Introduction to Java for FIRST Robotics

Andrew Merrill
Software Mentor, FRC Team 1540
Computer Science Teacher, Catlin Gabel School

Goldilocks and the Three Javas

- Small - Java Micro Edition (ME)
  - designed for mobile and embedded devices
  - used for FRC robotics

- Medium - Java Standard Edition (SE)
  - designed for regular laptop, desktop, server applications
  - the most common edition
  - widely used in computer science courses (including AP)

- Large - Java Enterprise Edition (EE)
  - designed for application servers, distributed systems

What is Java?

- Java is a *Programming Language*
  - documented by the Java Language Specification
  - "Java is object-oriented, strongly typed, with C-like syntax"

- Java is a *Class Library*
  - documented by the:
    - Java ME CLDC API Specification, or the
    - Java SE API Specification, or the
    - Java EE API Specification
  - "Java has graphics, threads, networking, hash tables, etc..."

- Java is a *Runtime Environment*
  - documented by the Java Virtual Machine Specification
  - "Java is compiled to bytecodes which are interpreted by a virtual machine"

Common Java Misconceptions

- Java is not just for Web programming or Applets
- Java is not JavaScript, despite the misleadingly similar name!

Sample Java Program

```
public class DemoBot extends IterativeRobot
{
  Joystick stick;
  Jaguar launcher;

  public void teleopInit()
  {
    stick = new Joystick(1);   // joystick on USB port 1
    launcher = new Jaguar(2);  // speed controller on PWM line 2
  }

  public void teleopPeriodic()
  {
    if (stick.getButton(Joystick.BUTTON_TRIGGER))
      launcher.set(0.75);
    else
      launcher.set(0.0);
  }
}
```

Classes

- A ***class*** defines a new type of ***object***
  - example classes: `DemoBot`, `Joystick`, `Jaguar`
- Each class contains two kinds of ***members*****:**
  - ***Fields***: variables that hold data needed by this object
    - example fields: `stick`, `launcher`
  - ***Methods***: functions that perform the object's actions
    - example methods: `getButton`, `set`
- By convention, class names begin with a capital letter

Objects

- An ***object*** is an instance of a class
- Objects are accessed via variables
- Variables are declared in advance to refer to objects from a particular class
  - Example: `Jaguar launcher;`
- Objects are created with the operator **`new`**
  - Example: `launcher = new Jaguar(2);`
- Members of an object are accessed with the syntax *variable.member*
  - Example: `launcher.set(0.75);`
- When no variable refers to an object anymore, it is automatically "garbage collected"
- By convention, variable and function names begin with a lower case letter

Inheritance

- A child class can ***extend*** a parent class
  - alternative terminology: a sub-class can extend a super-class
- Objects of the child class can do everything that objects of the parent class can do
  - child class objects ***inherit*** the fields and methods of the parent class
  - allows code to be shared between similar classes
- Examples:
  - `class DemoBot extends IterativeRobot`
  - `class Jaguar extends PWM`
  - `class Victor extends PWM`
  - `class Servo extends PWM`
- Child classes can ***override*** parent class methods
  - new method must have the same name and parameters
- Differences from C++
  - no multiple inheritance
  - all methods can be overridden by default
  - all classes extend the built-in `Object` class

Interfaces

- An ***interface*** is like a class, but...
  - it has no fields
  - its methods have no bodies
- So what does it have?
  - method names with their parameters and return type
- A class can ***implement*** an interface (or several interfaces)
- Think of an interface as a promise to write certain methods
- Example interface:

```
interface SpeedController
{
```

```
        double get();
        void set(double speed);
    }
```

- To implement an interface:
  - `class Jaguar extends PWM implements SpeedController`

Static and Final

- A ***static*** field is a class field, not an object field
- A ***final*** field is a constant - its value can't be changed
- Example: `static final int maxAngle = 90;`
- Example: `Joystick.BUTTON_TRIGGER`

- A ***static*** method can be run without making an object first
- Example: `time = Utility.getFPGATime();`

Packages and Importing

- Java classes can be organized into ***packages***
- Each package goes in a separate directory (or folder)
- How to use a class from a different package:
  - Write the package name every time you use the class name
    - Example: `stick = new FIRST.wpilibj.Joystick(1);`
  - or import the class from the package
    - Example: `import FIRST.wpilibj.Joystick;`
  - or import every class from the package
    - Example: `import FIRST.wpilibj.*;`
- The Java library package `java.lang` is always imported automatically

Class Member Access Control

- Java restricts who can access the members of a class

	Can be accessed from the *same class*	Can be accessed from the *same package*	Can be accessed from *any child class*	Can be accessed from *any class*

`private`	yes	no	no	no
(default)	yes	yes	no	no
`protected`	yes	yes	yes	no
`public`	yes	yes	yes	yes

Java Data Types

- Number Types
  - Integers
    - **`byte`** (8 bits, range from -128 to 127)
    - **`short`** (16 bits, range of ± 32767)
    - **`int`** (32 bits, range of about ± 2 billion)
    - **`long`** (64 bits, range of about ± 9 quintillion or 10^{19})
  - Floating Point
    - **`float`** (32 bits, range of about ± 10^{38}, precision of about 7 decimal digits)
    - **`double`** (64 bits, range of about ± 10^{308}, precision of about 16 decimal digits)
- Other Types
  - **`boolean`** (`true` or `false`)
  - **`char`** (one Unicode character)
- **`String`** (standard library class)
- wrapper classes: **`Byte`**, **`Short`**, **`Integer`**, **`Long`**, **`Float`**, **`Double`**, **`Boolean`**, **`Character`**

Note: No unsigned numbers, unlike C/C++

Math

- **`+`** for addition
- **`-`** for subtraction
- **`*`** for multiplication
- **`/`** for division (warning: if you divide two integer types, you'll get an integer type result)
- **`%`** for remainder after division (so 10 % 3 is 2)
- **`Math.sqrt(x)`** for square root
- **`Math.abs(x)`** for absolute value
- **`Math.sin(x), Math.cos(x), Math.tan(x)`** for trigonometry

Randomness

- The Java library provides a class called **`Random`**
- It is in the **`java.util`** package, so you can **`import java.util.Random;`**
- A **`Random`** object is a random number generator

- Only create one random number generator per program!
  - Example: **`public static Random generator = new Random();`**

- Example: How to generate a random integer in the range 0...359:
  - **`int spinDirection = generator.nextInt(360);`**

- Example: How to generate a random floating point number in the range 0...1:
  - **`double probability = generator.nextDouble();`**

Casting

- To force a **`double`** into an **`int`** (losing the decimal part):
  - **`int x = (int) 3.7;`**

- To force an **`int`** to be treated as a **`double`**:
  - **`double average = ((double) total) / count;`**

- To tell Java that an **`Object`** is really a **`Jaguar`**:
  - **`Jaguar launcher = (Jaguar) getSpeedController();`**

Java ME Data Structures

- Arrays
  - Fixed number of elements
  - All elements of the same type (or compatible types)
  - Random access by cell index number
  - Example:

```
int data[] = new int[100];
data[0] = 17;
data[5] = data[0] + 1;
System.out.println(data[17]);
```

- Vector
  - Variable number of elements
  - All elements must be objects
  - Random access by cell index number
  - Example:

```
Vector speedControllers = new Vector();
speedControllers.addElement(new Jaguar(1));
Jaguar controller = (Jaguar) speedControllers.elementAt(0);
```

Hashtable

- Otherwise known as a dictionary, map, associative array, lookup table
- Given a key, can quickly find the associated value
- Both the key and value must be objects
- Example:

```
Hashtable animals = new Hashtable();
animals.put("cow", "moo");
animals.put("chicken", "cluck");
animals.put("pig", "oink");
String chickenSound = (String) animals.get("chicken");
System.out.println("a chicken goes" + chickenSound);
```

Exceptions

- When your program crashes, Java throws an Exception
- Example:

```
java.lang.ArithmeticException: / by zero
   at DemoBot.teleopPeriodic(DemoBot.java:15)
```

- You can catch and handle Exceptions yourself:

```
try {
     // do possibly dangerous stuff here
     // keep doing stuff
}
catch (Exception e) {
    launcher.set(0);
    System.out.println("launcher disabled");
    System.out.println("caught: " + e.getMessage());
}
```

Resources

- Websites
  - Unofficial pre-release FIRST WPILIB Java library:
    - http://livemyst.comli.com/wpilibj/doc/index.html
    - Note: everything on that site is unofficial and subject to change at any time!
  - Java ME CLDC class library documentation:
    - http://java.sun.com/javame/reference/apis/jsr139/
  - WPI's Java for FRC page:
    - http://users.wpi.edu/~bamiller/java/index.html
  - FRC 2010 Java Beta Test Forum:
    - http://forums.usfirst.org/forumdisplay.php?f=1260
  - Java Forum on Chief Delphi
    - http://www.chiefdelphi.com/forums/forumdisplay.php?f=184
  - Sun's Java Tutorial

        - http://java.sun.com/docs/books/tutorial/
- Books
    - *Java in a Nutshell* by David Flanagan (O'Reilly)
    - *Effective Java* by Joshua Bloch