Elements of the Java Platform

Road Map

Instructor: Jonathan Doughty

E-mail: jdoughty@cs.gmu.edu

Times: Monday Evenings: 7:20 - 10:00 PM

Place: Classes will be held in S&T I, Room 124

Each week lecture materials will be found at: http://cs.gmu.edu/~jdoughty/cs161/

Topics this week:

- Course Description
- Grading Policy
- Honor Policy
- Expectations for This Class
- Textbook
- Syllabus
- Programming Assignment Road Map
- Program Development Environment
- IDEs
- Programming
- Some Keys to Java Programming
- What Does Java Look Like?
- Running a Java application
- A Better First Java program
- Do it!
- What the MyName Class demonstrates
- Assignment for Next Session
Course Description

Description
On completing this course, students will understand

- the essentials of computer programming,
- the fundamentals of the Java™ language and platform, and,
- writing and running simple Java applications.

Class sessions will not be primarily hands-on lab oriented.

This class will not teach

- JavaScript
- How to create graphical interfaces,
- Any advanced Java topics
Grading Policy

Programming assignments - 70%

- 4 programming assignments
- Due the following Monday after the assignment is given
- I’ll accept late submittals up to one week after their initial due date; however, all late submittals will be reduced one full letter grade.
- I’ll accept good faith attempts on time and I’ll revise those grades upwards (not to maximum credit however) with results from late submittals.

Final exam - 30%

- I base exam questions on the text’s self test questions and the lecture material.
Honor Policy

GMU Honor System and Code
http://www.gmu.edu/catalog/acadpol.html

Computer Science Department Honor Code Policy for Programming Projects
http://cs.gmu.edu/honor-code.html

- Do your own work
- Assignments to be done individually, NOT in teams
Expectations for This Class

- Assignments are expected to be done by the **individual**. Each individual is expected to complete and hand in their own version of programming assignments.

- **Full credit if correct and turned in on time. Only partial credit thereafter.**

- Programming assignments are expected to show individual "**value-added**". If you’ve used material you’ve found to help complete the assignment (including material from the text or examples I’ve provided) I expect you to have added to it in a *significant* way.

- Plagiarism (e.g., copying Java code from books, CDs, or web sources) *without proper and complete attribution* and adherence to copyrights, etc. is **not allowed**.

Final exam

- Will be completed by the individual.

- Will be taken without study aids.
Textbook

Java: An Introduction to Computer Science & Programming
Walter Savitch, Prentice Hall, 1998

Recommended Supplementary Reading
Useful but not required:

The Java Tutorial
Mary Campione and Kathy Walrath,

- Softcopy http://java.sun.com/docs/books/tutorial/index.html
  Download your own copy from
  http://java.sun.com/docs/books/tutorial/information/download.html (9.3MB)
  updated regularly
  - 22 "trails"
  - 64 lessons
  - Over 16 Megabytes of information all together.
- Printed - Addison-Wesley, 1998

Getting Started
A series of lessons on getting started in Java programming; assumes a little programming background.
http://developer.java.sun.com/developer/onlineTraining/Programming/

Thinking in Java
Bruce Eckels’ excellent book on Java programming.

- Printed - Prentice-Hall 1998
- Softcopy You can download the entire contents in either PDF (3.6MB) or zipped HTML (1.1MB) formats from
  http://www.bruceeckel.com/javabook.html
Syllabus

The topics I expect to cover in this class are:

**Week 1 - Elements of Programming in Java**

- What is *programming*?
- What is *Java*?
- What are the essentials for writing and running Java code?
- Introduction to Java primitive and object data types

**Week 2 - The Java Language**

- What are the *primitive* data types of Java
- What are the *fundamentals* of programming in Java?
- What is a Java *Object*?

**Week 3 - Java Objects, Methods, Fields, and Exceptions**

- Creating Java objects
- *Arrays* and other Java collections
- Decomposing a problem into classes

**Week 4 - Java’s Object Orientation, Input, and Output**

- Java’s approach to *object orientated programming*
- Associating data with objects
- Data *input* and *output* in Java
Week 5 - Inheritance and Graphical User Interfaces

- The Java platform’s use of inheritance and packages
- Event Driven Programming
- Making a graphic Java User Interface
- Applets and Java in Web Browsers
Programming Assignment Road Map

Assignment 1

Goal:

- Create an easy program and get your chosen Java environment to work.

Purpose:

- To get you used to the compiler and the interpreter;
- For you to experience common initial problems;
- Decide where you will do the remainder of the assignments;
- Get familiar with tools.

Assignment 2

Goal:

- Write a simple Java program from scratch;
- Learn how to write a Java class with fields and methods.

Purpose:

- To start thinking in terms of breaking a problem into smaller pieces.
- To start getting the computer to calculate answers. To start learning to be exact when giving the computer instructions.
Assignment 3

Goal:

- Write a second Java class and use the two classes written so far, this week’s uses last’s.

Purpose:

- Learn about instances of classes and calling methods on them.
- To start building solutions from smaller pieces: Java classes; to get those pieces to interact.

Assignment 4

Goal:

- Add input/output capabilities to a Java class;
- read data, process it, and produce some results calculated from the input using multiple classes.

Purpose:

- To access data from an external source,
- To learn how to instruct the computer to process it,
- How to break data into pieces associated with objects.
- Learn about I/O principles;
- Building more complex programs by combining simple pieces.
Program Development Environment

JDK

The Java Development Kit (JDK), *free from Sun* and any text editor is all that is needed.

- You should be using **JDK 1.1.x** or **JDK 1.2.x** (also known as the Java™ 2 SDK, Standard Edition)
  - S&T 1. Room 124 has the Windows version of JDK 1.2 installed
  - OSF1 has the Unix version of JDK 1.1
  - Either one will do for this class

- If you choose to work at home or on some other system
  - The Java Development Kit 1.1 (currently 1.1.8) can be downloaded from [http://java.sun.com/products/jdk/1.1/](http://java.sun.com/products/jdk/1.1/) (8 MB download, 12 MB installed).
  - The Java2 "platform" (currently 1.2.2) can be downloaded from [http://java.sun.com/products/jdk/1.2/](http://java.sun.com/products/jdk/1.2/) (20 MB download, 43 MB installed)

- The JDK documentation will be a useful reference, use it.
  - The JDK 1.1 documentation may be found [here](http://java.sun.com/products/jdk/1.1/)
  - The JDK 1.2 documentation may be found [here](http://java.sun.com/products/jdk/1.2/)

  If you are installing a copy of the JDK at home consider downloading the separate JDK documentation ZIP archive at the corresponding URLs above.
(the JDK 1.1 version is a 4 MB download and requires 12 MB installed; the JDK 1.2 documentation is a 16 MB download and requires 83 MB installed.)

- If you are working on a machine that uses something other than Windows or Solaris, let me know.
IDEs
Lots of Interactive Development Environments supporting Java, some free, some commercial products, are available.

- You don’t need an IDE to do the work in this course.

- You will need to be able to use a simple text editor (Notepad on Windows, PICO on Unix will do) and have access to a Java compiler and a Java Virtual machine.

- One free possibility for Windows environments is Javaedit. We’ll use Javaedit in the lab. You can download your own free copy at http://www.tiac.net/users/dchase/javaedit.htm (260K download)

- The CD-ROM that accompanies the text has a limited version of the Code Warrior IDE. It is useful for exploring the example programs that accompany the text. The computers in the S&T I Room 124 lab have Code Warrior installed. I don’t recommend using CodeWarrior for this class.
Programming

Question: What are Programs? What are some examples?

Question: What is Programming?
Some Keys to Java Programming

- In Java everything is an Object.

- Java programs consist of one or more objects.

  Question: What’s an "object"?

- One object is the starting point of any Java program.

- This one object has a main method that starts a Java application going.

  All Java applications have at least one method named "main"
Elements of the Java Platform

What Does Java Look Like?

/* One of the simplest complete Java programs you can write */

public class HelloClass {

    // This is a "method"; notice its name

    public static void main(String[] args) {

        System.out.println("Hello class, this is Java!");

    } // this ends the method definition

    } // this ends the class definition
Running a Java application

1. Write Java code in a source file using a text editor.

2. Activate the Java compiler; For example, by typing

   javac MyName.java

   This converts your human readable Java source code to Java interpreter readable bytecode.

3. Run the program; activate the Java interpreter
   For example, by typing

   java ClassWithMain

   ClassWithMain must be identical to filename and is case sensitive. For example:

   C:\ java MyName "Your name here"

   This starts the interpreter and tells it to look for a file MyName.class
A Better First Java program
/** a Java class to demonstrate simple Java principles. */

public class MyName {

    public static void main(String[] args) {

        // Make a MyName object ...
        MyName anObject = new MyName();

        // Assign its name from the command line argument
        anObject.name = args[0];

        // ... and ask it to identify itself
        System.out.println( anObject.toString() );

        // The following is only needed for running the program from
        // within the JavaEdit application on Windows
        try {
            byte[] line = new byte[80];
            System.out.println("press enter key to end");
            System.in.read(line);
        } catch (java.io.IOException e) {
            // ignored
        }

        // The rest of this relates to MyName "objects"

        // each MyName object will remember who you tell it is using
        // this "instance variable" named "name".

        String name;

        // This will allow MyName objects to identify themselves asked.

        public String toString() {
            return "Hello, I'm a MyName object and my name is " + name;
        }
    }
}
Do it!
In this case, just access the already written source code and copy it into a new Javaedit window.

- Notice what happens if you don’t include "Your name here"
- Notice what happens if you don’t include the quote marks
- Notice what happens if you just type nonsense within the quote marks
What the MyName Class demonstrates

Java starts by finding and running the method named *main*

```
public static void main(String[] arg)
```

*String name defines a variable (or field)*

Notice where it is declared: inside the class but outside any method.

*The name variable is associated with MyName objects*

These are called *instance variables*

*You assign values to variables*

A variable (also called a *field*) is given a value using

```
destination = source expression;
```

*You call methods*

```
object_reference.method_name();
```

*You generate output*

```
System.out.println( [ a string ] )
```

*Java forces you to check for possible exceptions*

```
try {
    ...
} catch ( ...) {
    ...
}
```
Assignment for Next Session

Reading
If you haven’t already done so, read

- Chapter 1 - Introduction and a Taste of Java
- Chapter 2 - Primitive Types and Strings
- Chapter 3 - Flow of Control

Programming Assignment

- Decide what platform you are going to work on
  - Windows
    - GMU CS lab - use JavaEdit and JDK 1.2.2
    - At home or on some other PC - use JavaEdit and JDK 1.1.8
  - Unix
    - osf1 supports JDK 1.1.8 for non-GUI applications
      - use pico or vi and the javac and java commands
  - Other systems you may have access to have other challenges
    - Email me if you have questions.
- Using a simple text editor, (like vi, pico, javaedit, or Notepad) create a file Hello.java.
Write the class Hello so that it prints on four lines just:
- Your name
- Your GMU ID
- Your email address
- The date and time - you decide how to accomplish this; we’ll discuss some possibilities next week

You can get these values into the program in any way you like: from the command line or just by having "strings embedded in the source code".

Do not prompt the user for the values or bother trying to read the values from the keyboard. Do not try to use SavitchIn.

Hint: Make a copy of the MyName.java file; rename it Hello.java; and add what is necessary to fulfill the homework requirements.
- Run javac compiler
- Fix any errors
- Repeat last two steps if necessary until you have no more errors
- Run java interpreter on class file
- Turn in transcript of showing successful execution of javac and java as well as a copy of source file(s)

- On Windows copy and paste the contents of a DOS window into a JavaEdit text document window or Notepad. Or take a snapshot of the window showing the program execution using Alt PrintScreen and then paste the snapshot into Paint to print the resultant image.

- On Unix, use the script command to start recording. Do

```
javac Hello.java
java Hello
```
to show the compiler execution with no errors, record the program running, and list the program source file. Use the *exit* command to end the script; print the result that is stored in the file named *transcript*. 