

# Intro to Java

- **Anatomy of a Class & Terminology**
- **Running and Modifying a Program**

# The Plan

- ❖ Go over MoveTest.java
  - Similar to Horstmann p. 48
- ❖ Basic coding conventions
- ❖ Review with GreeterTest.java (Horstmann)
- ❖ More terminology with Greeter.java (Horstmann)
- ❖ Homework 0 reminder
- ❖ Homework 1 assigned (due in 1 week)

# Why know the lingo?

- ❖ It's difficult to read the textbooks if you don't understand the words
- ❖ Your compiler error messages use specific words with specific meanings
- ❖ You need to be able to express your questions so others can understand them
- ❖ The answers you receive will use the lingo

# Terminology to Know

- ❖ **Package**
- ❖ **Class**
- ❖ **Import**
- ❖ **Keyword**
- ❖ **Public**
- ❖ **Object**
- ❖ **Identifier**
- ❖ **Declaration**
- ❖ **Definition**
- ❖ **Body**
- ❖ **Static**
- ❖ **Void**
- ❖ **Return**
- ❖ **Method**
- ❖ **Main**
- ❖ **Parameter**
- ❖ **String**
- ❖ **Array**
- ❖ **Type**
- ❖ **Variable**
- ❖ **Local**
- ❖ **Constructor**
- ❖ **Initialize**
- ❖ **Assign**
- ❖ **Arguments**
- ❖ **Comments**
- ❖ **Calling a method**
- ❖ **System.out.println**

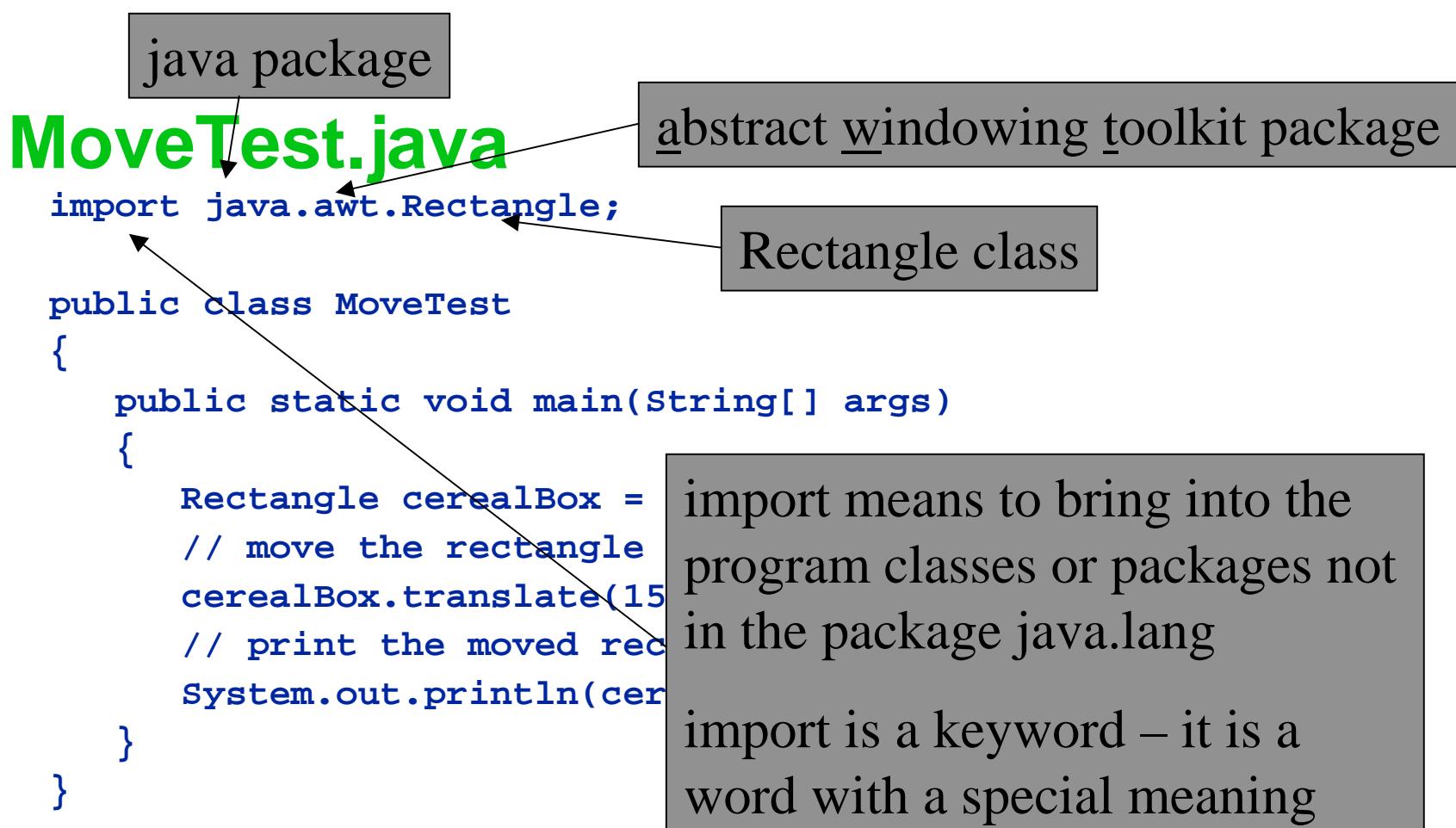
# MoveTester.java

```
import java.awt.Rectangle;

public class MoveTest
{
    public static void main(String[] args)
    {
        Rectangle cerealBox = new Rectangle(5, 10, 20, 30);
        // move the rectangle
        cerealBox.translate(15, 25);
        // print the moved rectangle
        System.out.println(cerealBox);
    }
}
```

*Prints*

```
java.awt.Rectangle[x=20, y=35, width=20, height=30]
```



### Prints

```
java.awt.Rectangle[x=20, y=35, width=20, height=30]
```

public means usable by everything, public is also a keyword

## MoveTest.java

```
import java.awt.Rectangle;
public class MoveTest
{
    public static void main(String[] args)
    {
        Rectangle cerealBox = new Rectangle(5, 10, 20, 30);
        // move the rectangle
        cerealBox.translate(15, 25);
        // print the moved rectangle
        System.out.println(cerealBox);
    }
}
```

**Prints**

```
java.awt.Rectangle@123456789012345678
```

class means instruction and data for  
making objects,  
class is a keyword

MoveTest is the name of the class  
A class name must match the file name.  
Names are also called identifiers.  
Identifiers and keywords are mutually exclusive.

# MoveTest.java

```
import java.awt.Rectangle;
```

```
public class MoveTest
```

```
{
```

```
    public static void main(String[] args)
    {
        Rectangle cerealBox = new Rectangle(5, 10, 20, 30);
        // move the rectangle
        cerealBox.translate(15, 25);
        // print the moved rectangle
        System.out.println(cerealBox);
    }
}
```

{ } Starts and ends  
class body

class declaration

class definition.

class body

# MoveTest.java

```
import java.awt.Rectangle;  
  
public class MoveTest  
{  
    public static void main(String[] args)  
    {  
        Rectangle cerealBox = new Rectangle(5, 10, 20, 30);  
        // move the rectangle  
        cerealBox.translate(15, 25);  
        // print the moved rectangle  
        System.out.println(cerealBox);  
    }  
}
```

Static means one per class

void means no return value

main is the name  
of the method

*Prints*

```
java.awt.Rectangle[x=20, y=35, width=20, height=30]
```

# MoveTest.java

```
import java.awt.Rectangle;

public class MoveTest
{
    public static void main(String[] args)
    {
        Rectangle cerealBox = new Rectangle(5, 10, 20, 30);
        // move the rectangle
        cerealBox.translate(15, 25);
        // print the moved rectangle
        System.out.println(cerealBox);
    }
}
```

String is a sequence of characters

[] means an array

args is a parameter

*Prints*

```
java.awt.Rectangle[x=20, y=35, width=20, height=30]
```

# MoveTest.java

```
import java.awt.Rectangle;  
  
public class MoveTest  
{  
    public static void main(String[] args)  
    {  
        Rectangle cerealBox = new Rectangle(5, 10, 20, 30);  
        // m  
        cerealBox.setRect(25, 25);  
        // p  
        System.out.println(cerealBox);  
    }  
}
```

method declaration

{} Starts and ends method body

method body

**Prints**

java.awt.Rectangle[x=20, y=35, width=20, height=30]

method definition

# MoveTest.java

```
import java.awt.Rectangle;  
  
public class MoveTest  
{  
    public static void main(String[] args)  
    {  
        Rectangle cerealBox = new Rectangle(5, 10, 20, 30);  
        // move the rectangle  
        cerealBox.translate(15, 25);  
        // print the moved rectangle  
        System.out.println(cerealBox);  
    }  
}
```

Rectangle is a type (also a class)

cerealBox is a variable

Creates a Rectangle  
Calls the constructor of  
the Rectangle class

**Prints**

```
java.awt.Rectangle[x=20, y=35, width=20, height=30]
```

# MoveTest.java

```
import java.awt.Rectangle; Declaring the cerealBox variable of type Rectangle
```

```
public class MoveTest
{
    public static void main(String[] args)
    {
        Rectangle cerealBox = new Rectangle(5, 10, 20, 30); // move the rectangle
        cerealBox.translate(15, 25); // print the moved rectangle
        System.out.println(cerealBox);
    }
}
```

Initializing the cerealBox variable

**Prints**

```
java.awt.Rectangle[x=20, y=35, width=20, height=30]
```

# MoveTest.java

```
import java.awt.Rectangle;

public class MoveTest
{
    public static void main(String[] args)
    {
        Rectangle cerealBox = new Rectangle(5, 10, 20, 30);
        // move the rectangle
        cerealBox.translate(15, 25);
        // print the moved rectangle
        System.out.println(cerealBox);
    }
}
```

**Prints**

```
java.awt.Rectangle[x=20, y=25]
```

Assignment operator

Pronounced “gets”

1. Computes the right hand side
2. Assigns value to left hand side

# MoveTest.java

```
import java.awt.Rectangle;
```

```
public class MoveTest
{
```

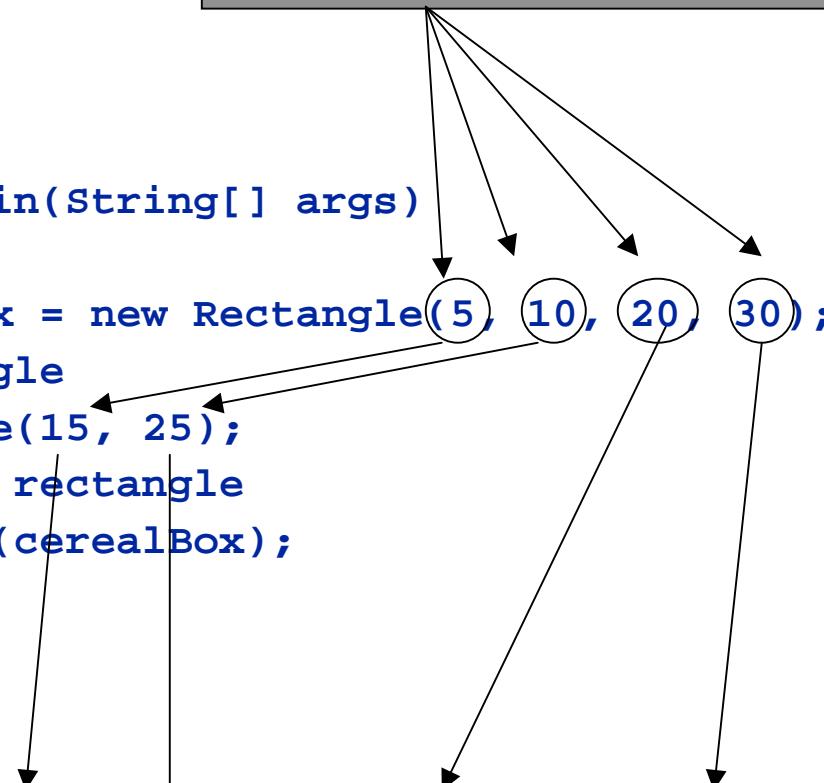
```
    public static void main(String[] args)
    {
```

```
        Rectangle cerealBox = new Rectangle(5, 10, 20, 30);
        // move the rectangle
        cerealBox.translate(15, 25);
        // print the moved rectangle
        System.out.println(cerealBox);
    }
}
```

**Prints**

```
java.awt.Rectangle[x=20, y=35, width=20, height=30]
```

Arguments – order matters



# MoveTest.java

```
import java.awt.Rectangle;

public class MoveTest
{
    public static void main(String[] args)
    {
        Rectangle cerealBox = new Rectangle(5, 10, 20, 30);
        // move the rectangle
        cerealBox.translate(15, 25);
        // print the moved rectangle
        System.out.println(cerealBox);
    }
}
```

Comments  
Ignored by compiler

*Prints*

`java.awt.Rectangle[x=20, y=35, width=20, height=30]`

# MoveTest.java

```
import java.awt.Rectangle;  
  
public class MoveTest  
{  
    public static void main(String[] args)  
    {  
        Rectangle cerealBox = new Rectangle(5, 10, 20, 30);  
        // move the rectangle  
        cerealBox.translate(15, 25);  
        // print the moved rectangle  
        System.out.println(cerealBox)  
    }  
}  
  
Prints  
java.awt.Rectangle[x=20, y=35, width=20, height=30]
```

Local variables are declared in method bodies

Calling the translate method  
On the cerealBox object

Rectangle object  
Also a local variable

# MoveTest.java

```
import java.awt.Rectangle;

public class MoveTest
{
    public static void main(String[] args)
    {
        Rectangle cerealBox = new Rectangle(5, 10, 20, 30);
        // move the rectangle
        cerealBox.translate(15, 25);
        // print the moved rectangle
        System.out.println(cerealBox);
    }
}
```

Calling the println method  
For console output

*Prints*

```
java.awt.Rectangle[x=20, y=35, width=20, height=30]
```

# Why know and follow the Java Coding Conventions?

- ❖ **Helps understand code**
  - makes purpose of identifiers clear
  - delineates separate pieces of code
  - assists in avoiding syntax errors
- ❖ **Expected if source code is to be viewed at any time by anyone other than the original author**
- ❖ **Helps standardize**

# Coding Conventions

- ❖ **Capitalization**
  - Class identifier
  - Variable identifier
  - Method identifier
- ❖ **Indentation**
  - Braces
  - Body of code (also called a code block)
- ❖ **See course webpage for a complete description**

# GreeterTest.java

```
public class GreeterTest
{
    public static void main(String[] args)
    {
        Greeter worldGreeter = new Greeter("World");
        System.out.println(worldGreeter.sayHello());

        Greeter daveGreeter = new Greeter("Dave");
        System.out.println(daveGreeter.sayHello());
    }
}
```

# Greeter.java

```
public class Greeter
{
    public Greeter(String aName)
    {
        name = aName;
    }

    public String sayHello()
    {
        String message = "Hello, " + name + "!";
        return message;
    }

    private String name;
}
```

Constructor

Used to initialize instance variables

Has no return type, not even void

Name is the same as the class name

Declaration of instance variable

Outside method body

Inside class body

# Greeter.java

```
public class Greeter
{
    public Greeter(String aName)
    {
        name = aName;
    }

    public String sayHello()
    {
        String message = "Hello, " + name + "!";
        return message;
    }

    private String name;
}
```

Empty parameter list

String concatenation

Private means only accessible within class body

```
graph TD; A[Empty parameter list] --> B["String concatenation"]; C[Private means only accessible within class body] --> D["Hello, " + name + "!"]
```

# Introduction to Java

## Downloading Source Code

- ❖ **Start up Eclipse**
  - (In ICC: In applications folder on desktop)
- ❖ **Set snarf site to:**  
<http://www.cs.duke.edu/courses/spring06/cps004/snarf>

# Assignment #1

- ❖ **Create your Class Web Page**
  - See assignment on web
- ❖ **Due Tuesday, 1/24**
- ❖ **(Assignment #0 Due Today!)**