Basic Java Course Syllabus

Tim Magoun and Aravind Koneru

Compiled on Sunday 3rd July, 2016 at 16:57

Abstract

In order to create a proficient programming sub-team, the new members must know how to program in Java, and become comfortable with the concept of inheritance. This will be accomplished through a series of Java courses instructed with the help of the various lesson plans and assessments included in this project. The *Basic Java Course* will use four segments of two hours each in order to teach students, from the ground up, about programming in Java. Note that this course is not a substitution to a proper course in Java, but instead is a crash-course to prepare students for basic robot controlling code for FIRST® Robotics Competition

Syllabus

- Setup Eclipse
- Structure of Programming
- Primitive Types
- Basic Operators
- Arrays
- Comparative Operators
- Flow Control
- Methods
- Objects
- Modifiers
- Java Library Features
- Inheritance

Day 1

Note to Instructor:

Bring a copy of both 32 bit and 64 bit Eclipse in case of slow or no internet.

Objective:

By the end of this lesson, the students will be able to perform basic calculations using Java's primitive types.

Prerequisites:

Working computer with wifi capabilities the authority to install software.

Install (or Update) Eclipse

1. Go to `https://eclipse.org/downloads/eclipse-packages/`
2. Click on the corresponding installer, 32 bit or 64 bit (if you don't know the version of OS present, choose the 32 bit installer)
3. Download the installer to a known location (ex. Downloads or Desktop)
4. Execute the installer file
5. Select Eclipse IDE for Java Developers
6. Confirm install location and select preferred shortcut locations
7. Accept EULA
8. Bogosort the digits of π
9. Launch Eclipse Neon and set up preferences, line numbers are highly recommended

Homework:

Write a line of code that will calculate from the right to left. ex `int x = 4 + 5 * ( 6 - 7)`

Day 2

Note to Instructor:

It is essential that the students recognize the modular nature of comparison operators, as it is used very often in FRC programming.

Objective:

In this lesson, the students will learn about single dimensional arrays, and the various control flow statements that exists in Java. By the end of this lesson, the students will be able to recognize and analyze logical and bit-wise comparisons, and use those comparisons to create simple loops that will manipulate a single-dimensional array.

Prerequisites:

Knowledge of the primitive types and basic operators.

Homework:

Write a program that will begin by automatically fill in an array of `double` with multiples of 1.7, and then traverse through the array to change all of the elements that are odd to two times the initial value. Print out those values in a single line.

Day 3

Note to Instructor:

This is the hardest portion of learning basic Java. Use examples to emphasize the nature of objects and how classes act as a blueprint while objects act like machines produced by those blueprints. Students must have a good understanding of Objects in order to understand the Command Based Programming used in FRC.

Objective:

In this lesson, students will dive into the world of Object-Oriented Programming (OOP). After this lesson, students will be able to write basic classes. They will also learn about modifiers such as `final` and `static`

Prerequisites:

Before starting this lesson, the students must be comfortable with the manipulation of variables and the various operators. They should also know the different ways in which `for`, `while`, `if` etc. could influence the execution of the program.

Homework:

Write a class named Circle, and include a default constructor and a overloaded constructor for a `double radius`. Write methods that will return the area and perimeter using a value of π from the included Math class.

Write a Triangle class that takes in a `int[][]` that stores the integer coordinates of the points on the triangle. Then write a method in that class that returns the perimeter, area, and for challenge, the centroid of the triangle.

Write a NumberProcessor class that will contain a set of `static` methods and constants.

1. Implement the algorithm in Day 2's homework. As `public double[] oddDoubler{double[] input}`
2. Implement an algorithm that returns the sum of the digits of the input integer