# 6.270 Java Crash Course

**Jan-07-2000**

Written by kenlu & mdeeds



# Agenda

- OOP Vocabulary
- Objects and Abstraction
- Language Basics
- Files and Naming
- APIs
- JDK
- Good Habits
- Resources



# OOP Vocabulary

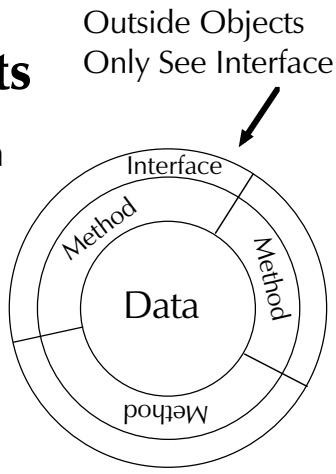
- Classes/Objects
- Instance/Instantiate
- Fields/Constructor/Method
- Static Classes
- Visible/Visibility (Public/Private)
- Call/Pass/Return



# Objects

- Object Encapsulation

An Object:




- Subclassing
- Examples:

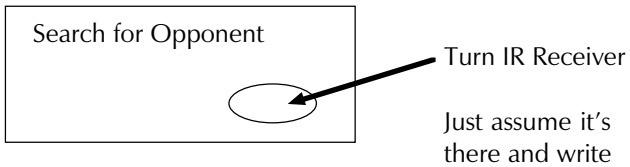
Woodpecker ⊏ Bird ⊏ Animal

PIII ⊏ PII ⊏ i386 ⊏ i286



# Abstraction

- Visibility (Public/Private)
- The Black Box
- Wishful Thinking:




Just assume it's there and write it later!

- Interfaces



# Language Basics

- Sequential Flow of Control
- Not a Constraints System

  (Statements, Not Assertions)
- Expressions vs. Statements
- Value vs. Reference

# Language Basics (cont'd)

**Data Types:**

```
byte, short, int, long,
float, double,
boolean,
char, String*
```

*`String` is not really a primitive data type!



# Language Basics (cont'd)

**Comments**

```
// This is a comment.

x=5; // This part is a comment.

/* This comment can
   span multiple lines! */
```



# Language Basics (cont'd)

**A Basic Class**

```
public class Foo {
  // Foo's value
  private int val;

  // Default Constructor
  public Foo() {
    val = 270;
  }

  // Returns value
  public int getVal() {
    return val;
  }
}
```

These lines are part of the interface



# Language Basics (cont'd)

**Method Overloading**

```
public class Foo {
  private int val;

  public Foo() {
    val = 0;
  }

  // initialVal: The value we want
  // to preset Foo's value to
  public Foo(int initialVal) {
    val = initialVal;
  }
}
```

Two constructors with the same name! (But different parameters.)



# Language Basics (cont'd)

```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }
}
```



# Language Basics (cont'd)

**Using Methods and Fields (1 of 2)**

```
public class Foo {
  public static final int DEFAULT_VAL = 42;
  private int val;

  public Foo() {
    val = DEFAULT_VAL;
  }

  // newVal: The initial value for Foo
  public void setVal(int newVal) {
    val = newVal;
  }

  // returns: Foo's value
  public int getVal() {
    return val;
  }
}
```

# Language Basics (cont'd)

## Using Methods and Fields (2 of 2)

```
public class MyProgram {
   public static void main(String[] args) {
      int v;
      Foo f;
      Foo g;
      f = new Foo();
      g = new Foo();
      v = f.getVal();       // v is now 42
      g.setVal(17);
      v = f.getVal();       // v is still 42
      v = g.getVal();       // v is now 17
      v = Foo.DEFAULT_VAL;  // v is again 42
   }
}
```



# Language Basics (cont'd)

## Subclasses (Extending Classes) (1 of 2)

```
public class Foo {
   public static final int DEFAULT_VAL = 42;
   private int val;

   public Foo() {
      this(DEFAULT_VAL);
   }

   public Foo(int initialVal) {
      val = initialVal;
   }

   ...
}
```



# Language Basics (cont'd)

## Subclasses (Extending Classes) (2 of 2)

```
// Bar is like foo, with the option of returning
// value as String description.
public class Bar extends Foo {
   public Bar() {
      super();
   }

   public Bar(int initialVal) {
      super(initialVal);
   }

   // Returns String describing Foo's value.
   public String getValString() {
      String s;
      s = "My value is: " + super.getVal();
      return s;
   }
}
```



# Files and Naming

This Is Quasi-Mandatory

class MyProgram -> MyProgram.java

**Fields and Methods:**
`lowerFirstButUpperForEachNewWord`

**Classes:**
`AllWordsBeginWithUpper`

**Constants:**
`ALL_CAPS_SEPARATED_BY_UNDERSCORE`



# Application Program Interfaces (APIs)

- Why APIs?
  - Black Box
  - Abstraction
  - Distribution


- How Do You Read It?
  - Interface
  - Documentation



# Java Development Kit (JDK)

setenv CLASSPATH .:roboskiff.jar:${CLASSPATH}
Sets up Java's enviornment so you can compile properly.

javac *.java
Compiles your code into .class files

jar -cvf usercode.jar *.class
Combines .class files into a single file, for upload to your robot

# Good Habits

- Why Abstraction/Black Boxes?
  - Ease of Code Modification
  - More Debuggable
- Visibility (Public vs. Private)
  - Private Fields and Public Methods
  - When to Use Public Fields?
    - Hardly Ever — Instead Use getX()/setX(int x)
    - Constants (public static final int)
  - When to Use Private Methods?
    - Helper Functions



# Good Habits (cont'd)

- Use Long, Descriptive Variable Names!
- Be Consistent in Style (Resolve Inconsistencies with Teammates)
- **DOCUMENT DOCUMENT DOCUMENT!**
  - Document Every Method, Field, and Class
  - Purpose and Definitions
  - Any Weird (Complicated) Code
  - What is Clear to You Isn't Necessarily Clear to Anyone Else, Including (You + A Few Days)
- Have fun!



# Resources

- The Java API:
  http://java.sun.com/products/jdk/1.1/docs/api/packages.html

- Syntax References:
  http://www.cs.brown.edu/courses/cs015/1999/ReferenceGuides/JavaRefGuide/contents.html

- Tutorial:
  http://www.bath.ac.uk/~ccsnad/java/javatutorial.html