CS49J
Lecture 1

INTRODUCTION

INSTRUCTOR: KATERINA POTIKA
CS SJSU

Information

- Lectures:
  - MW 1:30-2:45 PM

Office Hours

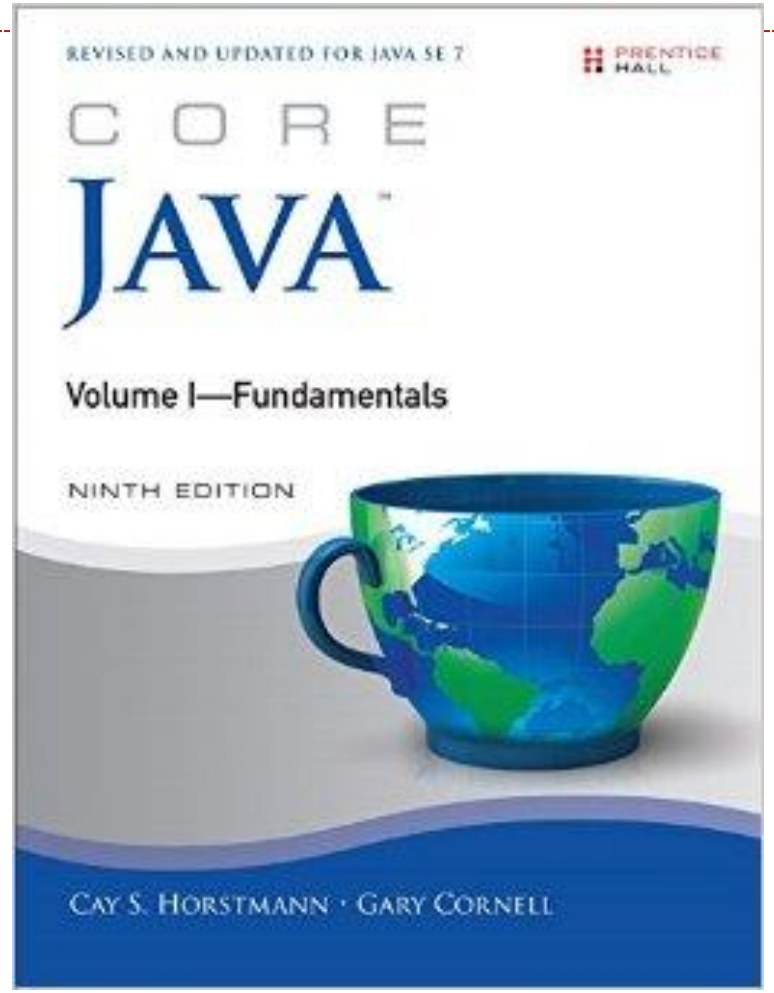
- Monday: 12:00-12:30PM & 3-4PM
- Wednesday: 12:30-13:00PM
- or by appointment

- Office: MH215

Prerequisites

- Programming with OOP other than Java; or instructor consent
- Need Transcripts! Highlight courses and show me till next lecture (otherwise dropped)

Textbook


REVISED AND UPDATED FOR JAVA SE 7
PRENTICE HALL
CORE
JAVA
Volume I—Fundamentals
NINTH EDITION
CAY S. HORSTMANN · GARY CORNELL

Important Dates (fixed)

- Midterms (30% of grade)
  - Wednesday, 10/7
  - Wednesday, 11/4
- Final (30% of grade)
  - 12/16, 12:15-14:30AM

Remaining 40% of grade

- 30% Programming assignments (~every 2 weeks)
- 10% Quizzes
- 0% Homework Assignments (~every week)
- 0% Reading Assignments

Final Grade

- Students must obtain >50% in each component of the course in order to be eligible for a grade of C- or better.

Course Objectives

- Ensure that students are familiar with fundamental concepts in software design, object-oriented programming and basic graphical user interface programming.
- Give students experience with the basic syntax and semantics of the Java language and programming environment.
- Give students an understanding of the concept of Abstract Data Types, and the distinction between interface and implementation.
- Give students an understanding of the standard collections interface types and classes such as lists, stacks, queues, hash tables, binary search trees and iterators, and how to use them in efficient Java applications.
- Enable students to write simple graphics programs involving the drawing of basic shapes.
- Acquaint students with exception handling mechanisms.

Student Learning Objectives

- Upon successful completion of this course, students should be able to:
- Write Java applications which are appropriately documented using Javadoc
- Use Java to read and write text files
- Implement from specifications Java classes that embody data structures
- Use and work with pre-existing implementations in the Java collections framework
- Use an iterator to traverse any collection
- Write a graphics program that draws simple shapes
- Use Java exceptions for error handling

CS49J Web site @ canvas

- Contains/Will be uploaded throughout the semester:
  - Announcements (News)
  - Green Sheet (Syllabus)
  - Lecture slides
  - Homework Assignments
  - Project Assignments
  - Upload your Assignments
  - Important dates
  - Online forum
    - Discussion forum
    - Email (may forward to your regular email)

It is your responsibility to visit the web site regularly (every 1-2 days) in order to access important information related to the course!

ID please…

- Please say your name every time you speak
- Also, upload your photo on your profile @ canvas

Remember:

- Students must obtain a passing grade (>50%) in all components of the course in order to pass the class
  - Failing to do so will result in them failing the class :(
- No late assignments will be accepted.
  - An extension will be granted only if a student has a written medical excuse (doctor's note)
- The exam dates are final and there will be no makeup exams
- No email submission of assignments

Java

1. Simple
2. Object-Oriented
3. Network-Savvy
4. Robust
5. Secure
6. Architecture-Neutral
7. Portable
8. Interpreted
9. High-Performance
10. Multithreaded
11. Dynamic

Java Versions

Version	Year	Important New Features
1.0	1996	
1.1	1997	Inner classes
1.2	1998	Swing, Collections framework
1.3	2000	Performance enhancements
1.4	2002	Assertions, XML support
5	2004	Generic classes, enhanced `for` loop, auto-boxing, enumerations, annotations
6	2006	Library improvements
7	2011	Small language changes and library improvements

Java Versions

Features of Java 8 (2014)
Lambda Expression and Virtual Extension Methods
Date and Time API
Nashhorn JavaScript Engine
Improved Security

Becoming Familiar with Your Programming Environment

1. Start the Java development environment.
2. Write a simple program.
3. Run the program.
4. Organize your work.

FirstSample.java


```
1  public class FirstSample
2  {
3     public static void main(String[] args)
4     {
5        // Display a greeting in the console window
6
7        System.out.println("We will not use 'Hello, World!‘ ");
8     }
9  }
```

Analyzing Your First Program: Class Declaration

- Classes are the fundamental building blocks of Java programs:
- Declaration of a class called `FirstSample`

```
public class FirstSample
```

- public is called an *access* modifier
  - these modifiers control the level of access other parts of a program have to this code (more later)
- class reminds you that everything in a Java program *lives inside a class*
  - as a container for the program logic
- In Java, every source file can contain, at most one public class.
- The name of the public class must match the name of the file containing the class:
  - Class `FirstSample` must be contained in a file named `FirstSample.java`

Analyzing Your First Program: Methods

- Each class contains declarations of `methods`.
- Each method contains a sequence of instructions.
- A method contains a collection of programming instructions that describe how to carry out a particular task.
- A method is called by specifying the method and its arguments.

Analyzing Your First Program: `main` Method

- Every Java application contains a class with a `main` method
  - When the application starts, the instructions in the `main` method are executed
- Declaring a `main` method

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

Analyzing Your First Program: Statements

- The *body* of the main method contains **statements**.
- Our method has a single statement:

```
System.out.println("We will not use 'Hello,
  World!'");
```

- It prints a line of text:

```
We will not use 'Hello, World!'
```

Analyzing Your First Program: Method Call

- A method call:

```
System.out.println("We will not use 'Hello,
  World!'");
```

- A method call requires:
  1. The method you want to use (in this case, `System.out.println`)
  2. Any values the method needs to carry out its task enclosed in parentheses (in this case, `"We will not use 'Hello, World!'"`)

- The technical term for such values is arguments

Analyzing Your First Program: Strings

- **String:** a sequence of characters enclosed in double quotation marks:

```
"We will not use 'Hello, World!'"
```

Analyzing Your First Program: Printing

- You can print numerical values

```
System.out.println(3 + 4);
```

  - evaluates the expression 3 + 4
  - displays the number 7.
- `System.out.println` method prints a string or a number and then starts a new line.
  - The sequence of statements

```
System.out.println(“No Hello");
System.out.println("World!");
```

  - Prints two lines

```
No Hello
World!
```

- There is a second method, `System.out.print`, that you can use to print an item without starting a new line

Errors

- A compile-time error (syntax error)
  - is a violation of the programming language rules
  - detected by the compiler.

```
System.ou.println(“Bye bye");
```

- A run-time error (logic error)
  - causes a program to perform an action that the programmer did not intend.

```
System.out.println("Hello, Word!");
```

Errors

- Exception - a type of run-time error
  - Generates an error message from the Java virtual machine
  - This statement

```
System.out.println(1 / 0)
```

  - Generates this run-time error message

```
"Division by zero"
```

Comments


```
 1  /**
 2   * This is the first sample program
 3   * @version 1.01 1997-03-22
 4   * @author Gary Cornell
 5   */
 6  public class FirstSample
 7  {
 8     public static void main(String[] args)
 9     {
10        System.out.println("We will not use
'Hello, World!'"); // or may use it?
11     }
12  }
```