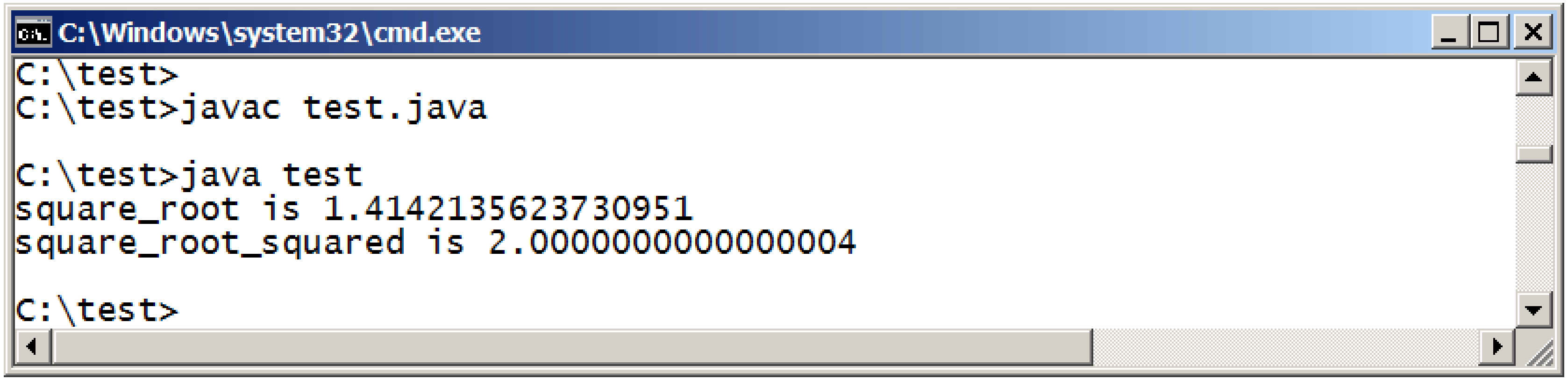
- Think of floating point values as *approximations*.
  - Fuzzy values.
  - Tests for equality are usually not meaningful.

Floating-point Round-off Error

- Results of floating point *calculations* may not be exact
  - even when the operands are exact.
  - Round-off error can build up over many calculations.

Floating-point Round-off Error

```
class test
{
    public static void main(String[] args)
    {
        double number = 2.0;
        double square_root = Math.sqrt(number);
        System.out.println("square_root is " + square_root);
        double square_root_squared = square_root * square_root;
        System.out.println("square_root_squared is " +
                           square_root_squared);
    }
}
```



```
C:\test>
C:\test>javac test.java

C:\test>java test
square_root is 1.4142135623730951
square_root_squared is 2.0000000000000004

C:\test>
```

End of Section

Formatting Output

Section 3.6

Formatting Output

- It is often necessary to format values in certain ways so that they can be presented properly.
- The Java standard class library contains classes that provide formatting capabilities.
- For example, a currency formatter can be used to format 5.8792 to monetary value $5.88
- A percent formatter can be used to format 0.492 to 49%.
  - Values are *rounded* to nearest value that can be represented.

Formatting Output

- The **`NumberFormat`** class allows you to format values as currency or percentages.
- The **`DecimalFormat`** class allows you to format values based on a pattern.
- Both are part of the **`java.text`** package.

Formatting Output

- The **`NumberFormat`** class has static methods that return a formatter object.

```
NumberFormat fmtCur = NumberFormat.getCurrencyInstance();

NumberFormat fmtPct = NumberFormat.getPercentInstance();
```

- Each formatter object has a method called **`format`** that returns a string with the specified information in the appropriate format.

Factory Methods

Key Concept

These objects are different from what we have seen before. You don't use *new* to instantiate them.

```
NumberFormat fmtCur = NumberFormat.getCurrencyInstance();

NumberFormat fmtPct = NumberFormat.getPercentInstance();
```

getCurrancyInstance() and getPercentInstance() are *factory methods*.

- Methods that create an object and return a reference to it.

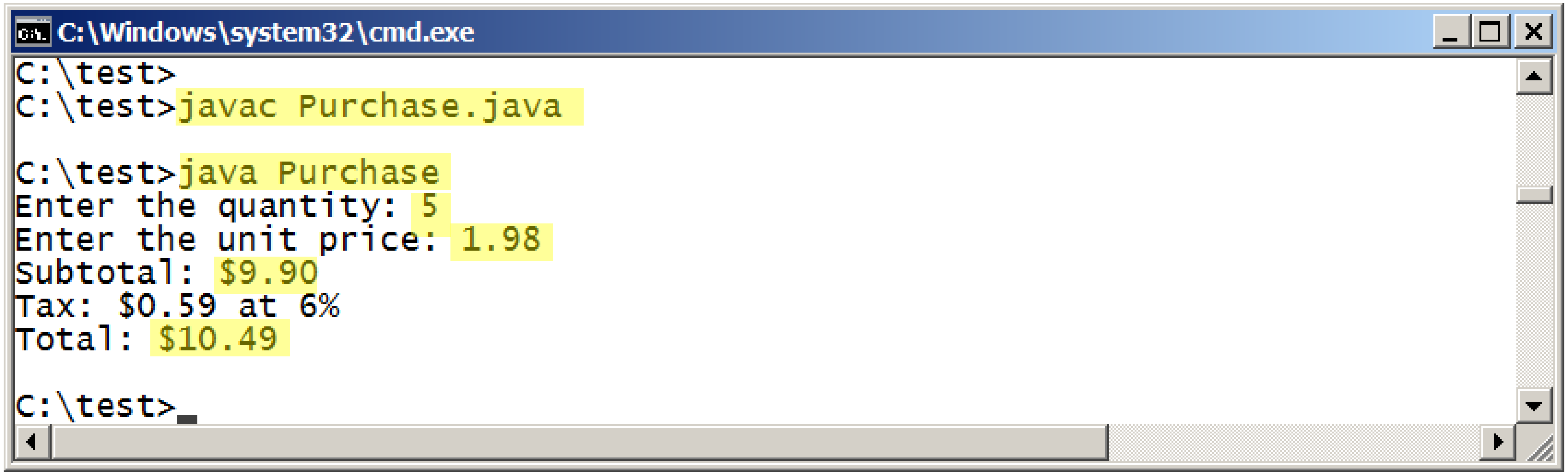
```
//********************************************************************
//  Purchase.java       Author: Lewis/Loftus
//
//  Demonstrates the use of the NumberFormat class to format output.
//********************************************************************

import java.util.Scanner;
import java.text.NumberFormat;

public class Purchase
{
   //-----------------------------------------------------------------
   //  Calculates the final price of a purchased item using values
   //  entered by the user.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      final double TAX_RATE = 0.06;  // 6% sales tax

      int quantity;
      double subtotal, tax, totalCost, unitPrice;

      Scanner scan = new Scanner (System.in);
```

continued

continued

```
      NumberFormat fmtCur = NumberFormat.getCurrencyInstance();
      NumberFormat fmtPct = NumberFormat.getPercentInstance();

      System.out.print ("Enter the quantity: ");
      quantity = scan.nextInt();

      System.out.print ("Enter the unit price: ");
      unitPrice = scan.nextDouble();

      subtotal = quantity * unitPrice;
      tax = subtotal * TAX_RATE;
      totalCost = subtotal + tax;

      // Print output with appropriate formatting
      System.out.println ("Subtotal: " + fmtCur.format(subtotal));
      System.out.println ("Tax: " + fmtCur.format(tax) + " at "
                          + fmtPct.format(TAX_RATE));
      System.out.println ("Total: " + fmtCur.format(totalCost));
   }
}
```

Purchase.java in Action



```
C:\test>
C:\test>javac Purchase.java

C:\test>java Purchase
Enter the quantity: 5
Enter the unit price: 1.98
Subtotal: $9.90
Tax: $0.59 at 6%
Total: $10.49

C:\test>_
```

End of Section

Formatting Decimal Values

- The **`DecimalFormat`** class can be used to format a floating point value in various ways.
- The constructor of the **`DecimalFormat`** class takes a string that represents a pattern for the formatted number.

```
DecimalFormat fmt = new DecimalFormat("0.###");
```

Indicates the fractional portion of the value should be rounded to three digits and show one leading zero.

Formatting Output

- Method format(double number)
  - Returns a string containing the specified number formatted according to the current pattern.
- For example, you can specify that the number should be rounded to three decimal digits.

  65.83752→65.838
- Note that the variable passed to the method is not affected.

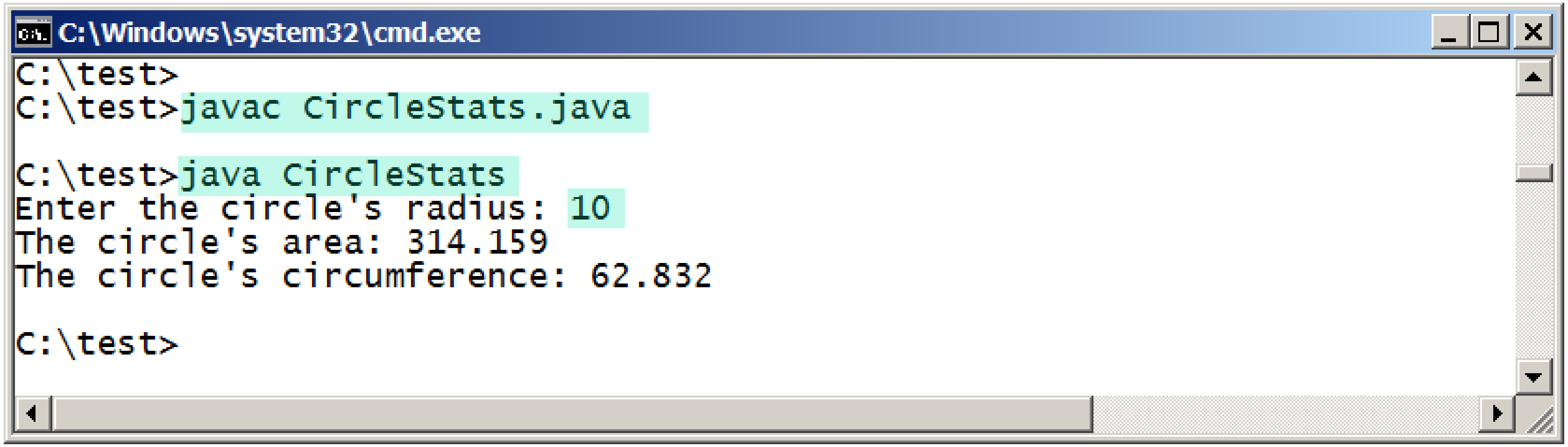
```
//********************************************************************
//  CircleStats.java       Author: Lewis/Loftus
//
//  Demonstrates the formatting of decimal values using the
//  DecimalFormat class.
//********************************************************************

import java.util.Scanner;
import java.text.DecimalFormat;

public class CircleStats
{
   //-----------------------------------------------------------------
   //  Calculates the area and circumference of a circle given its
   //  radius.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      int radius;
      double area, circumference;

      Scanner scan = new Scanner (System.in);
```

continued

continued

```
      System.out.print ("Enter the circle's radius: ");
      radius = scan.nextInt();

      area = Math.PI * Math.pow(radius, 2);
      circumference = 2 * Math.PI * radius;

      // Round the output to three decimal places
      DecimalFormat fmt = new DecimalFormat ("0.###");

      System.out.println ("The circle's area: " + fmt.format(area));
      System.out.println ("The circle's circumference: "
                          + fmt.format(circumference));
   }
}
```

Program CircleStats in Action



```
C:\test>
C:\test>javac CircleStats.java

C:\test>java CircleStats
Enter the circle's radius: 10
The circle's area: 314.159
The circle's circumference: 62.832

C:\test>
```

Readings and Assignments

- Reading: Chapter 3.3, 3.5, 3.6

- Self-Assessment Exercises:
  - Self-Review Questions
    SR 3.3, 3.4, 3.8, 3.9, 3.22, 3.28
  - After Chapter Exercises
    EX 3.1, 3.9

- These are not to be submitted in Canvas.
- Check your own answers
  - SR Answers in back of the book
  - EX Write a program if you are not sure.

Readings and Assignments

- Lab Assignment:
  - Project 4: Computing Distance

  http://www.csee.usf.edu/~turnerr/Programming_Concepts/045_Project_4_Computing_Distance.pdf

- Project to be submitted in Canvas.