<table>
<thead>
<tr>
<th>Instructor:</th>
<th>Craig Campbell</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office Hours:</td>
<td>Google Hangouts, email, and/or text</td>
</tr>
<tr>
<td>Office Location:</td>
<td>Google Hangouts: <a href="mailto:craig.s.campbell@gmail.com">craig.s.campbell@gmail.com</a></td>
</tr>
<tr>
<td>Phone:</td>
<td>801-499-9947</td>
</tr>
<tr>
<td>E-Mail:</td>
<td><a href="mailto:craigcampbell1@weber.edu">craigcampbell1@weber.edu</a></td>
</tr>
<tr>
<td>Time and Room:</td>
<td>5:30-7:20 MW, Weber State Davis, Rm: 312</td>
</tr>
</tbody>
</table>
| Text: | Core Java, Volume 1, Fundamentals (Required)  
Volume II, Advanced Features 9th ed (Recommended) |
| Authors: | Cay S. Horstmann and Gary Cornell |
| Prerequisites: | CS 2420 (which has prerequisites of CS 1400 [Java] and CS 1410 [C++]) |

| Grading: | Program: 40%  
Midterm: 30%  
Programming Tests: 15%  
Assignments: 15%  
| 92% ≤ total < 100% | A  
| 88% ≤ total < 92% | A-  
| 84% ≤ total < 88% | B+  
| 80% ≤ total < 84% | B  
| 76% ≤ total < 80% | B-  
| 72% ≤ total < 76% | C+  
| 64% ≤ total < 72% | C  
| 0 ≤ total < 64% | E |

| Reading: | Reading assignments are posted on the Canvas Calendar. |
| Programming Labs: | Programming Labs will be creating a single program over the course of the term. It will be turned in during Finals week and a grade will be assessed:  
| Midterms: | 20% Console application  
50% Engine (Core)  
| Programming Tests: | 20% GUI  
10% Tests |
| Assignments: | Assignments and Programming Tests will be small assignments that test your understanding of concepts (much like an employer might ask in an interview question).  
See the Canvas Calendar for dates. |
<p>| Extra Credit | No extra credit beyond that already specified in the assignments is available. |</p>
<table>
<thead>
<tr>
<th><strong>Incomplete Grades:</strong></th>
<th>An “Incomplete” may be given only when the student, having satisfactorily completed approximately 80% of the required work, is unable to complete the class work for a legitimate reason (such as illness or accident) and can reasonably finish on his/her own.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Study Time:</strong></td>
<td>A common “rule of thumb” for allocating study time is 2 to 3 hours of study per credit hour (i.e., 8 - 12 hours per week for CS 3230) in addition to classroom time. This implies that you should plan on 12 to 16 hours of study per week. Please do not overload your schedule.</td>
</tr>
<tr>
<td><strong>Students with Disabilities:</strong></td>
<td>Any student requiring accommodations or services due to a disability must contact Services for Students with Disabilities (SSD) in room 181 of the Student Services Center. SSD can also arrange to provide course materials (including this syllabus) in alternative formats if necessary.</td>
</tr>
<tr>
<td><strong>Course Fees:</strong></td>
<td>Course fees for the Computer Science major are designed to cover the costs of lab equipment maintenance and replacement including desktop and server computer systems and software; consumable materials and supplies; and support for lab aides, student tutors, and online instructional resources.</td>
</tr>
<tr>
<td><strong>Disclaimer:</strong></td>
<td>This syllabus is subject to change at any time. Alterations made on in class or on Canvas supersede old content and become effective immediately.</td>
</tr>
<tr>
<td><strong>Honesty and Fair Use:</strong></td>
<td>Each student is expected to maintain high standards of honesty and ethical behavior. Each assignment must represent the student’s own, best effort. You are encouraged to study together and to work together on the labs. This means that it is okay to discuss algorithms, syntax, and Java in general with others. You may also get ideas and code fragments from books or from the Internet. However, you must at all times write your own code. Specifically, you may not copy whole methods, classes, files, or programs from someone else, from a book or from the Internet; nor may you exchange or share code in any format - including code from previous semesters. <strong>If this is or any other dishonesty is demonstrated, you will fail the course and may face University disciplinary action.</strong> If you have any questions about what is acceptable and what is not acceptable, you may ask the instructor. <strong>Please do not ask students (passed or present) to see copies of their assignments. Please do not give other students (current or future) copies of your assignments.</strong> You are guilty of cheating whenever you submit code that you did not write or that was not explicitly provided to you by the instructor. <strong>CS Department policy dictates that any verifiable evidence of student academic cheating, as defined above, will result in: 1) an automatic failing grade for the class and 2) a report to the Dean of Students that will include the student’s name and a description of the student’s dishonest conduct. In the case of one student providing code or other assessment material to another student, these sanctions will apply to both parties. The university expels students when plagiarism is proven the second time. The University affords you certain rights, including the right to challenge the accusation of cheating. The Dean of Students will explain these rights to you if you are accused of cheating.</strong></td>
</tr>
<tr>
<td><strong>Course Description:</strong></td>
<td>You will learn the Java programming language this semester, which is frequently used for Internet and multimedia programming. Java has, in a few short years, become one of the most popular and most successful programming languages of all time. Its success and growth has paralleled and has been driven by the success and the growth of the Internet. Java is a member of a small, exclusive group of programming languages: the languages that operate over and drive the Internet. Java really is an appropriate topic for an Internet class, a topic that will easily fill an entire semester.</td>
</tr>
</tbody>
</table>
**Instructor’s Goal:** I want to help you learn how to solve problems and how to code the solution in Java. This entails several sub-goals. I want to help you understand: (a) how the object-model works; (b) how to solve problems before you attempt to code the solution in any language; (c) how to debug a program; and (d) how to build appropriate mental models of computer/language systems that will help you understand the tasks on which computer scientists work.

**Course Goals:** The text for the course is Core Java, which is an appropriate description of what you will learn. At the conclusion of the course, you will understand the “core” of the Java language. You will learn in this class those “core” features of the Java language that are used regardless of what area of specialized Java programming you may wish to pursue. This includes programming within the object model and advanced features such as graphics, GUIs, networking, and multithreading. Following are the objectives that we will undertake together as a class this semester.

**Objectives and Outcomes:** At the conclusion of the course you will:

1. Understand the fundamental or core concepts of the Java language; specifically, those parts of the language which are generally used regardless of the problem domain
2. Understand the Object-Oriented model and its relationship to and implementation in the Java programming language; specifically, you will understand the components and concepts:
   a. classes, objects, and instantiation
   b. relationships: inheritance, association, aggregation, composition, and dependency
   c. attributes and methods, and their relationship to encapsulation and abstraction
   d. abstract classes
   e. polymorphism
3. Understand the physical organization of the Java language system and the relationship of this organization with the Java Development Kit (JDK), used to create Java programs:
   a. .java and .class files
   b. the Java Virtual Machine (JVM) and how it relates to applications and to applets
   c. how multi-class programs are assembled into programs (including executable JAR files)
4. Understand and use interfaces and inner classes
5. Understand and use Java’s event delegation model (i.e., be able to create programs based on the event-drive programming model)
6. Be able to write graphical programs
7. Be able to write Graphical User Interface (GUI) programs
   a. AWT
   b. Swing
   c. Applications
   d. Applets
   e. Java’s GUI event interfaces
8. Understand and be able to use event handling, including Java’s event classes
9. Understand and be able to use exception handling
10. Understand and be able to use Java’s input and output philosophy and classes
11. Understand and be able to use Java’s networking classes
12. Understand the concept of multitasking in general and Java’s multithreading features specifically, and be able to write complex, multithreaded programs