Using Classes and Objects

CSC 2014 – Java Bootcamp

Dr. Mary-Angela Papalaskari
Department of Computing Sciences
Villanova University

Some slides in this presentation are adapted from the slides accompanying Java Software Solutions by Lewis & Loftus
The Java class library or Java API (Application Programming Interface)
# Packages

- For purposes of accessing them, classes in the Java API are organized into *packages*.

<table>
<thead>
<tr>
<th>Package</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.lang</td>
<td>General support</td>
</tr>
<tr>
<td>java.applet</td>
<td>Creating applets for the web</td>
</tr>
<tr>
<td>java.awt</td>
<td>Graphics and graphical user interfaces</td>
</tr>
<tr>
<td>javax.swing</td>
<td>Additional graphics capabilities</td>
</tr>
<tr>
<td>java.net</td>
<td>Network communication</td>
</tr>
<tr>
<td>java.util</td>
<td>Utilities</td>
</tr>
<tr>
<td>javax.xml.parsers</td>
<td>XML document processing</td>
</tr>
</tbody>
</table>

Imported automatically, includes String and Math classes.
The Math Class

- The Math class contains methods that perform various mathematical functions
- These include:
  - absolute value
  - square root
  - exponentiation
  - trigonometric functions

\[ \text{value} = \text{Math.cos}(90) + \text{Math.sqrt}(\text{delta}); \]
The Math Class

• The **Math class is part of the java.lang package**
  
  – No need to import anything!
  
  – The Math class methods are **static**
  
  – Static methods are invoked through the **class name**

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

See Quadratic.java
Some methods from the Math class

<table>
<thead>
<tr>
<th>Method Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public class Math</code></td>
</tr>
<tr>
<td><code>double abs(double a)</code></td>
</tr>
<tr>
<td><code>absolute value of a</code></td>
</tr>
<tr>
<td><code>double max(double a, double b)</code></td>
</tr>
<tr>
<td><code>maximum of a and b</code></td>
</tr>
<tr>
<td><code>double min(double a, double b)</code></td>
</tr>
<tr>
<td><code>minimum of a and b</code></td>
</tr>
<tr>
<td><code>double sin(double theta)</code></td>
</tr>
<tr>
<td><code>sine function</code></td>
</tr>
<tr>
<td><code>double cos(double theta)</code></td>
</tr>
<tr>
<td><code>cosine function</code></td>
</tr>
<tr>
<td><code>double tan(double theta)</code></td>
</tr>
<tr>
<td><code>tangent function</code></td>
</tr>
<tr>
<td><code>Degrees in radians. Use toDegrees() and toRadians() to convert.</code></td>
</tr>
<tr>
<td><code>double exp(double a)</code></td>
</tr>
<tr>
<td><code>exponential (e^a)</code></td>
</tr>
<tr>
<td><code>double log(double a)</code></td>
</tr>
<tr>
<td><code>natural log (log_e a, or ln a)</code></td>
</tr>
<tr>
<td><code>double pow(double a, double b)</code></td>
</tr>
<tr>
<td><code>raise a to the bth power (a^b)</code></td>
</tr>
<tr>
<td><code>long round(double a)</code></td>
</tr>
<tr>
<td><code>round to the nearest integer</code></td>
</tr>
<tr>
<td><code>double random()</code></td>
</tr>
<tr>
<td><code>random number in [0, 1)</code></td>
</tr>
<tr>
<td><code>double sqrt(double a)</code></td>
</tr>
<tr>
<td><code>square root of a</code></td>
</tr>
<tr>
<td><code>double E</code></td>
</tr>
<tr>
<td><code>value of e (constant)</code></td>
</tr>
<tr>
<td><code>double PI</code></td>
</tr>
<tr>
<td><code>value of π (constant)</code></td>
</tr>
</tbody>
</table>

Also defined for int, long, and float

Inverse functions also available: asin(), acos(), and atan

You can discard your calculator now (please).
The Random Class

- **Part of the java.util package, so import it**
  ```java
  import java.util.Random;
  ```

- **Create a Random object named gen:**
  ```java
  Random gen = new Random();
  ```

- **Use Random method `nextInt()` to generate a random number:**
  ```java
  int a = gen.nextInt(4);
  // integer in range [0, 1, 2, 3]
  ```
What is a random number?

“Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.”
- John Von Neumann

“God does not play dice.”
- Albert Einstein

The Random class provides methods that generate pseudorandom numbers
Example:

- Get some snow into the Snowman Applet!

```java
int flake = 1;
while (flake <= 1000)
{
    int x = gen.nextInt(300);
    int y = gen.nextInt(225);
    page.fillOval(x, y, 2, 2);
    flake++;
}

* can you get the snowflakes to also vary in size (say, 2-4 pixels)?
```
Example: Using Random methods

```java
Random gen = new Random();
int a = gen.nextInt(4);
    // integer in range [0,1,2,3]
float b = gen.nextFloat();
    // float in range [0,1), eg: 0.4589
int c = gen.nextInt(4) + 1;
    // int in range [1,2,3,4]
int d = gen.nextInt();
    // int in range [-2147483648 ... 2147483647]
```

List of some Random methods: page 126

See RandomNumbers.java
How about a random color?

Color mystery = new Color(__, __, __);
Example: counting “snake eyes”

// Roll two dice 100,000 times and count how many
// times you roll snake eyes, i.e., two 1’s.

Random gen = new Random();
int trial = 0, count = 0;

while (trial < 100000)
{
    int die1 = gen.nextInt(6) + 1;
    int die2 = gen.nextInt(6) + 1;
    if (die1 == 1 && die2 == 1)
        count++; // snake eyes
    trial++;
}
System.out.println ("Probablility of snake eyes = " +
                (double)count/100000);
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(50) + 100`
- `gen.nextInt(10) - 5`
- `(int)(gen.nextFloat() * 10 + 1)`
- `(int)(Math.random() * 10 + 1)`

alternative way to generate pseudorandom number in the range 0…1
Quick Check

Given a Random object named gen, write an expression that produces a random integer in the following ranges:

<table>
<thead>
<tr>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 to 12</td>
</tr>
<tr>
<td>1 to 20</td>
</tr>
<tr>
<td>15 to 20</td>
</tr>
<tr>
<td>-10 to 0</td>
</tr>
</tbody>
</table>
The Strings Class

- Strings are objects defined by the `String` class

```
"This is a string literal."
"123 Main Street"
"X"
```

- the `String` class has many methods that can be used to process text. Examples:
  - finding the length of a string
  - finding the char at a certain position of a string
  - producing an all-caps version of a string
Invoking String Methods

As with other kinds of objects, we use the dot operator to invoke a String’s methods:

```java
String name = "Betsy";

int numOfCharsInName = name.length();
```

`length()` is one of the methods of `String` objects (defined in the `String` class).
More String Methods

- List of some String methods: see textbook, page 119

```java
String name = "Betsy";

char initial = name.charAt(0);

String newName = name.replace('s', 't');

String capsName = name.toUpperCase();

int comp = name.compareTo(newName);
```

```
compareTo(b)
```

Compare strings, alphabetically:
- a>b ➞ positive
- a=b ➞ zero
- a<b ➞ negative

See also textbook example StringMutation.java
Example: Palindrome tester

• **Problem:** Input a string, determine whether it is a palindrome, i.e.:
  – first char is the same as last char
  – 2nd char is the same as 2nd last char
  – and so on...

• How to express this as an algorithm?
• How to implement it?
```java
System.out.println("Enter a potential palindrome:");
str = scan.nextLine();

left = 0;
right = str.length() - 1;

while (str.charAt(left) == str.charAt(right) && left < right)
{
    left++;
    right--;
}

if (left < right)
    System.out.println("NOT a palindrome");
else
    System.out.println("palindrome");
```
System.out.println("Enter a potential palindrome:");
str = scan.nextLine();

left = 0;
right = str.length() - 1;

while (str.charAt(left) == str.charAt(right) && left < right) {
    left++;
    right--;
}

if (left < right)
    System.out.println("NOT a palindrome");
else
    System.out.println("palindrome");
Declaring Variables, revisited

• Examples of variable declarations:

```java
int count = 0;
double mpg;
String title;
Graphics page;
Color aquamarine;
Scanner scan;
```

• A class name can be used as a type to declare an *object reference variable*

• The object itself must be created separately
Creating Objects

- We have already seen something like this:

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

- The `new` operator calls the Scanner constructor, which is a special method that sets up the object.

- Variable refers to a `Scanner` object.

- Constructing a new object is called `instantiation`.

- An `instance` of the Scanner class.
Creating Objects

• Another example:

```java
String title = new String("Java Software Solutions");
```

The `new` operator calls the String constructor, which is a special method that sets up the object.

Variable refers to a **String** object

an *instance* of the String class

Constructing a new object is called **instantiation**
The String Class is SPECIAL!

- Exception to the use of `new` operator: Because strings are so common, we don't have to use the `new` operator to create a `String` object

```java
String title = new String("Java Software Solutions");
```

- This is special syntax that works **only** for strings
import javax.swing.JOptionPane;

public class EvenOdd {
    public static void main(String[] args) {
        String numStr, result;
        int num, again;
        do {
            numStr = JOptionPane.showInputDialog("Enter an integer: ");
            num = Integer.parseInt(numStr);
            result = "That number is " + ((num % 2 == 0) ? "even" : "odd");
            JOptionPane.showMessageDialog(null, result);
            again = JOptionPane.showConfirmDialog(null, "Do Another?");
        } while (again == JOptionPane.YES_OPTION);
    }
}
import javax.swing.JOptionPane;

public class EvenOdd {
    public static void main (String[] args) {
        String numStr, result;
        int num, again;
        do {
            numStr = JOptionPane.showInputDialog("Enter an integer: ");
            num = Integer.parseInt(numStr);
            result = "That number is " + ((num%2 == 0) ? "even" : "odd");
            JOptionPane.showMessageDialog(null, result);
            again = JOptionPane.showConfirmDialog(null, "Do Another?");
        } while (again == JOptionPane.YES_OPTION);
    }
}