

## Course Number, Course Title and Semester Hours (post-assessment May, 2009)

CSC 230 Computer Science II 4sh

### Course Coordinator(s)

Joel Hollingsworth and Dave Powell

### Current Catalog Description

This course continues the study of object-oriented programming with an emphasis on graphical user interfaces, event handling, inheritance, polymorphism, linear data structures, software engineering, recursion and the social context of computing. Prerequisite: CSC 130. Offered fall and spring.

### Textbook

A Comprehensive Introduction to Object-Oriented Programming with Java, ISBN 978-0-07-352339-2, McGraw-Hill, Author Wu/Otani

### References

<a href="#">Pair Programming</a> - Laurie Williams
<a href="#">Eclipse Tutorial</a> - This is an Eclipse Tutorial that I wrote in January 2006 to give a quick intro to Eclipse 3.1. Though we are using Eclipse 3.4 for this course, the tutorial is still applicable. The tutorial has been taken by over 20,000 students and computer professionals throughout the world.
<a href="#">Javadoc Tutorial</a> - This is a link to a pdf file for a simple javadoc tutorial
<a href="#">How to Write Doc Comments for Javadoc Tool</a> – An extremely detailed discussion of Javadoc from Sun Microsystems
<a href="#">Java Documentation Comments</a> – A short but complete discussion of Javadoc
<a href="#">Writing Robust Java Code</a> – Superb, detailed description of coding conventions from a leader in agile programming, Scott Ambler
<a href="#">Coding Style Guide</a> - The company that has written Java and the Java class libraries has established a standard for coding style and conventions.
<a href="#">Android Coding Conventions</a> – Extension of Sun coding conventions for the Android mobile platform
<a href="#">JUnit</a> : JUnit 4 in 10 minutes tutorial. The software comes installed with Eclipse 3.4 for both version 3.8 and 4.0. We will use JUnit 4.0 this semester.
<a href="#">UML Tutorial</a> - Simple introduction to UML class diagrams
<a href="#">The UML</a> – Simple but comprehensive introduction to class diagrams by a company specializing in object oriented programming

### Course Goals

1. Expand the material from CSC 130 to broaden the students' understanding of object oriented software development.

2. Establish the object-oriented principles of inheritance, interfaces and polymorphism.
3. Require a stronger emphasis on problem analysis by students.
4. Understand the Java linear collection classes, hash tables and generics.
5. Establish the concepts of Graphical User Interfaces and Event Driven Programming

### **Prerequisites by Topic**

CSC 130 is a prerequisite for this course. CSC 130 requirements are:

1. PF1: Fundamental programming constructs
2. PF2: Algorithms and problem-solving
3. PF3: Fundamental data structures
4. PL4: Declarations and type
5. PL6: Object-oriented programming
6. SE2: Using APIs
7. SE3: Software tools and environments

### **Major Topics Covered in the Course**

1. PF1. Fundamental programming constructs (2)
2. PF2. Algorithms and problem solving (3)
  - a. Debugging strategies
3. PF3. Fundamental data structures (8)
  - a. Lists, stacks, queues, hash tables
4. PF4. Recursion (5)
5. AL1. Basic algorithm analysis (1 – Big Oh)
6. PF5. Event driven programming (4)
7. PL6. Object oriented programming (10)
8. HC2. Building a simple graphical user interface (2)
9. GV1. Fundamental techniques in graphics (1)
10. SE1. Software design (1)
11. SE2. Using APIs (2)
12. SE3. Software tools and environments (1)
13. SE6. Software validation (1) – Unit testing with JUnit

### **Learning Objective/Outcomes with Cross Reference to CSC Program Outcomes**

1. Understand and apply the three major concepts of object oriented programming: data abstraction, polymorphism and inheritance. (PO 5, 8, 9)
2. Learn and develop a simple graphical user interface using Swing and Android with event programming techniques. (PO 8, 9)
3. Describe the fundamental data structures for linked lists, stacks, queues and hash maps. (PO 5, 8, 10)
4. Learn and apply application programming interfaces from the Java library for Linked lists, stacks, queues and hash maps using generics. (PO 10)
5. Use an interactive development environment. (PO 8)
6. Learn the basics of algorithm analysis with the Big Oh notation. (PO 5)
7. Learn the fundamentals of input and output using the java.io library. (PO 5)

8. Learn and apply recursion. (PO 5)
9. Learn and apply industry standard code documentation conventions, best practices and Javadoc code documentation. (PO 8, 11)
10. Learn and apply unit testing. (PO 8, 11)
11. Use pairwise programming for software development. (PO 2)

**Laboratory projects (specify number of weeks on each)**

There will be 12 - 13 projects/homeworks during the course. The intent is to have at least one homework programming assignment due for each chapter.

1. Using a single java class following coding conventions and encapsulation.
2. Using java packages for namespaces and java doc for user interface descriptions.
3. Exception handling
4. Character and Strings
5. Arrays and Collections
6. Sorting and Searching
7. File I/O both text based and object serialization
8. Interfaces and polymorphism
9. GUI and event driven programming in Swing
10. GUI and event driven programming in Android
11. Recursion
12. Generics
13. Collections, maps and iterators

**Estimate Curriculum Category Content in semester hours. Maximum number of semester hours per course is 4. Times should be in increments of .5. (Advanced is a topic requiring previous core materials – most likely found in a 300 or 400 level class.)**

Category	Core	Advanced
Data Structures	.5	
Algorithms		
Software Design	2.0	
Computer Architecture		
Programming Languages	1.5	

**Oral and Written Communications**

Every student is required to submit at least   0   written reports (not including exams, tests, quizzes, or commented programs) of typically   0   pages and to make   0   oral presentations of typically        minute’s duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

## **Social and Ethical Issues**

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

Pairwise programming is covered and used. Students read an article on what pairwise programming is and the benefits of using it. Students have the mandatory requirement to work in pairs for four assignments and two different partners. Students are required to place a copyright notice on all of their code to indicate that it was created by them.

## **Theoretical Content**

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

- Inheritance – 2 hours
- Polymorphism – 2 hours
- Linked lists and hash maps – 1 hour

## **Problem Analysis**

Students learn the art of problem formulation to select and apply the Java 6.0 language structures to chapter programming assignments listed in laboratory projects section. Each lab provides a description of the problem to solve. The student must design the algorithm, test case and code to solve.

## **Solution Design**

Object oriented design principles are heavily emphasized. Students must encapsulate all classes with particular attention to private data members with getters and setters, a default constructor, a toString method and consideration for an equals method and implementation of Comparable.

Students will become familiar with Javadoc both in writing their own for each class and public method along with using Sun on line documentation for using Sun api's for their class libraries for Collections, IO, swing and awt.

Students will use JUnit 4 testing to validate each major method.

Students will be exposed to GUI layout using xml in Android with the implementation still written in Java.

## **Course Assessment (Pre assessment done on Jan. 20, 2009)**

### ***Changes from last offering in Spring 2008***

There will be three changes to the course for the Spring 2009 semester:

1. The department has chosen a different text to use as the same text in CSC 130 and CSC 230. The text is “A Comprehensive Introduction to Object-Oriented Programming with Java” by Wu. The book was used in the fall 2008 CSC 130 class and will be used this spring for the first time for CSC 230. The book covers the same topics as the previous text, Absolute Java.
2. The Android platform was introduced with a SDK and Eclipse plug-in in October 2008. Android uses the emerging concept of specifying a GUI layout in XML and the exception handling in Java. After Swing is covered then simple GUI creation in Android will be introduced.
3. The enrollment in CSC 230 is 30 students. This is twice the normal amount. Pairwise programming will be used for all assignments so that we can still have one homework assignment a week with a quick turnaround on grading.

### **Learning outcomes assessed, how they were assessed and success criteria**

The focus this semester will be on assessing learning outcome 2 and 11. Both outcomes will be assessed from two homework assignments. I will assign the same homework for the students to program a Calculator in Swing and then in Android. The Swing implementation will be done in pure java and the Android implementation will be done in xml for presentation and java for event handling.

My goal is a class average of 70% on each outcome.

### **Assessment data and analysis (Final assessment to be completed at end of semester)**

Course Outcome 2: Learn and develop a simple graphical user interface using Swing and Android with event programming techniques. Figure 1 shows the student grade distribution on Homework 9 where pairwise teams developed a Swing calculator. The grades were outstanding. All fourteen teams received an A- or A. The students really enjoyed the development of a GUI. Figure 2 shows the student grade distribution on Homework 10 where the same pairwise teams developed an Android calculator. The grades were again outstanding. Thirteen of the 14 teams received an A- or A. The fourteenth team had one student not participate so the one student who did not do the work received a grade of zero. The other team member did the entire project and received a 92. The students really enjoyed the novelty of Android and quickly picked up the use of XML for Linear Layouts and Table Layouts. The students easily exceeded the outcome goal of 70. This was the best CSC 230 class that I have had in the past 8 years. The students were sharp, enthusiastic and worked well in teams. The three major changes that I made to Swing this semester were:

1. I required a class exercise to be done before each class on an aspect of Swing covered in the previous class.
2. I used inner classes from the very beginning.
3. I spent two additional classes on Swing.

Students were excited about using the latest in smart phone technology. I gave an anonymous survey and one question was used to solicit the students’ enthusiasm for using android. The results are shown in Figure 3 and 59% were strongly enthusiastic to use Android. I will use it again next semester and also use it in the follow on course for CSC 330.

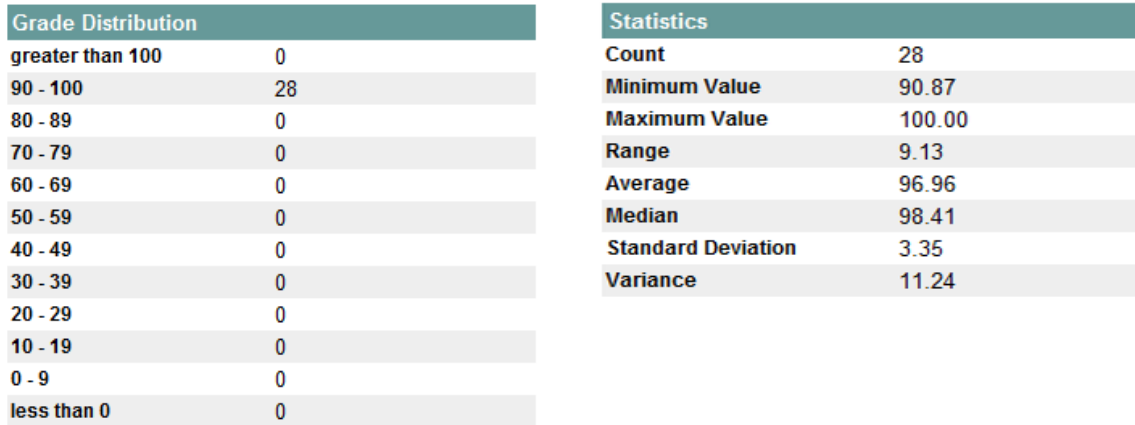


Figure 1: Swing pairwise grades on Homework 9.

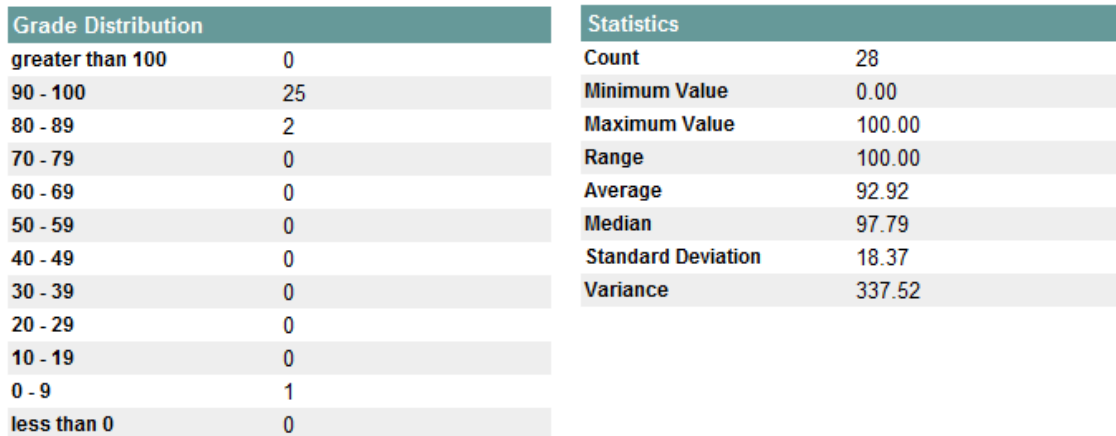


Figure 2: Android pairwise grades on Homework 10

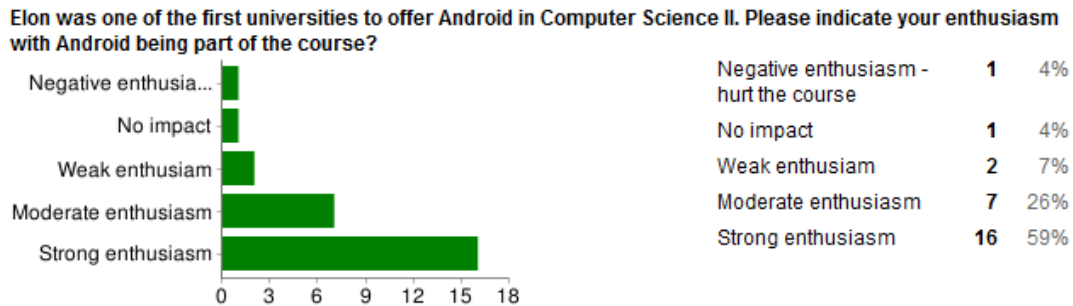


Figure 3: Course anonymous survey result on Android use

Course outcome 10: Learn and apply unit testing

The students were required to use JUnit testing on Homework 4 to verify that the string Regexp matching was working properly on their palindrome code. The students had to write 11 individual test cases along with providing a Before and After Fixture. Thirty six points on the homework which was worth 111 points were related to testing. The average grade on the testing portion was 81.48. The students did very well and easily met the goal of 70. This is the first time that I used JUnit 4 with annotations instead of JUnit 3 and the students picked it up quickly. No changes are recommended for the next offering.

**Proposed changes for next offering (Final assessment to be completed at end of semester)**

CSC 230 has been continuously and incrementally improved each year. The course is pretty much perfect. We have evolved the course to shift more and more to GUI development and the introduction to data structures using the Java APIs instead of developing your own data structures which is in our opinion a carryover from the old days when libraries were not readily available. The larger use of GUI is exciting to the students and provides a strong ending to the last third of the semester as the first 2/3rds of the semester are teaching the concepts of abstraction, inheritance and polymorphism which are needed to develop GUIs.

The Wu/Otani book proved to be marginal. A large part of the book was dedicated to discussion of examples and of not much use to learning the concepts. The department has decided to return to Absolute Java for next year.