

# CS 120: Computer Science II

## Marshall University, Spring 2005

### Course Description

Object-oriented analysis and design, advanced programming with classes, arrays, strings, sorting, searching, I/O, GUI development, system life cycle and software development methodologies.

### Instructors Information

**Team taught** by professors Venkat N Gudivada (gudivada@marshall.edu) and Joe Fuller (fullerj@marshall.edu).

**Phone** Gudivada: 304-696-5452; Fuller: 304-696-6204.

**Office Location:** Gudivada: Gullickson Hall, Room 205; Fuller: Gullickson Hall, Room 115.

**Office Hours:** Gudivada: 12.30 PM - 4.00 PM Mondays and Thursdays; 12.30 PM - 2.00 PM Tuesdays. Other times by appointment.

Fuller:

### Course Topics

1. Advanced programming with classes: inheritance and polymorphism.
2. Arrays, sorting, searching, and strings.
3. Directory and file creation and manipulation, and IO streams.
4. Object-Oriented Analysis and Design (OOAD).
5. Event-driven programming and GUI design.
6. Core interfaces in an industry standard class library.
7. Software-intensive systems and their complexity management.
8. System development life cycle and software development methodologies.

## Course Objectives

1. Provide a conceptual introduction to Object-Oriented Analysis and Design (OOAD) techniques and artifacts (Week 1).
2. Introduce advanced programming with classes: inheritance and polymorphism (Weeks 2 and 3).
3. Introduce array concepts and illustrate programming with the array data structure (Week 4).
4. Introduce sorting concepts and explain basic sorting algorithms (Week 5).
5. Introduce searching concepts and explain basic searching algorithms (Week 6).
6. Introduce string as a data structure, explain string operations, and discuss pattern matching and regular expressions (Week 7).
7. Explain input/output (I/O) concepts, illustrate directory creation and navigation, file creation and manipulation, and reading from and writing to files (Week 8).
8. Introduce user interface concepts and event-driven programming paradigm. Illustrate programming GUI applications by using various control and attaching event handlers to them (Week 9).
9. Introduce the notions of interface and interface reference, and illustrate interface implementation. Discuss core interfaces in a class library (Week 10).
10. Explore basic data structures — Stack, Queue, and List — their operations and implementation alternatives. Examine interfaces related to these data structures in a class library (Weeks 11 and 12).
11. Introduce advanced programming concepts — delegates, events, runtime type discovery, generics, object-persistence and serialization, and threads (Weeks 13 and 14).
12. Enumerate software-intensive system characteristics and principles for managing their complexity (Week 15).
13. Introduce System Development Life Cycle (SDLC) and software development methodologies (Week 15).
14. Provide a conceptual introduction to software design patterns (Week 15).
15. Discuss Unified Process (UP) and its variant Rational Unified Process (RUP) as a software development methodology (Week 15).

## Instructional Methods

Classroom lectures, hands-on exploration and experimentation, guided classroom discussions, and collaborative problem solving.

## Evaluation Methods

The course is evaluated in terms of measurable student learning outcomes. Grade assignment is based how many learning outcomes the student has demonstrated and to what degree.

### Measurable Student Learning Outcomes

A high course grade in CS 120: Computer Science II requires that the student demonstrate most or all of the following:

1. **Demonstrates** a conceptual understanding of Object-Oriented Analysis and Design (OOAD) techniques.
2. **Understands** and uses inheritance and polymorphism features in problem solving.
3. **Demonstrates** understanding of array as a primary data structure for indexed access to a list of elements; **gained proficiency** in programming with arrays.
4. **Demonstrates** understanding of sorting algorithms and their implementation.
5. **Demonstrates** understanding of searching algorithms and their implementation.
6. **Demonstrates** understanding of string concepts, operations on strings, pattern matching and regular expressions; **gained proficiency** in programming with strings.
7. **Explains** file system concepts; **gained** programming proficiency in creation and manipulation of files and directories as well as reading from and writing to files.
8. **Explains** the concept of interfaces and implements interfaces; **understands** functions of core interfaces in an industry standard class library.
9. **Demonstrates** understanding of event-driven programming paradigm; **develops** simple GUI applications.
10. **Demonstrates** understanding of recursion both conceptually and programmatically.
11. **Demonstrates** conceptual understanding of stack, queue, and list data structures including their operations; **understands** trade offs involved in their implementation; **develops** programs which require these data structures.
12. **Understands** how to make use of interfaces related to stack, queue, and list data structures in an industry standard class library.
13. **Gained** conceptual understanding of advanced programming concepts including delegates, events, runtime type discovery, generics, object-persistence and serialization, and threads; **understands** programs that employ these advanced concepts.

14. **Explains** the characteristics of software-intensive systems and **understands** principles for managing their complexity.
15. **Enumerates** phases in the System Development Life Cycle (SDLC) and **recognizes** the appropriateness of a software development methodology for a given project.
16. **Demonstrates** knowledge of workflows and phases in the Unified Process (UP).
17. **Recognizes** the use and importance of software system architecture.
18. **Understands** the role of design patterns in Object-Oriented Analysis and Design (OOAD) tasks.

## Course Assessment

The course assessment components include: quizzes (10%), programming assignments (25%), two exams (40%), and a final (25%). Maximum possible score is 100. Course grade is awarded based on the following scheme:

<i>Score</i>	<i>Letter Grade</i>
> 90	A
> 80 & < 90	B
> 70 & < 80	C
< 70	F

## Instructional Materials

**Required Textbook** Cay Horstman, *Computing Concepts with Java Essentials* (3<sup>rd</sup> edition), ISBN: 0-471-24371-x, John Wiley, 2003.

**Additional Resources** Course notes and other handouts will be available on the course WebCT Vista. URLs for additional resources will also be listed on the WebCT.

## Course WebCT Vista

It is important to visit the course WebCT for up-to-date information about the course. It hosts all the course materials including quizzes, handouts, lecture notes, and reading materials. Also, you will use the Vista for submitting your quizzes and programming projects.