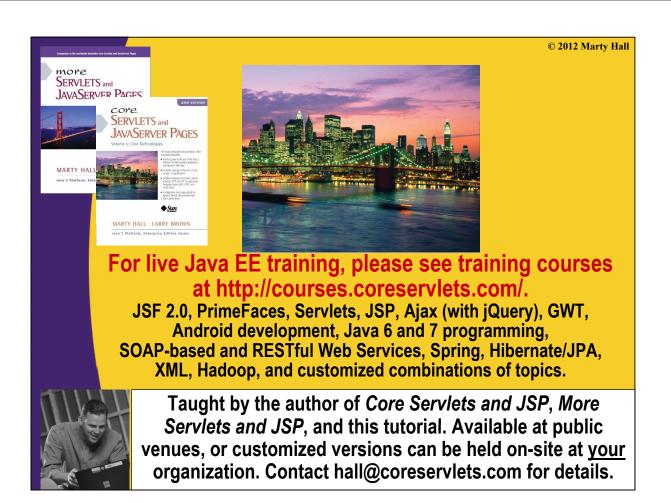
© 2012 Marty Hall

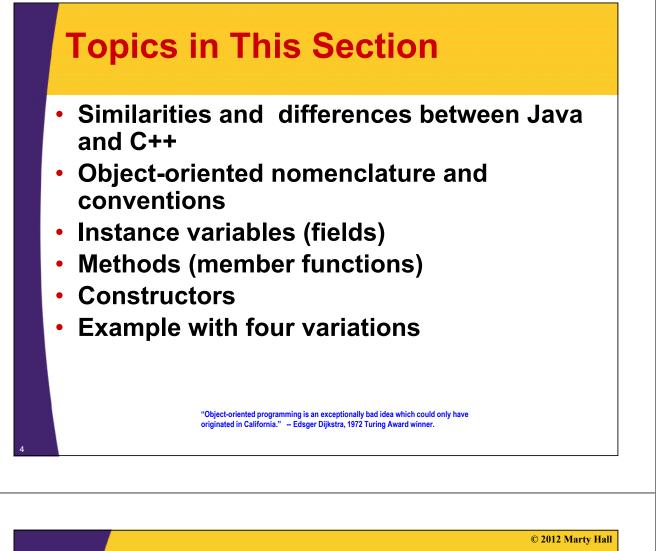


Basic Object-Oriented Programming in Java

Originals of Slides and Source Code for Examples: http://courses.coreservlets.com/Course-Materials/java.html

Customized Java EE Training: http://courses.coreservlets.com/ Java 6 or 7, JSF 2.0, PrimeFaces, Servlets, JSP, Ajax, Spring, Hibernate, RESTful Web Services, Android. Developed and taught by well-known author and developer. At public venues or onsite at *your* location.







Basics

Customized Java EE Training: http://courses.coreservlets.com/ Java 6 or 7, JSF 2.0, PrimeFaces, Servlets, JSP, Ajax, Spring, Hibernate, RESTful Web Services, Android.

Developed and taught by well-known author and developer. At public venues or onsite at your location.

Object-Oriented Programming in Java

Similarities with C++

- User-defined classes can be used like built-in types.
- Basic syntax

Differences from C++

- Methods (member functions) are the only function type
- Object is the topmost ancestor for all classes
- All methods use the run-time, not compile-time, types (i.e. all Java methods are like C++ virtual functions)
- The types of all objects are known at run-time
- All objects are allocated on the heap (always safe to return objects from methods)
- Single inheritance only

Comparisons to C#

 C# very similar to Java in OOP. For details, see http://www.harding.edu/fmccown/java1_5_csharp_comparison.html

Object-Oriented Nomenclature

"Class" means a category of things

 A class name can be used in Java as the type of a field or local variable or as the return type of a function (method)

"Object" means a particular item that belongs to a class

- Also called an "instance"

Example

String s1 = "Hello";

 Here, String is the class, and the variable s1 and the value "Hello" are objects (or "instances of the String class")

© 2012 Marty Hall



Instance Variables

Customized Java EE Training: http://courses.coreservlets.com/ Java 6 or 7, JSF 2.0, PrimeFaces, Servlets, JSP, Ajax, Spring, Hibernate, RESTful Web Services, Android. Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Overview

Definition

 Data that is stored inside an object. "Instance variables" can also be called "data members" or "fields".

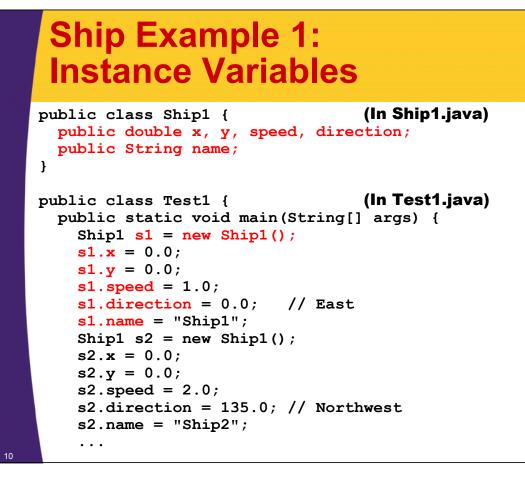
Syntax

public class MyClass {
 public SomeType field1, field2;

In any class that also has methods, it is almost always better to declare instance variables private. We will show how and why in the next tutorial section.

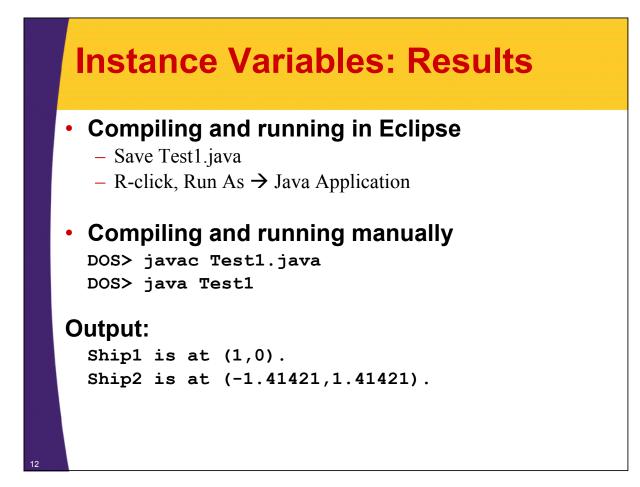
Motivation

- Lets an object have persistent values.
 - It is often said that in OOP, objects have three characteristics: state, behavior, and identity. The instance variables provide the state.



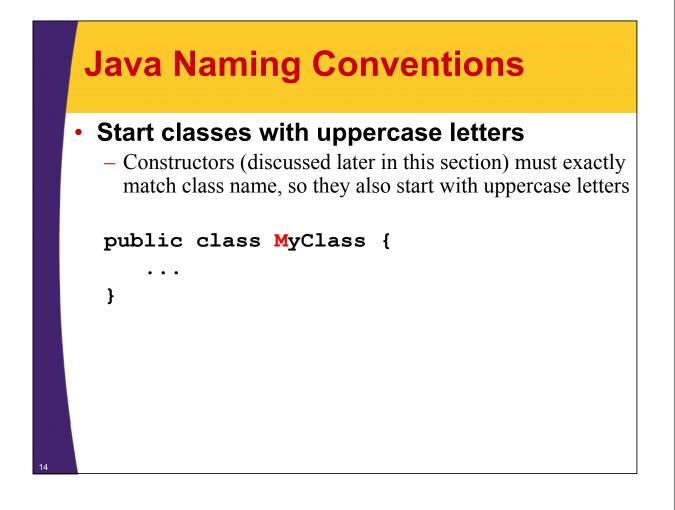
Instance Variables: Example (Continued)

}



Example 1: Major Points

- Java naming conventions
- Format of class definitions
- Creating classes with "new"
- Accessing fields with "variableName.fieldName"

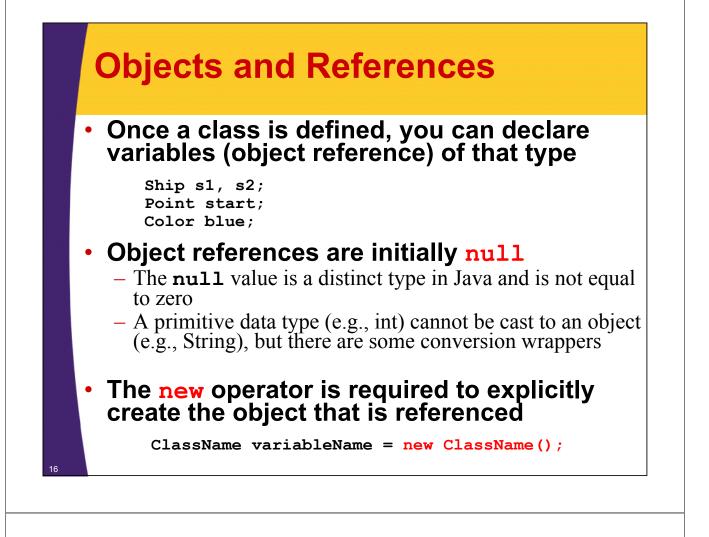


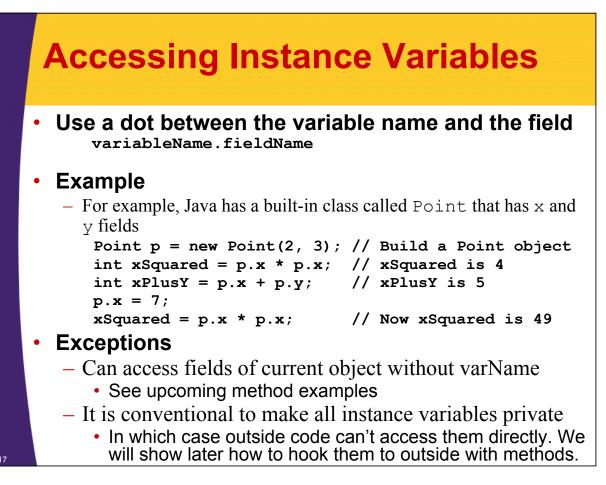
Java Naming Conventions

Start other things with lowercase letters

- Instance vars, local vars, methods, parameters to methods

```
public class MyClass {
   public String firstName, lastName;
   public String fullName() {
     String name =
     firstName + " " + lastName;
     return(name);
   }
}
```





© 2012 Marty Hall



Methods

Customized Java EE Training: http://courses.coreservlets.com/ Java 6 or 7, JSF 2.0, PrimeFaces, Servlets, JSP, Ajax, Spring, Hibernate, RESTful Web Services, Android. Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Overview

Definition

- Functions that are defined inside a class. "Methods" can also be called "member functions".

Syntax

public class MyClass {

public ReturnType myMethod(...) { ... } If you want code that uses your class to access the method, make it public. If your method is

If you want code that uses your class to access the method, make it public. If your method is called only by other methods in the same class, make it private. Make it private unless you have a specific reason to do otherwise.

Motivation

- Lets an object calculate values or do operations, usually based on its current state (instance variables).
 - It is often said that in OOP, objects have three characteristics: state, behavior, and identity. The methods provide the behavior.

Ship Example 2: Methods

```
(In Ship2.java)
public class Ship2 {
  public double x=0.0, y=0.0, speed=1.0, direction=0.0;
  public String name = "UnnamedShip";
  private double degreesToRadians(double degrees) {
    return(degrees * Math.PI / 180.0);
  }
  public void move() {
    double angle = degreesToRadians(direction);
    x = x + speed * Math.cos(angle);
    y = y + speed * Math.sin(angle);
  }
  public void printLocation() {
    System.out.println(name + " is at ("
                       + x + "," + y + ").");
  }
```

Methods (Continued)

20

```
public class Test2 {
                                          (In Test2.java)
 public static void main(String[] args) {
    Ship2 s1 = new Ship2();
    s1.name = "Ship1";
    Ship2 s2 = new Ship2();
    s2.direction = 135.0; // Northwest
    s2.speed = 2.0;
    s2.name = "Ship2";
    s1.move();
    s2.move();
    s1.printLocation();
    s2.printLocation();
  }
}
  Compiling and Running: (R-click, Run As in Eclipse)
       javac Test2.java
       java Test2
  Output:
      Shipl is at (1,0).
       Ship2 is at (-1.41421,1.41421).
```

Example 2: Major Points

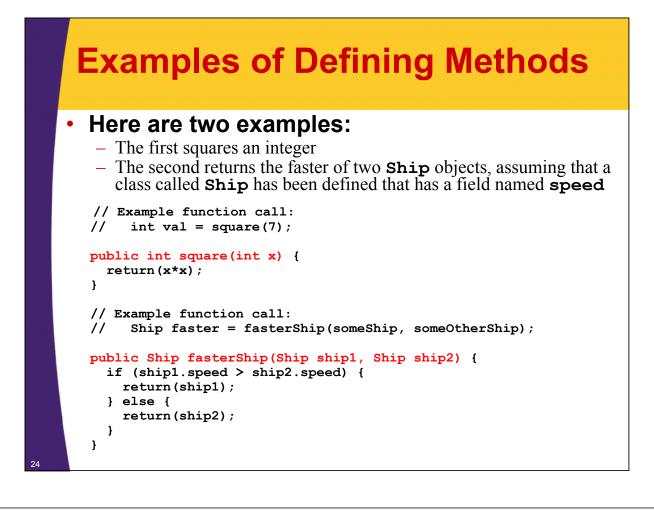
- Format of method definitions
- Methods that access local fields
- Calling methods
- Static methods
- Default values for fields
- public/private distinction

Defining Methods (Functions Inside Classes)

Basic method declaration:

Exception to this format: if you declare the return type as void

- This special syntax that means "this method isn't going to return a value – it is just going to do some side effect like printing on the screen"
- In such a case you do not need (in fact, are not permitted),
 a return statement that includes a value to be returned



Calling Methods

 The term "method" means "function associated with an object" (I.e., "member function")

- The usual way that you call a method is by doing the following:

variableName.methodName(argumentsToMethod);

 For example, the built-in String class has a method called toUpperCase that returns an uppercase variation of a String

- This method doesn't take any arguments, so you just put empty parentheses after the function (method) name.

```
String s1 = "Hello";
String s2 = s1.toUpperCase(); // s2 is now "HELLO"
```

Accessing External and Internal Methods

Accessing methods in other classes

- Get an object that refers to instance of other class
 Ship s = new Ship();
- Call method on that object
 - s.move();

Accessing instance vars in same class

- Call method directly (no variable name and dot in front)
 - move();
 - double d = degreesToRadians()
 - For local methods, you can use a variable name if you want, and Java automatically defines one called "this" for that purpose. See constructors section.

Accessing static methods

- Use ClassName.methodName(args)
 - double d = Math.cos(Math.PI/2);

Calling Methods (Continued)

There are two exceptions to requiring a variable name for a method call

- Calling a method defined inside the current class definition
 Use "methodName(area)" instead of "verblame methodName(area)
- Use "methodName(args)" instead of "varName.methodName(args)"
- Functions (methods) that are declared "static"
 - Use "ClassName.methodName(args)"

Calling a method of the current class

- You don't need the variable name and the dot
- For example, a ship class might define a method called degreeesToRadians, then, within another function in the same class definition, do this:

```
double angle = degreesToRadians(direction);
```

 No variable name and dot is required in front of degreesToRadians since it is defined in the same class as the method that is calling it

Static Methods" (vs. "instance methods") Static functions do not access any non-static methods or fields within their class and are almost like global functions in other languages. So call a static method through the class name (arguments): So you can call Math.cos(3.5) without ever having any object (instance) of the Math class Shoe the system calls main without first creating an object, static methods are the only type of methods that main can call directly (i.e. without building an object and calling the method of that object)



public/private distinction

- A declaration of private means that "outside" methods can't call it – only methods within the same class can
 - Thus, for example, the main method of the Test2 class could not have done

double x = s1.degreesToRadians(2.2);

- Attempting to do so would have resulted in an error at compile time
- Only say **public** for methods that you *want to guarantee your class will make available to users*

 You are free to change or eliminate private methods without telling users of your class

private instance variables

 In next lecture, we will see that you almost always make instance vars private and use methods to access them

Declaring Variables in Methods

Format

 When you declare a local variable inside of a method, the normal declaration syntax looks like:

Type varName = value;

The value part can be:

- A constant
- Another variable
- A function (method) call
- A constructor invocation (a special type of function prefaced by **new** that builds an object)
- Some special syntax that builds an object without
 - explicitly calling a constructor (e.g., strings)

Declaring Variables in Methods: Examples

```
int x = 3;
int y = x;
// Special syntax for building a String object
String s1 = "Hello";
// Building an object the normal way
String s2 = new String("Goodbye");
String s3 = s2;
String s4 = s3.toUpperCase(); // Result: s4 is "GOODBYE"
// Assume you defined a findFastestShip method that
// returns a Ship
Ship ship1 = new Ship();
Ship ship2 = ship1;
Ship ship3 = findFastestShip();
```

© 2012 Marty Hall



Constructors

Customized Java EE Training: http://courses.coreservlets.com/ Java 6 or 7, JSF 2.0, PrimeFaces, Servlets, JSP, Ajax, Spring, Hibernate, RESTful Web Services, Android. Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Overview

Definition

- Code that gets executed when "new" is called

Syntax

 "Method" that exactly matches the class name and has no return type (not even void).

```
public class MyClass {
```

```
public MyClass(...) { ... }
```

}

Motivation

- Lets you build an instance of the class, and assign values to instance variables, all in one fell swoop
- Lets you enforce that all instances have certain properties
- Lets you run side effects when class is instantiated

Example: No User-Defined Constructor

Person

```
public class Person1 {
   public String firstName, lastName;
}
```

PersonTest

```
public class Person1Test {
   public static void main(String[] args) {
     Person1 p = new Person1();
     p.firstName = "Larry";
     p.lastName = "Ellison";
     // doSomethingWith(p);
   }
}
```

Example: User-Defined Constructor

Person

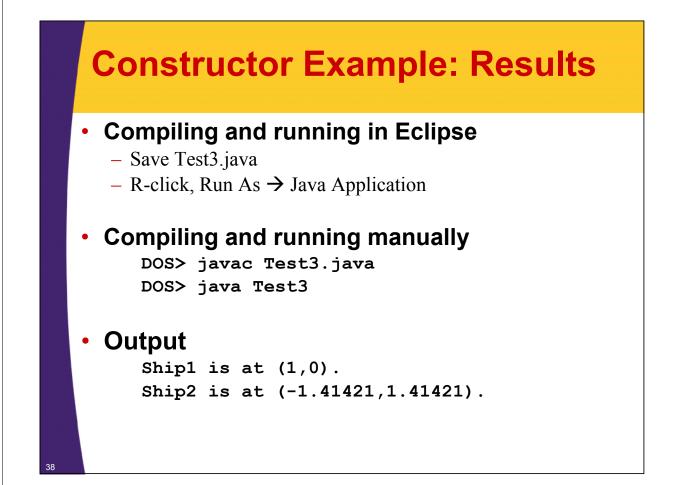
```
public class Person2 {
    public String firstName, lastName;
    public Person2(String initialFirstName,
                        String initialLastName) {
       firstName = initialFirstName;
       lastName = initialLastName;
     }
  }
PersonTest
  public class Person2Test {
    public static void main(String[] args) {
       Person2 p = new Person2("Larry", "Page");
       // doSomethingWith(p);
                                                    It took one line of code to make a properly
     }
                                                    constructed person. It would not be possible for a
                                                    programmer to build a person and forget to assign a
  }
                                                    first or last name
```

Ship Example 3: Constructors

```
public class Ship3 {
                                        (In Ship3.java)
  public double x, y, speed, direction;
  public String name;
  public Ship3(double x, double y,
               double speed, double direction,
               String name) {
    this.x = x; // "this" differentiates instance vars
    this.y = y; // from local vars.
    this.speed = speed;
    this.direction = direction;
    this.name = name;
  }
  private double degreesToRadians(double degrees) {
    return(degrees * Math.PI / 180.0);
  }
  . . .
```

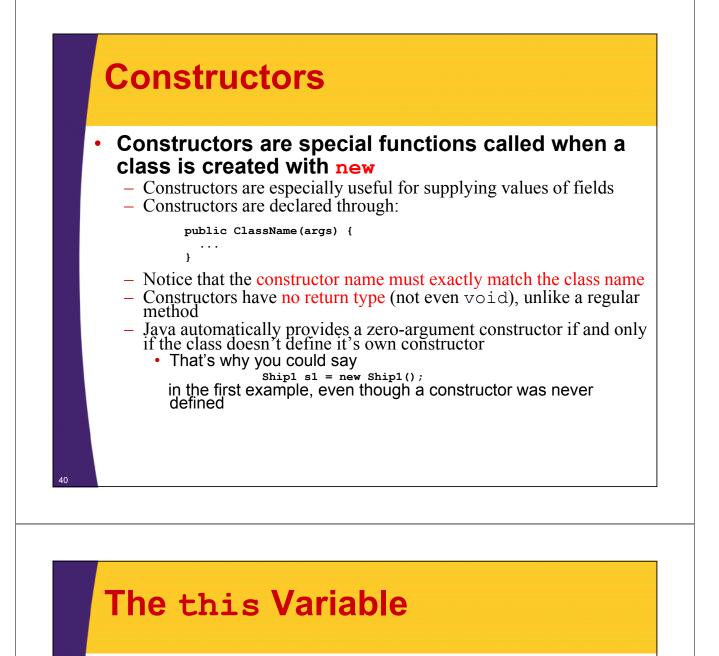
Constructors (Continued)

```
public void move() {
    double angle = degreesToRadians(direction);
    x = x + speed * Math.cos(angle);
    y = y + speed * Math.sin(angle);
  }
  public void printLocation() {
    System.out.println(name + " is at ("
                       + x + "," + y + ").");
  }
}
                                              (In Test3.java)
public class Test3 {
  public static void main(String[] args) {
    Ship3 s1 = new Ship3(0.0, 0.0, 1.0, 0.0, "Ship1");
    Ship3 s2 = new Ship3(0.0, 0.0, 2.0, 135.0, "Ship2");
    s1.move();
    s2.move();
    s1.printLocation();
    s2.printLocation();
  }
```



Example 3: Major Points

- Format of constructor definitions
- The "this" reference
- Destructors (not!)

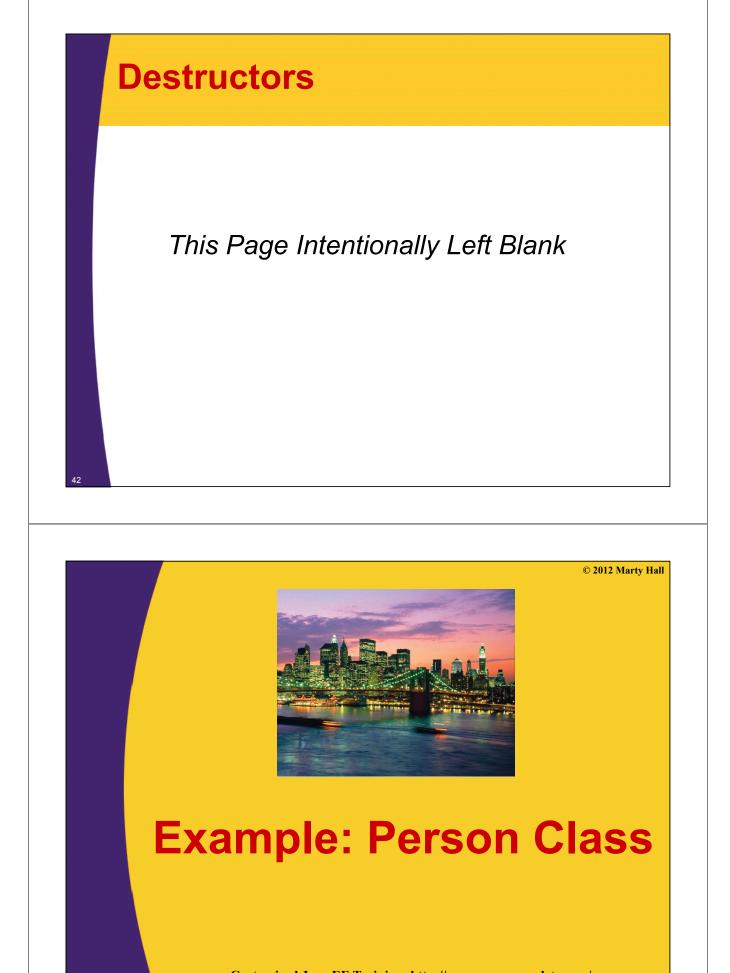


• The this object reference can be used inside any non-static method to refer to the current object

- The common uses of the this reference are:
 - 1. To pass a reference to the current object as a parameter to other methods

someMethod(this);

- 2. To resolve name conflicts
 - Using this permits the use of instance variables in methods that have local variables with the same name
 - Note that it is only necessary to say this.fieldName when you have a local variable and a class field with the same name; otherwise just use fieldName with no this



Customized Java EE Training: http://courses.coreservlets.com/ Java 6 or 7, JSF 2.0, PrimeFaces, Servlets, JSP, Ajax, Spring, Hibernate, RESTful Web Services, Android. Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Idea

Goal

- Make a class to represent a person's first and last name

Approach: 4 iterations

- Person with instance variables only
 - And test case
- Add a getFullName method
 - And test case
- Add a constructor
 - And test case
- Change constructor to use "this" variable
 - And test case
 - Also have test case make a Person[]

Iteration 1: Instance Variables

<u>Person.java</u>

public class Person {
 public String firstName, lastName;
}

<u>PersonTest.java</u>

Iteration 2: Methods

<u>Person.java</u>

}

public class Person {
 public String firstName, lastName;

public String getFullName() {
 return(firstName + " " + lastName);
}

PersonTest.java

Iteration 3: Constructors

Person.java

public class Person {
 public String firstName, lastName;

```
public String getFullName() {
    return(firstName + " " + lastName);
```

<u>PersonTest.java</u>

public class PersonTest {
 public static void main(String[] args) {
 Person p = new Person("Larry", "Page");
 System.out.println("Person's full name: " +
 p.getFullName());
 }
}

} }

}

Iteration 4: Constructors with the "this" Variable (and Arrays)

<u>Person.java</u>

}

}

<u>PersonTest.java</u>

this.firstName = firstName; this.lastName = lastName;

public String getFullName() {
 return(firstName + " " + lastName);

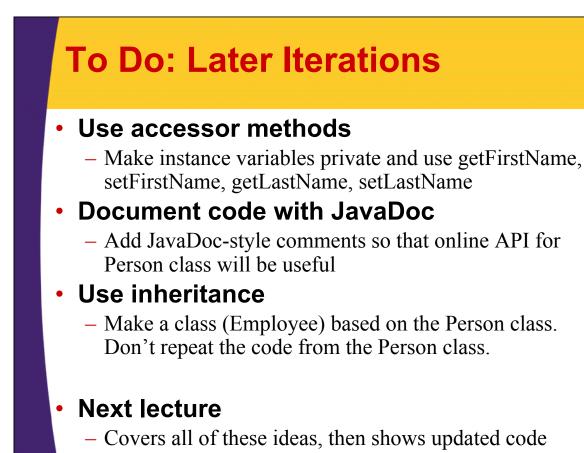
public class PersonTest {
 public static void main(String[] args) {
 Person[] people = new Person[20];
 for(int i=0; i<people.length; i++) {
 people[i] =
 new Person(NameUtils.randomFirstName(),
 NameUtils.randomLastName());
 }
 for(Person person: people) {
}
</pre>

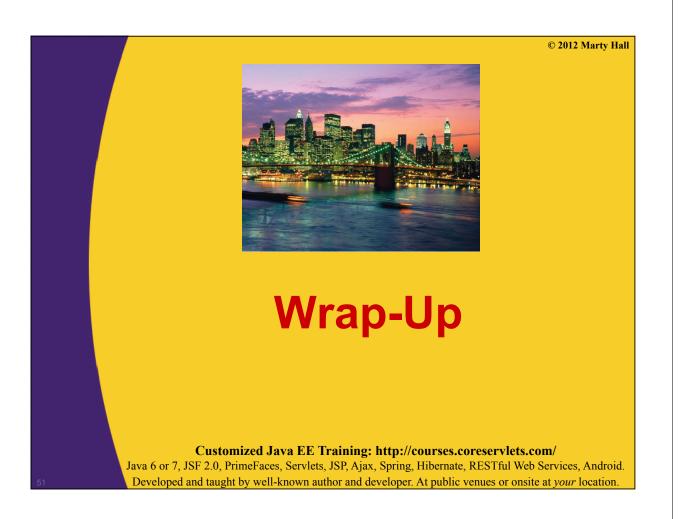
or(Person person: people) { System.out.println("Person's full name: " + person.getFullName());

Helper Class for Iteration 4

} }

```
public class NameUtils {
   public static String randomFirstName() {
      int num = (int) (Math.random()*1000);
      return("John" + num);
   }
   public static String randomLastName() {
      int num = (int) (Math.random()*1000);
      return("Smith" + num);
   }
}
```





<section-header>Summary Summary Summary



© 2012 Marty Hall

Questions?

Customized Java EE Training: http://courses.coreservlets.com/ Java 6 or 7, JSF 2.0, PrimeFaces, Servlets, JSP, Ajax, Spring, Hibernate, RESTful Web Services, Android. Developed and taught by well-known author and developer. At public venues or onsite at *your* location.