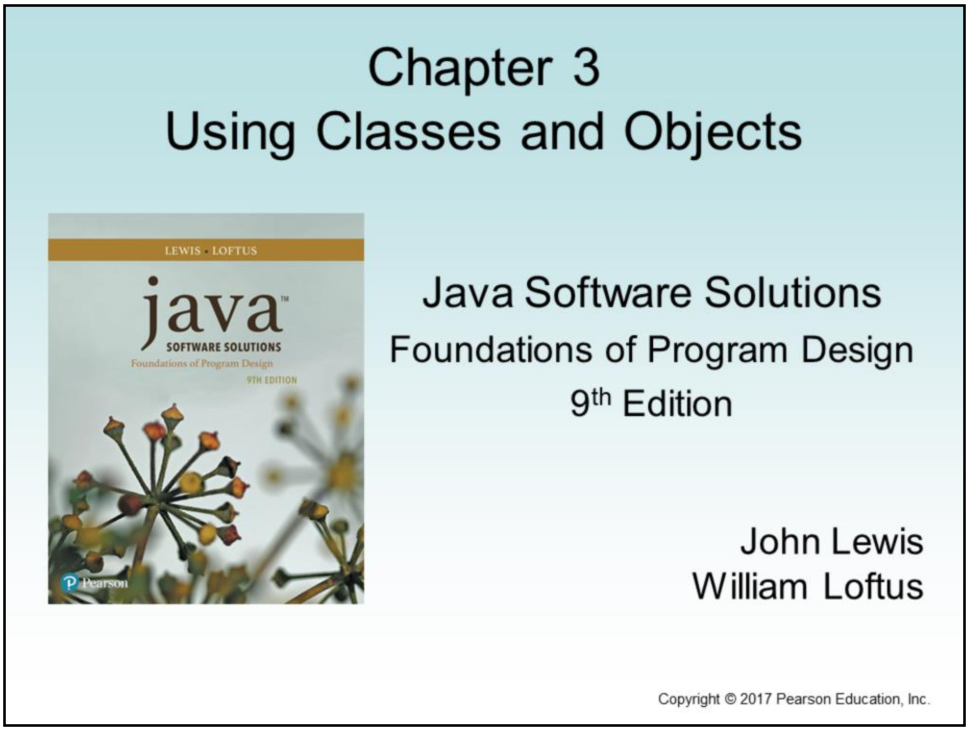


Chapter 3
Using Classes and Objects

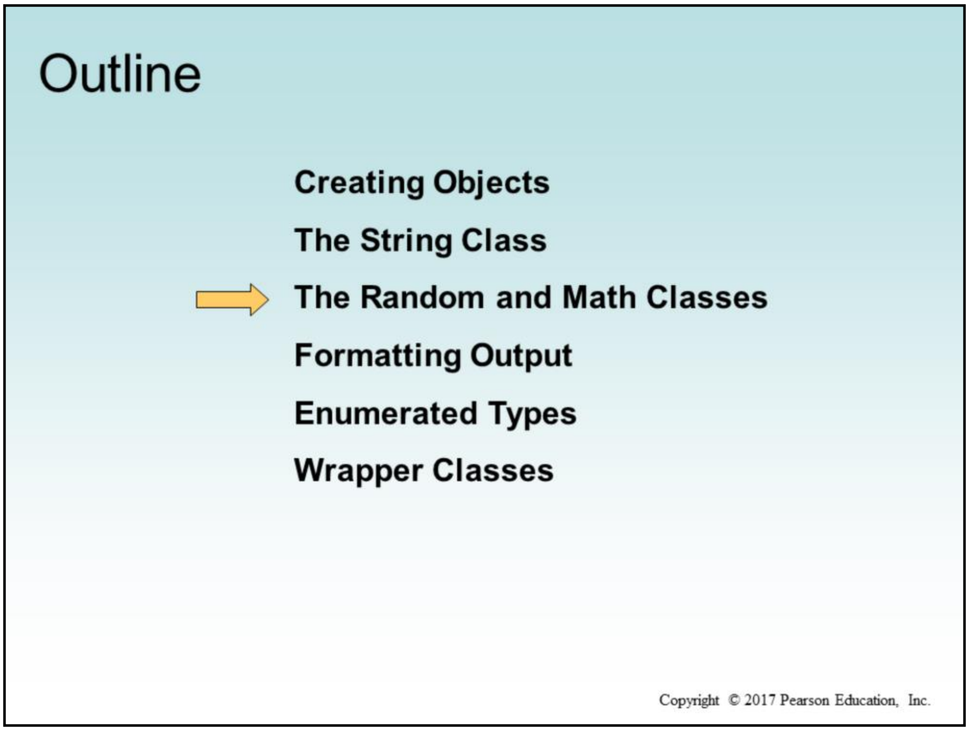
LEWIS · LOFTUS
java™
SOFTWARE SOLUTIONS
Foundations of Program Design
9TH EDITION
Pearson

Java Software Solutions

Foundations of Program Design

9th Edition

John Lewis
William Loftus



Outline

Creating Objects

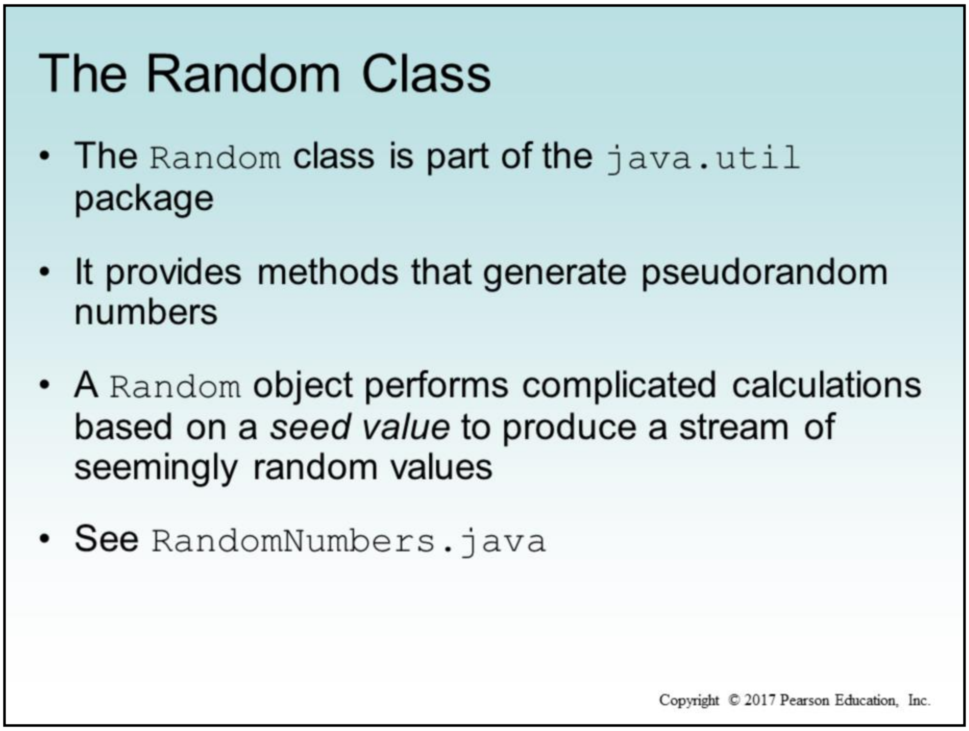
The String Class

The Random and Math Classes

Formatting Output

Enumerated Types

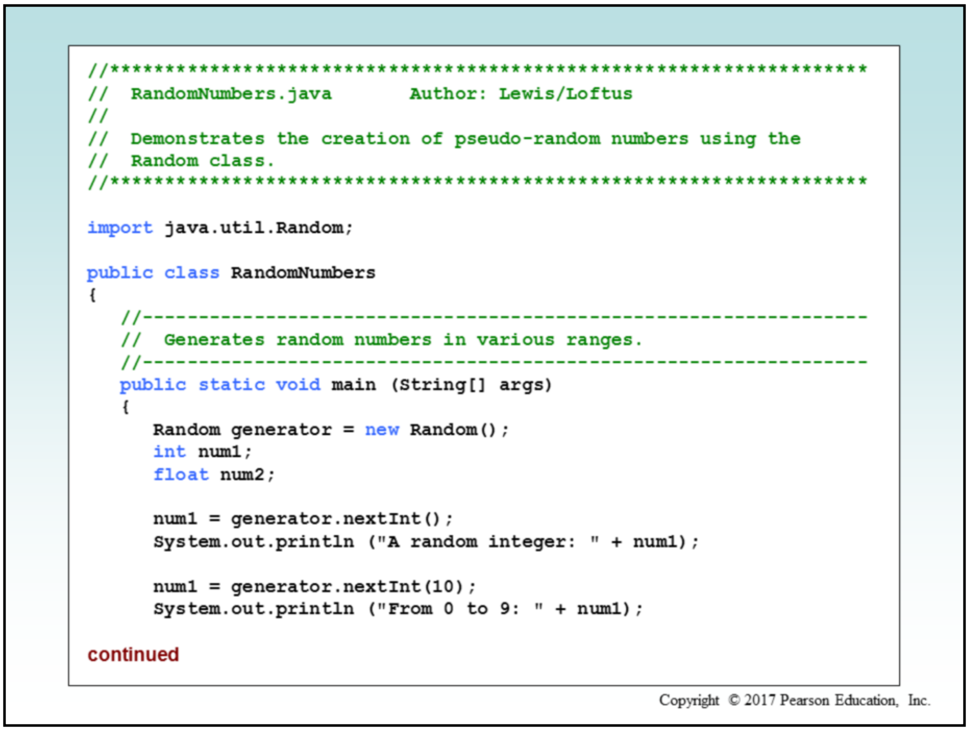
Wrapper Classes



The Random Class

- The `Random` class is part of the `java.util` package
- It provides methods that generate pseudorandom numbers
- A `Random` object performs complicated calculations based on a *seed value* to produce a stream of seemingly random values
- See `RandomNumbers.java`

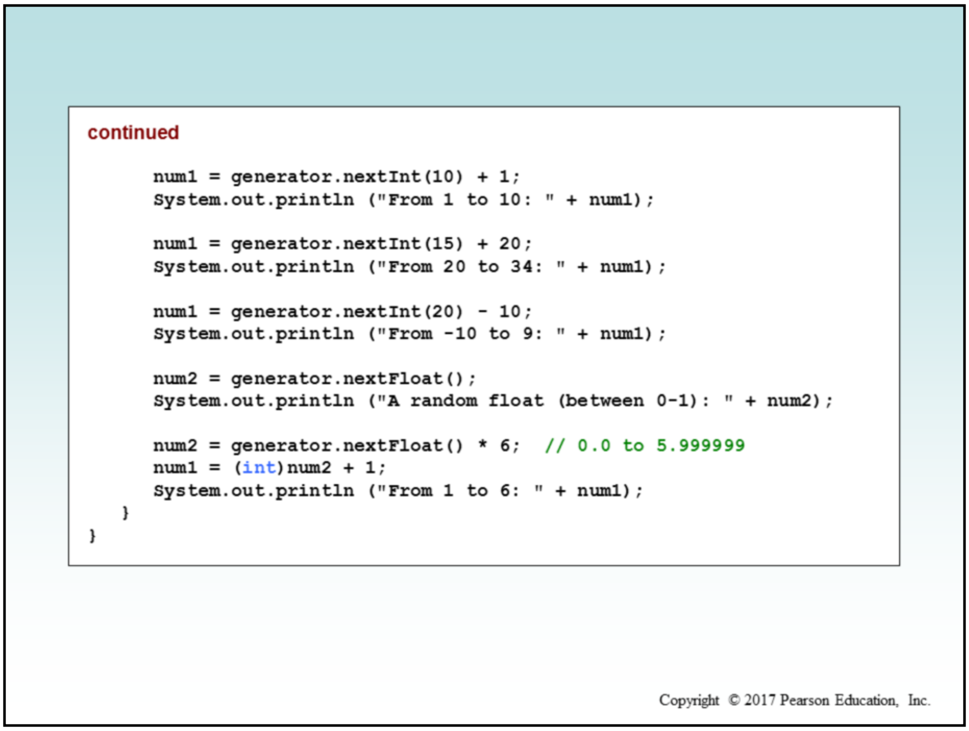
-Pseudorandom refers to the fact that the numbers are actually **computed** to appear as random (as opposed to **selected** randomly)



```
//********************************************************************
//  RandomNumbers.java       Author: Lewis/Loftus
//
//  Demonstrates the creation of pseudo-random numbers using the
//  Random class.
//********************************************************************

import java.util.Random;

public class RandomNumbers
{
   //-----------------------------------------------------------------
   //  Generates random numbers in various ranges.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      Random generator = new Random();
      int num1;
      float num2;

      num1 = generator.nextInt();
      System.out.println ("A random integer: " + num1);

      num1 = generator.nextInt(10);
      System.out.println ("From 0 to 9: " + num1);
```

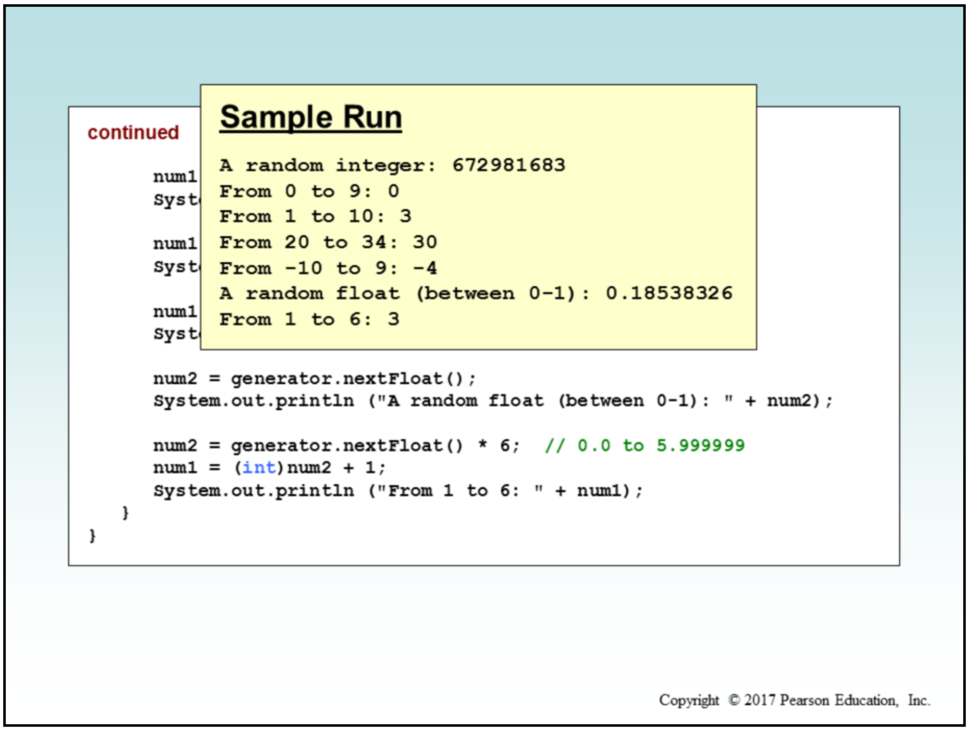
continued

-Note the import statement specifying the Random class in the java.util package



continued

```
      num1 = generator.nextInt(10) + 1;
      System.out.println ("From 1 to 10: " + num1);

      num1 = generator.nextInt(15) + 20;
      System.out.println ("From 20 to 34: " + num1);

      num1 = generator.nextInt(20) - 10;
      System.out.println ("From -10 to 9: " + num1);

      num2 = generator.nextFloat();
      System.out.println ("A random float (between 0-1): " + num2);

      num2 = generator.nextFloat() * 6;  // 0.0 to 5.999999
      num1 = (int)num2 + 1;
      System.out.println ("From 1 to 6: " + num1);
   }
}
```

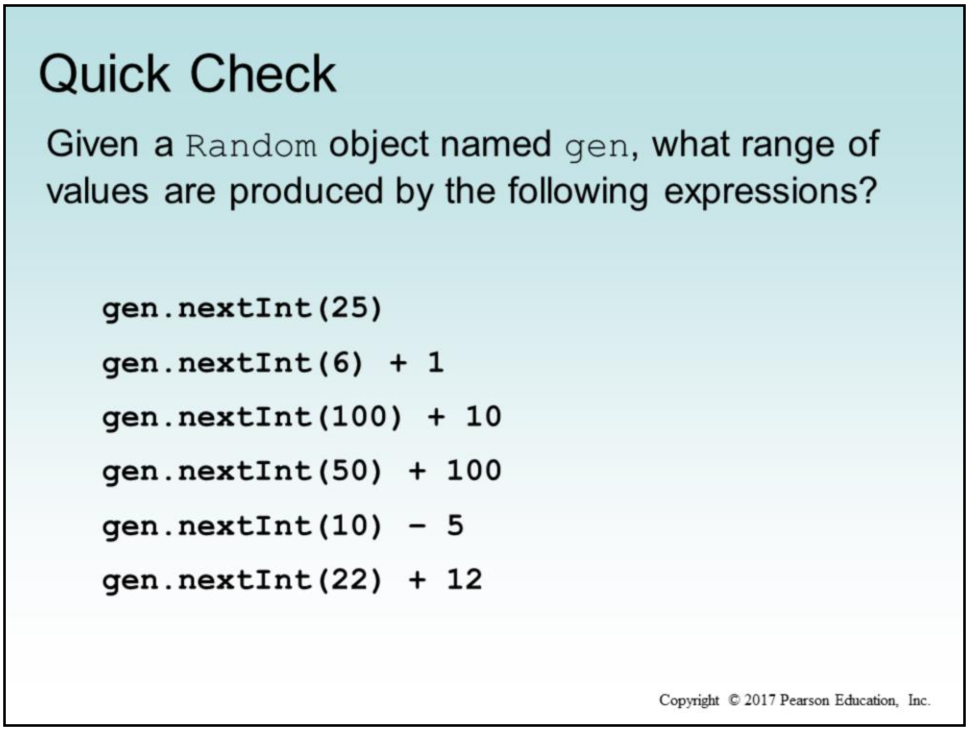


continued

```
      num1
      Syst

      num1
      Syst

      num1
      Syst

      num2 = generator.nextFloat();
      System.out.println ("A random float (between 0-1): " + num2);

      num2 = generator.nextFloat() * 6;  // 0.0 to 5.999999
      num1 = (int)num2 + 1;
      System.out.println ("From 1 to 6: " + num1);
   }
}
```

Sample Run

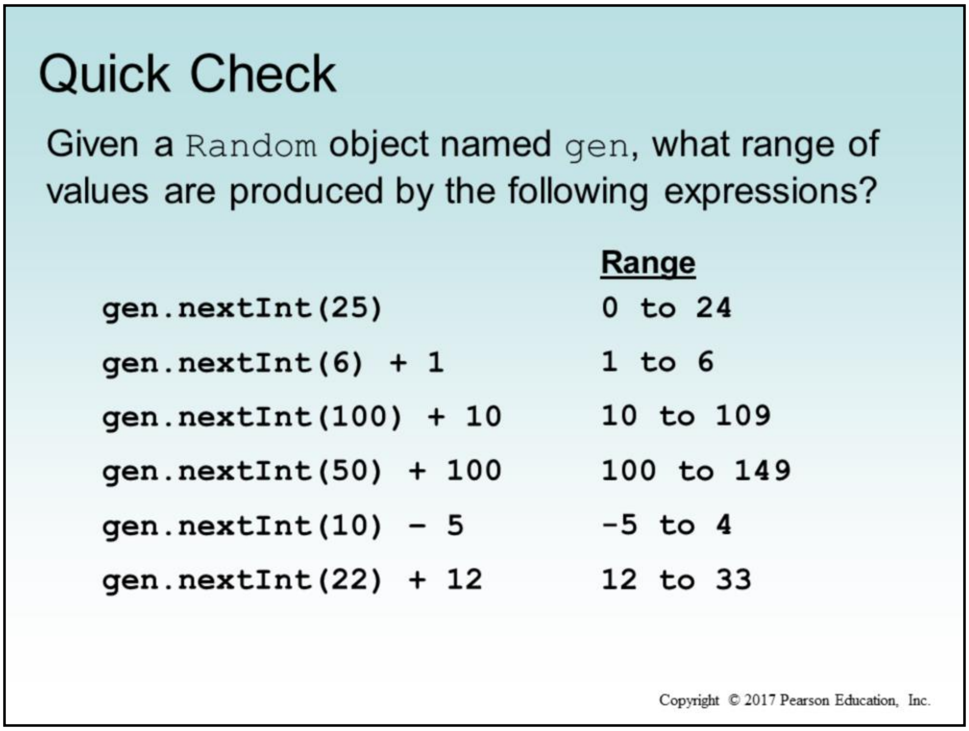
```
A random integer: 672981683
From 0 to 9: 0
From 1 to 10: 3
From 20 to 34: 30
From -10 to 9: -4
A random float (between 0-1): 0.18538326
From 1 to 6: 3
```



Quick Check

Given a `Random` object named `gen`, what range of values are produced by the following expressions?

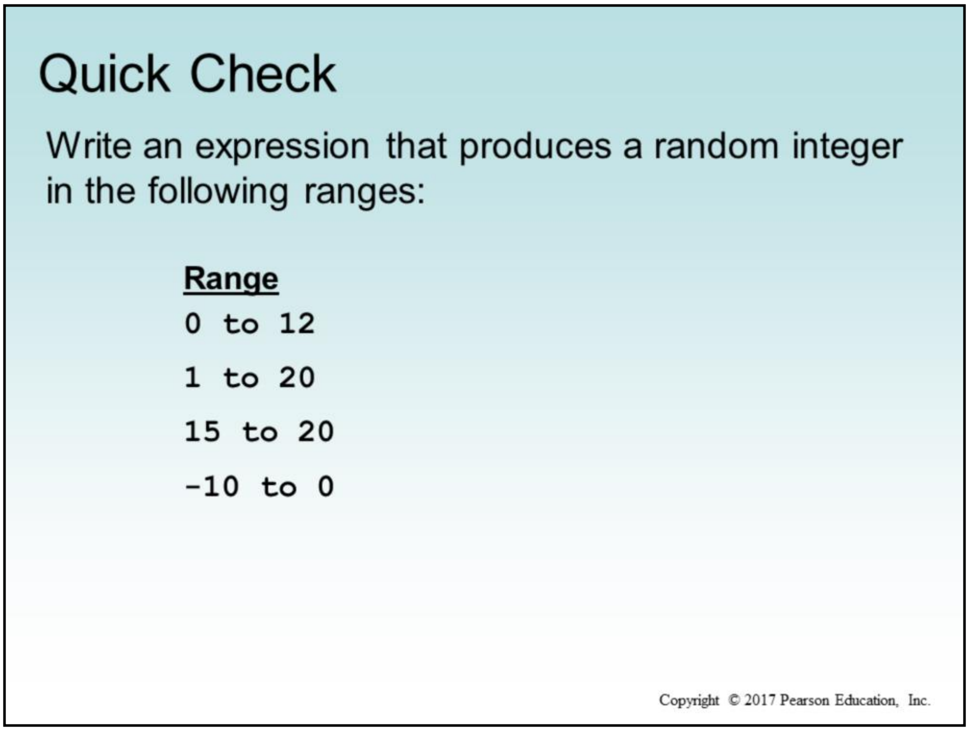
```
gen.nextInt(25)
gen.nextInt(6) + 1
gen.nextInt(100) + 10
gen.nextInt(50) + 100
gen.nextInt(10) – 5
gen.nextInt(22) + 12
```



Quick Check

Given a `Random` object named `gen`, what range of values are produced by the following expressions?

	Range
`gen.nextInt(25)`	0 to 24
`gen.nextInt(6) + 1`	1 to 6
`gen.nextInt(100) + 10`	10 to 109
`gen.nextInt(50) + 100`	100 to 149
`gen.nextInt(10) – 5`	-5 to 4
`gen.nextInt(22) + 12`	12 to 33

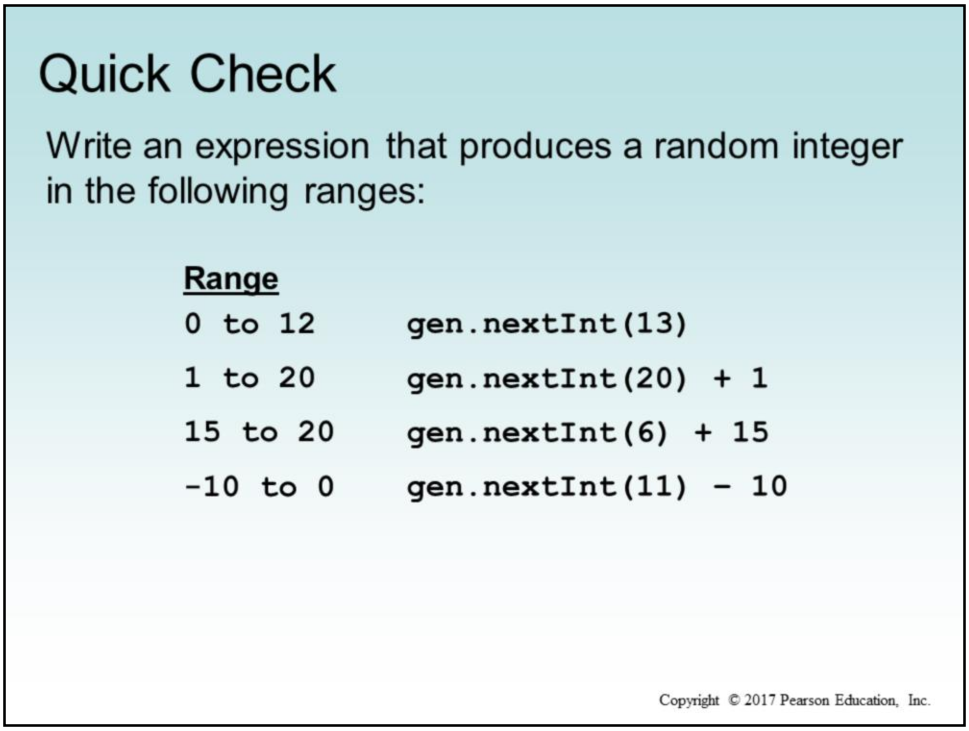


Quick Check

Write an expression that produces a random integer in the following ranges:

Range

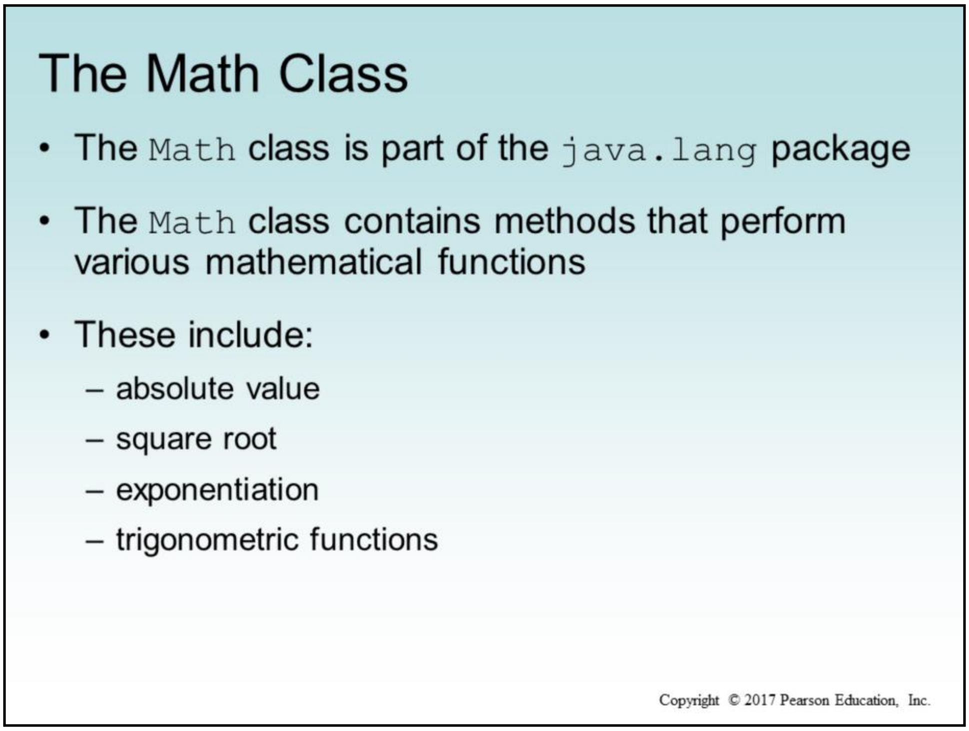
```
0 to 12
1 to 20
15 to 20
-10 to 0
```



Quick Check

Write an expression that produces a random integer in the following ranges:

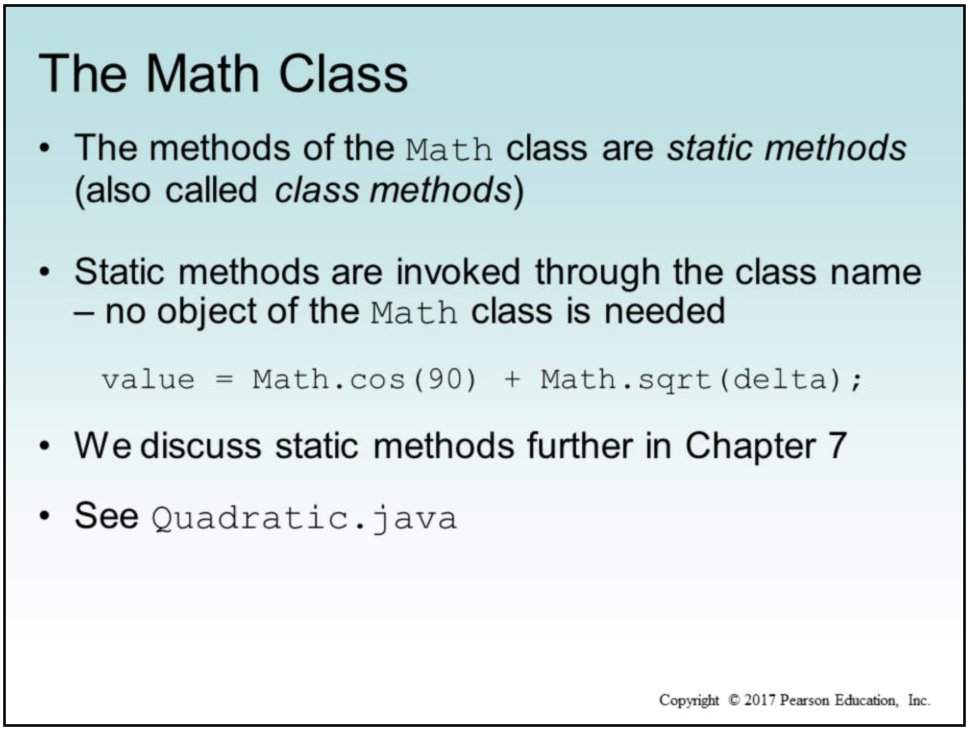
Range	
`0 to 12`	`gen.nextInt(13)`
`1 to 20`	`gen.nextInt(20) + 1`
`15 to 20`	`gen.nextInt(6) + 15`
`-10 to 0`	`gen.nextInt(11) – 10`



The Math Class

- The `Math` class is part of the `java.lang` package
- The `Math` class contains methods that perform various mathematical functions
- These include:
  - absolute value
  - square root
  - exponentiation
  - trigonometric functions

-Since Math class is in java.lang, no need to import



The Math Class

- The methods of the `Math` class are *static methods* (also called *class methods*)
- Static methods are invoked through the class name – no object of the `Math` class is needed

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

- We discuss static methods further in Chapter 7
- See `Quadratic.java`

-Static methods do not require us to create an object of the Math class

-Instead of:

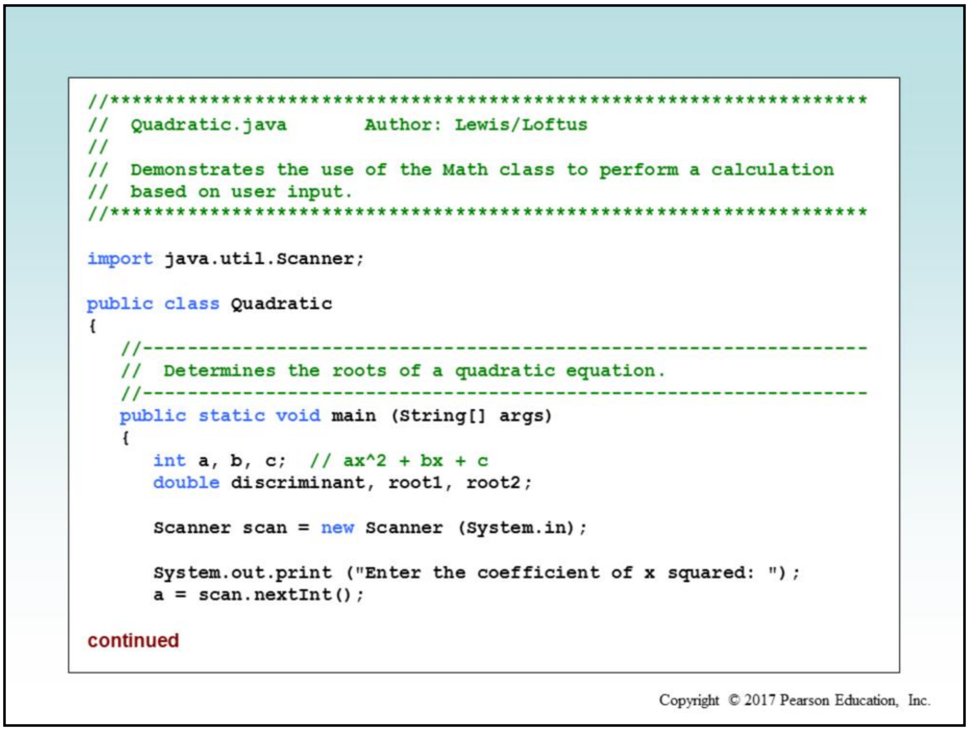
```
Math m = new Math();
double val = m.cos(90);
```

-We can simply use:

```
double val = Math.cos(90);
```

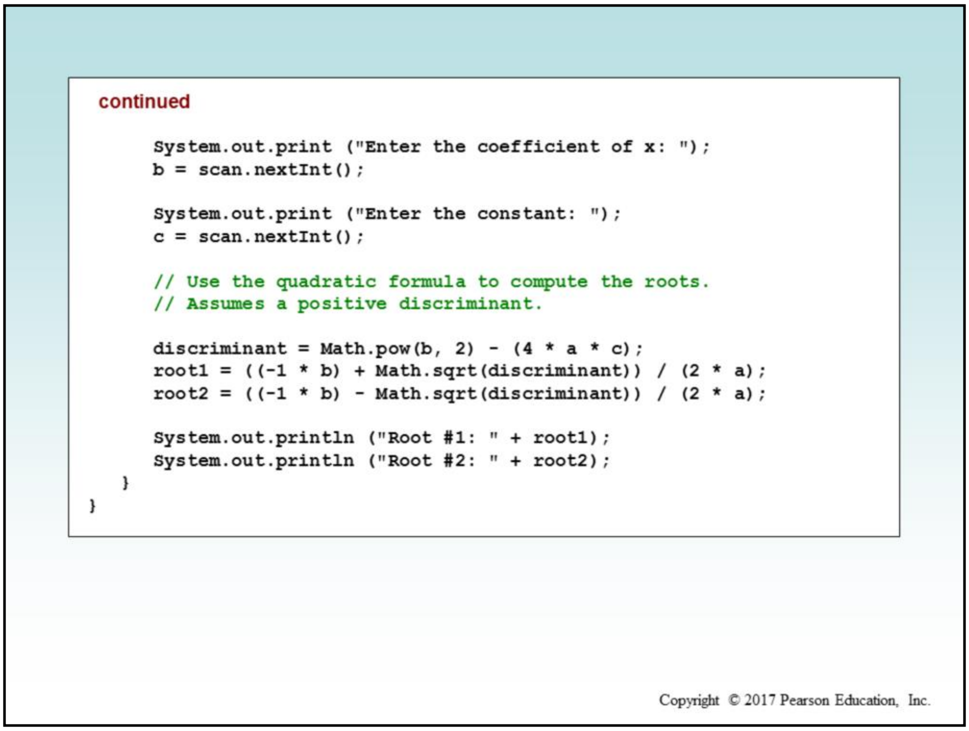
-Note in the expression, we can use the value returned from the cos and sqrt methods directly

-We do not need to assign them to a variable, we can use them directly in expressions



```
//*********************************************************************
//  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

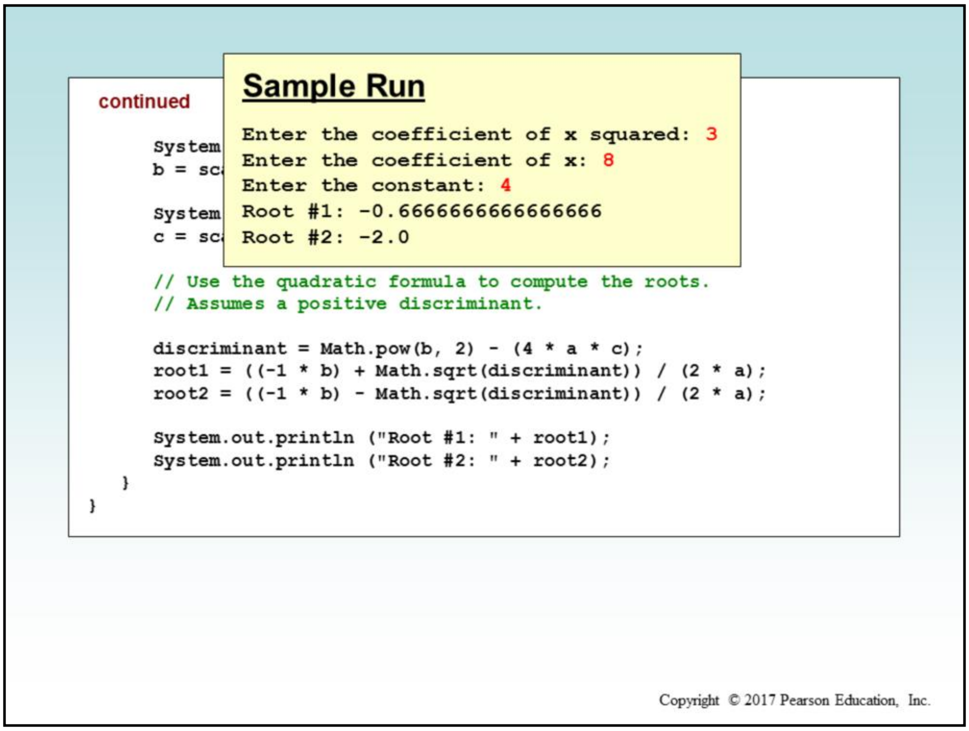
```
      System.out.print ("Enter the coefficient of x: ");
      b = scan.nextInt();

      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);
   }
}
```

-Note that some parts of the expression are repeated (e.g. Math.sqrt(discriminant))

-If we plan on repeating calculations, we can perform them one time and store in a variable

-This can help performance by reducing the number of calculations:

```
double val = Math.sqrt(discriminant);
double a2 = (2*a);
double b1 = (-1*b);
root1 = (b1 + val)/a2;
root2  = (b1 – val)/a2;
```

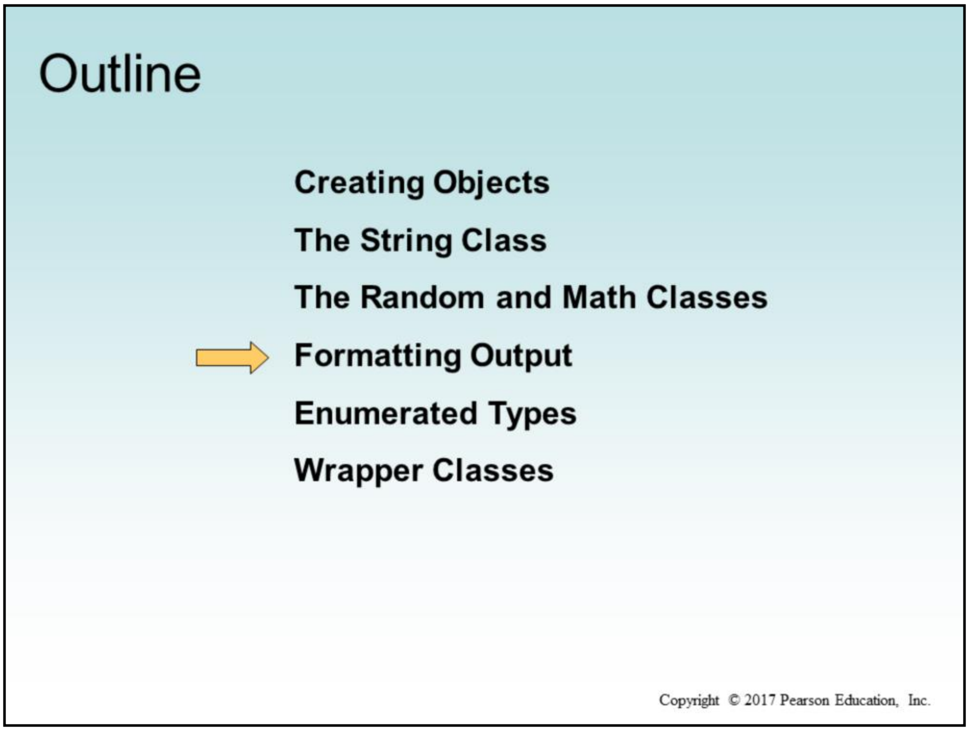


continued

Sample Run

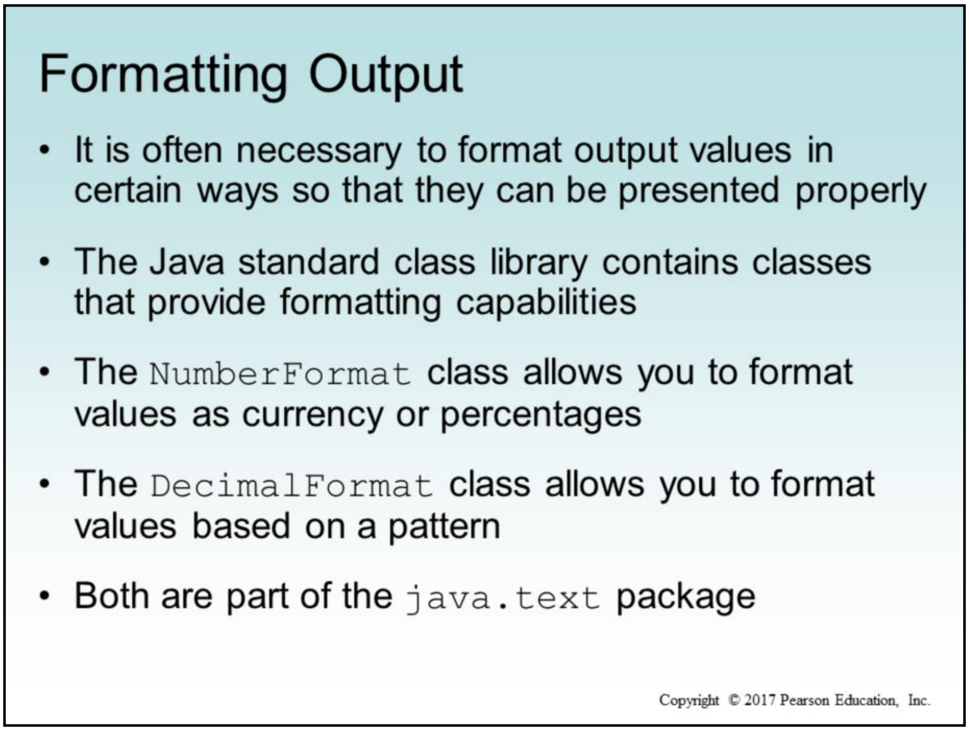
```
Enter the coefficient of x squared: 3
Enter the coefficient of x: 8
Enter the constant: 4
Root #1: -0.6666666666666666
Root #2: -2.0
```

```
      System
      b = sc

      System
      c = sc

      // 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);
   }
}
```



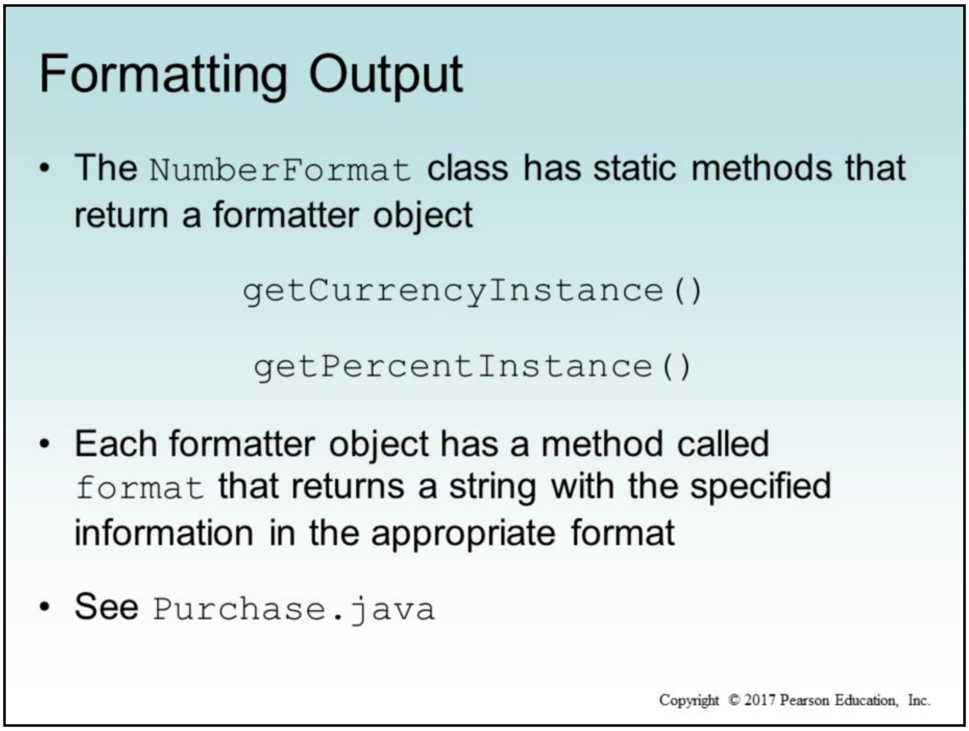
Outline

- Creating Objects
- The String Class
- The Random and Math Classes
- ➡ Formatting Output
- Enumerated Types
- Wrapper Classes



Formatting Output

- It is often necessary to format output values in certain ways so that they can be presented properly
- The Java standard class library contains classes that provide formatting capabilities
- 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

  `getCurrencyInstance()`

  `getPercentInstance()`

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

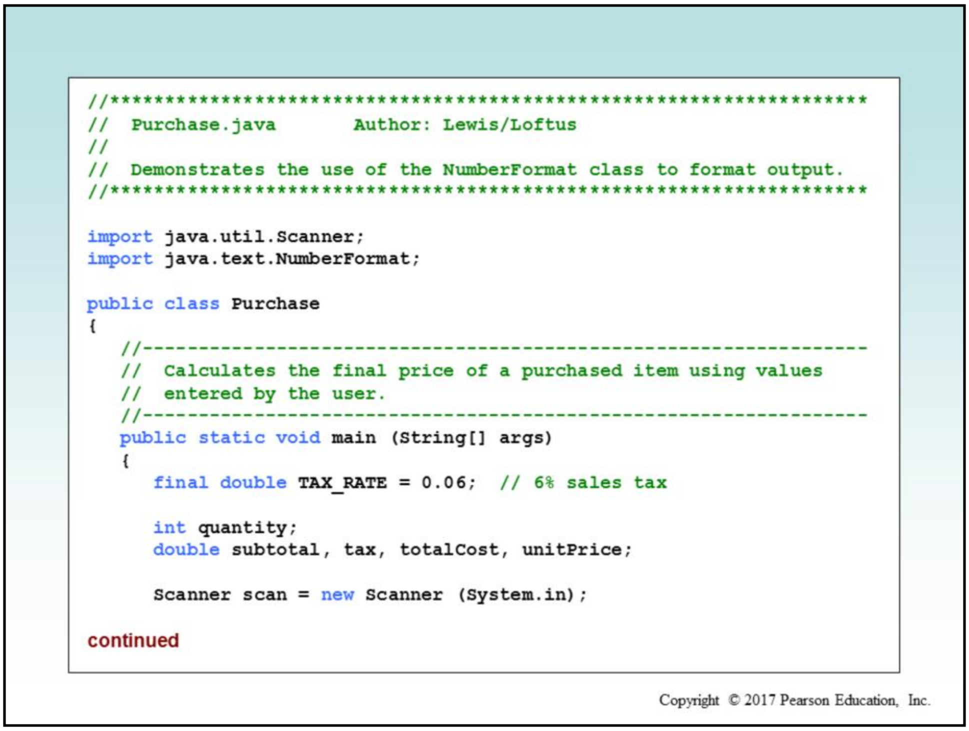
-Note the uniqueness of how to use this class

-With this class, we **don’t** use the new operator

-Instead, one is returned for us to use from the **static** methods:

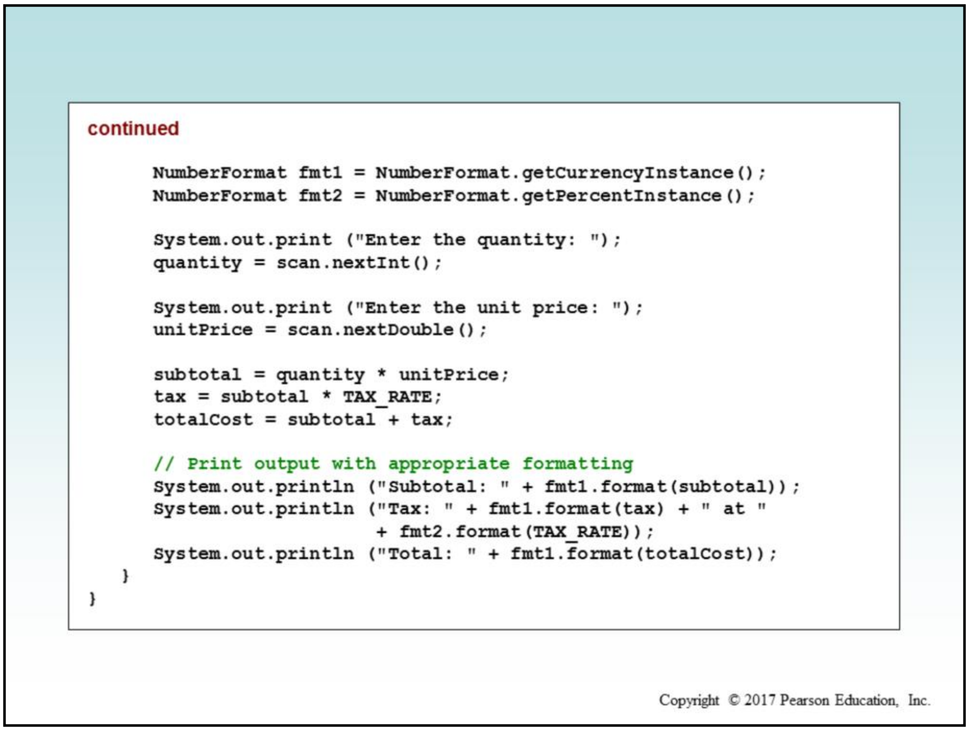
getCurrencyInstance(), getPercentInstance()

-We then call the format method from this return object to return a formatted string



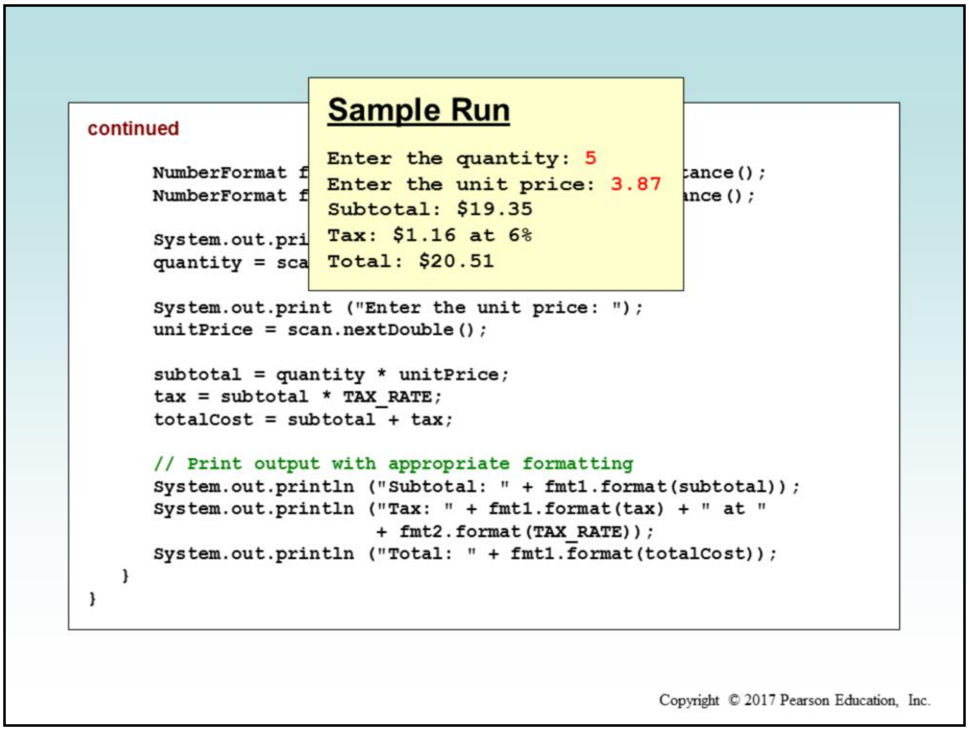
```
//********************************************************************
//  Purchase.java       Author: Lewis/Loftus
//
//  Demonstrates the use of the NumberFormat class to format output.
//********************************************************************

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 fmt1 = NumberFormat.getCurrencyInstance();
      NumberFormat fmt2 = 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: " + fmt1.format(subtotal));
      System.out.println ("Tax: " + fmt1.format(tax) + " at "
                          + fmt2.format(TAX_RATE));
      System.out.println ("Total: " + fmt1.format(totalCost));
   }
}
```

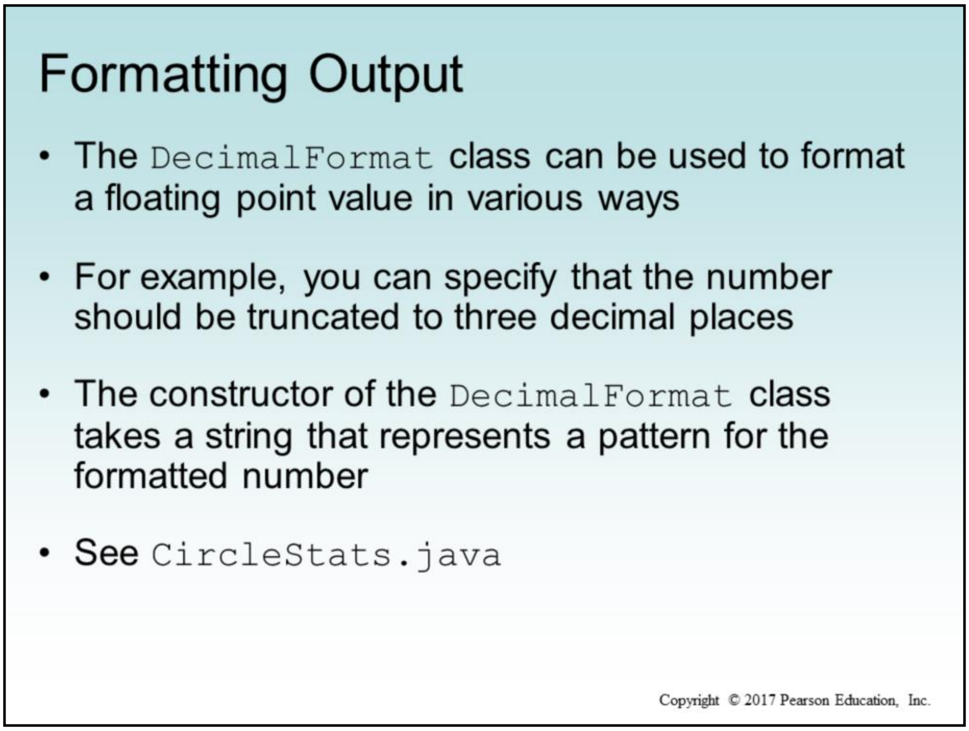


continued

```
      NumberFormat f                                 tance();
      NumberFormat f                                 ance();

      System.out.pri
      quantity = sca

      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: " + fmt1.format(subtotal));
      System.out.println ("Tax: " + fmt1.format(tax) + " at "
                          + fmt2.format(TAX_RATE));
      System.out.println ("Total: " + fmt1.format(totalCost));
   }
}
```

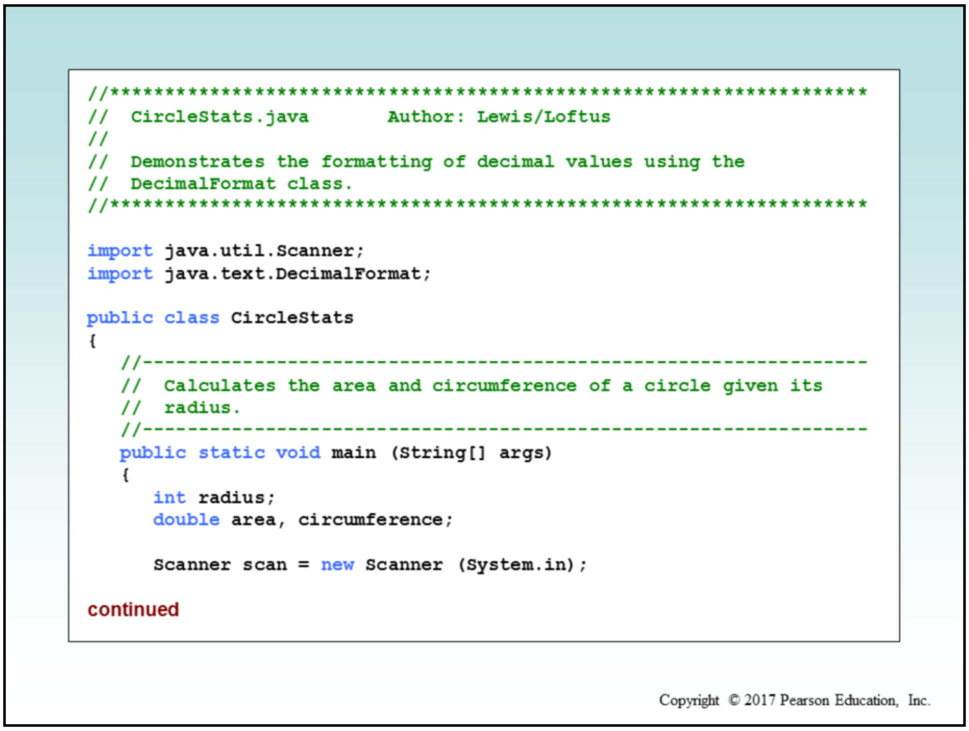
Sample Run

```
Enter the quantity: 5
Enter the unit price: 3.87
Subtotal: $19.35
Tax: $1.16 at 6%
Total: $20.51
```



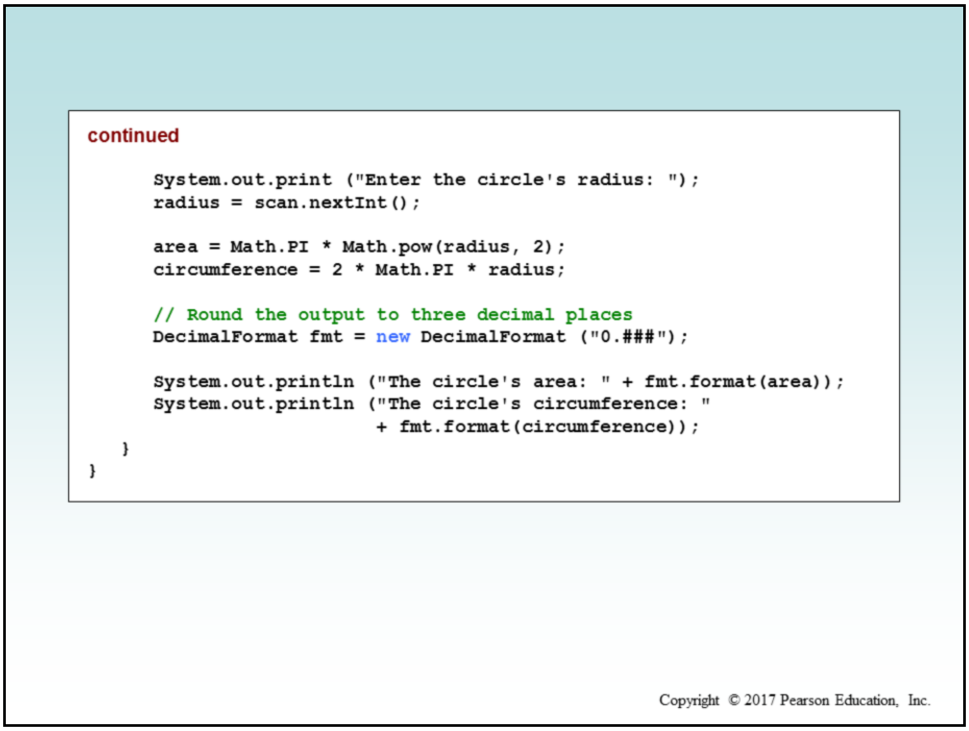
Formatting Output

- The `DecimalFormat` class can be used to format a floating point value in various ways
- For example, you can specify that the number should be truncated to three decimal places
- The constructor of the `DecimalFormat` class takes a string that represents a pattern for the formatted number
- See `CircleStats.java`



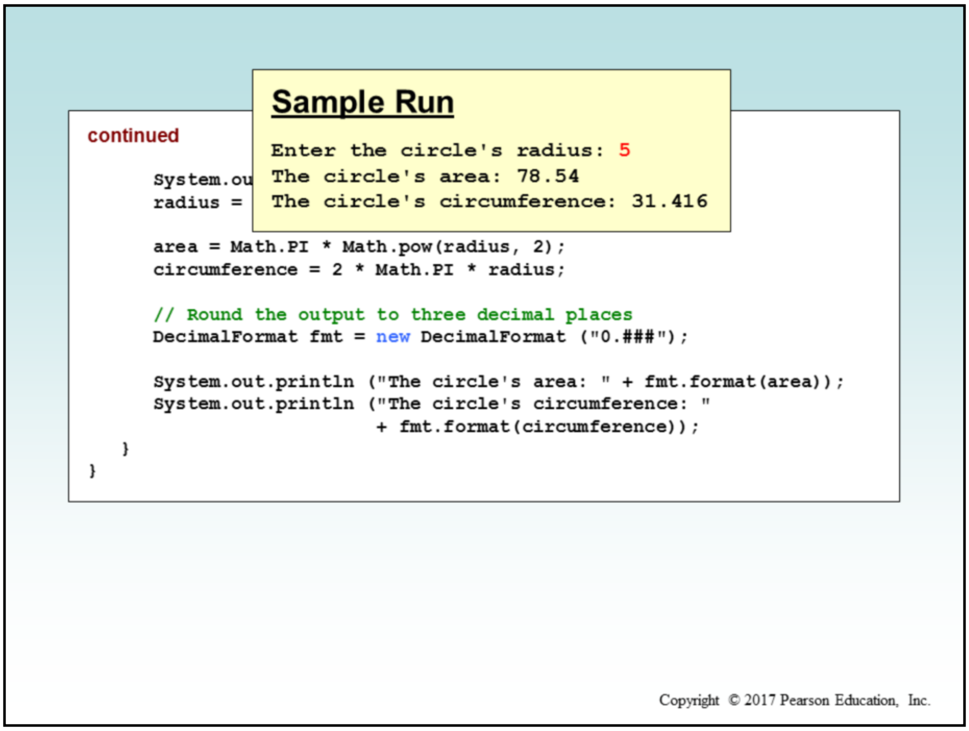
```
//********************************************************************
//  CircleStats.java       Author: Lewis/Loftus
//
//  Demonstrates the formatting of decimal values using the
//  DecimalFormat class.
//********************************************************************

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));
   }
}
```

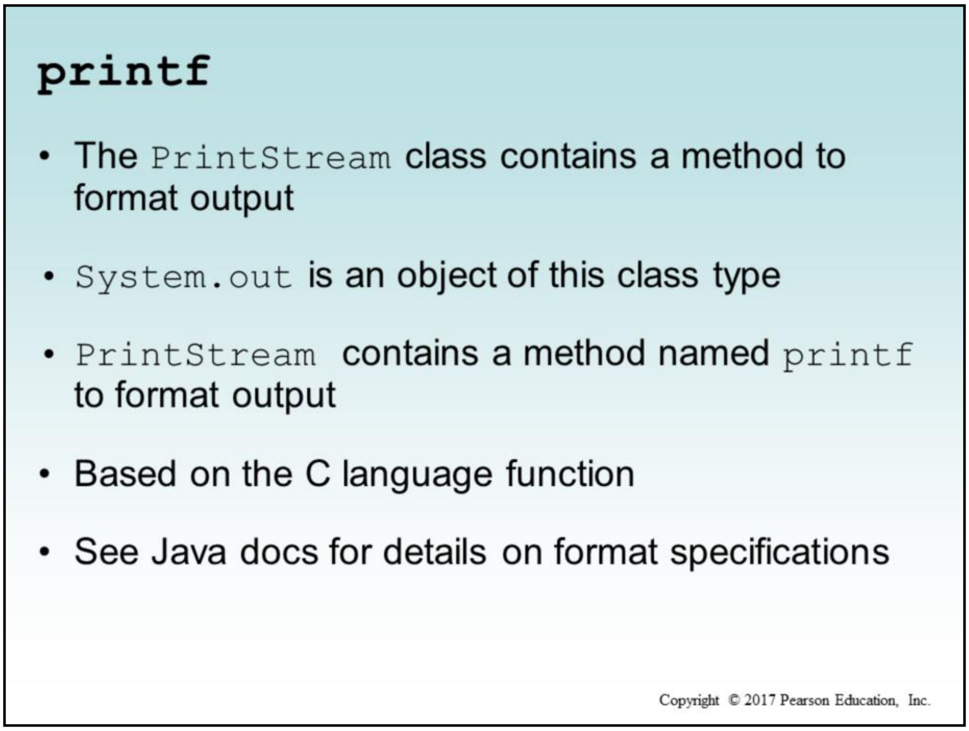


continued

```
      System.ou
      radius =

      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));
   }
}
```

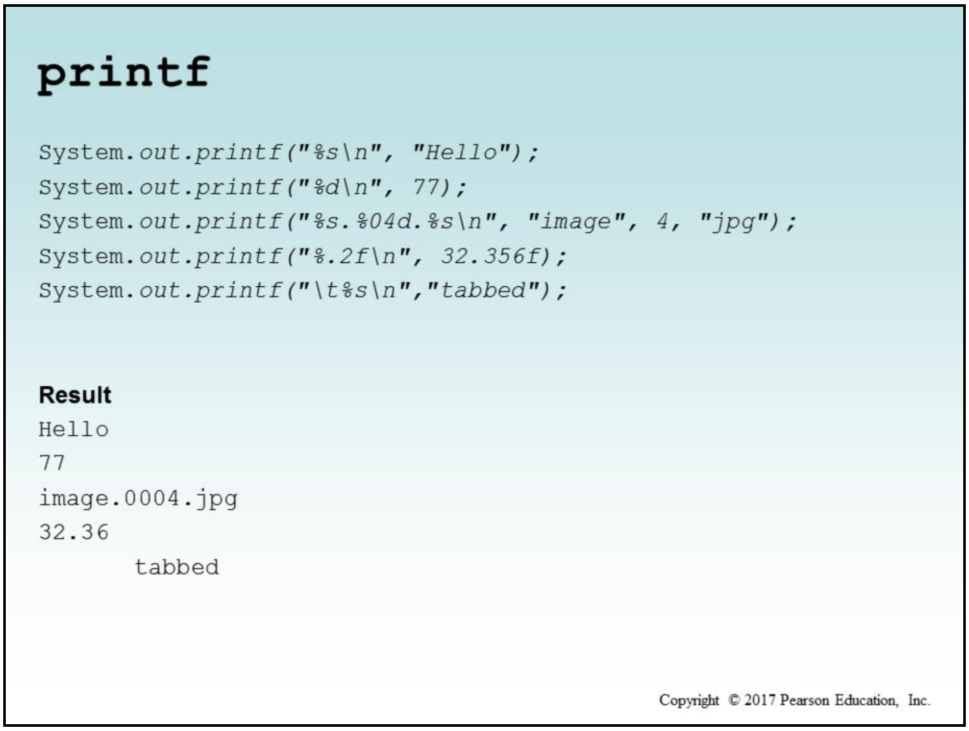
Sample Run

```
Enter the circle's radius: 5
The circle's area: 78.54
The circle's circumference: 31.416
```



`printf`

- The `PrintStream` class contains a method to format output
- `System.out` is an object of this class type
- `PrintStream` contains a method named `printf` to format output
- Based on the C language function
- See Java docs for details on format specifications



printf

```
System.out.printf("%s\n", "Hello");
System.out.printf("%d\n", 77);
System.out.printf("%s.%04d.%s\n", "image", 4, "jpg");
System.out.printf("%.2f\n", 32.356f);
System.out.printf("\t%s\n","tabbed");
```

Result

```
Hello
77
image.0004.jpg
32.36
	tabbed
```