

CS 110: Computer Science I

Marshall University, Fall 2004

Course Description

Object-oriented and algorithmic problem solving principles and techniques; programming with classes in an integrated programming environment; and program debugging.

Instructor Information

Name: Dr. Venkat N. Gudivada.

Phone and Email: 304-696-5452; gudivada@marshall.edu.

Office Location: Gullickson Hall, Room 205.

Office Hours: MW 2.00 PM – 4.00 PM; TR 3.30 PM – 5.00 PM. Other times by appointment.

Course Topics

1. Computer Science as an academic discipline and its practice as a profession.
2. Types of computers, categories of software, programming paradigms, and abstract machine hierarchy.
3. A Conceptual introduction to Object-Oriented (OO) problem solving and UML.
4. Programming with classes.
5. Syntax, logical and runtime Errors; exception processing and debugging.
6. Exploring and using class libraries.
7. Exploring and using integrated programming environments.
8. Algorithmic problem solving, method implementation, and parameter passing.
9. structures and enumerations.

Course Objectives

1. Introduce panoramic view of Computer Science as an academic discipline and its practice as a profession (week 1).
2. Introduce the societal context and social responsibility in which the profession is practiced, and the code of conduct governing the professional practice (week 1).
3. Explain types of computers and categories of software (Week 1).
4. Explain the notion of abstract machine hierarchy and virtual machines (week 1).
5. Discuss various programming paradigms (week 2).
6. Provide a conceptual introduction to Object-Oriented (OO) problem solving concepts and UML (week 2).
7. Introduce the notion of variables, value and reference types, operators, and expression evaluation (week 3).
8. Illustrate preliminary programming with classes and establish coding standards (weeks 4 and 5).
9. Explain syntax and logical errors, runtime errors and exception processing (week 5).
10. Preview arrays to facilitate discussion of control structures (week 6).
11. Introduce the three fundamental control structures — sequence, selection, and iteration — and illustrate programming using the control structures (weeks 6 and 7).
12. Discuss and illustrate strategies for debugging programs (week 8).
13. Discuss the notion of class libraries, and their exploration and usage (week 8).
14. Illustrate programming in an Integrated Development Environment (IDE). Discuss components of the IDE and its utilities, and demonstrate debugging with a GUI Debugger (week 9).
15. Introduce algorithmic problem solving concepts — the notion of an algorithm and its specification, and characteristics of good algorithms (week 10).
16. Explain declaring and implementing class methods. Illustrate method parameter passing techniques and method overloading (weeks 10 and 11).
17. Explain type casting (week 12).
18. Illustrate implementing relationships between classes (weeks 12, 13, 14).
19. Introduce structures as light-weight classes (week 15).
20. Introduce Enumeration type and illustrate programming with this type (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 110: Computer Science I requires that the student demonstrate most or all of the following:

1. **Understands** the facets of Computer Science and their relationship to each other.
2. **Recognizes** that there are ethical and social issues related to the practice of the profession and that the practice is governed by a code of professional conduct.
3. **Understands** the notions of abstract machine hierarchy and virtual machines, and their practical significance.
4. **Articulates** the various programming paradigms and the contexts in which they are suitable.
5. **Defines** with ease the terminology related to the object-oriented computing paradigm.
6. Given a simple domain description, **identifies** conceptual classes, their high-level attributes and services, and relationships between the classes.
7. **Understands** the significance of UML at a conceptual level, especially the class diagrams.
8. **Explains** the differences between the value and reference types — when one is preferred over the other, memory allocation, garbage collection, aliasing, and dangling references.
9. **Demonstrates competence** in basic programming with classes — understands syntax and semantics related to namespaces, classes and their components.
10. **Explains** various operators available in the programming language, understands the notions of operator precedence and associativity, and their significance in expression evaluation.
11. **Evaluates** various types of expressions and **distinguishes between** relational, boolean, arithmetic, and compound expressions.
12. **Recognizes** the distinction between syntax, logical, and runtime errors.

13. **Gained clear understanding** of the syntax and semantics of the three fundamental control structures; **demonstrates** how to choose the right control structure for a given context.
14. **Understands** the design philosophy behind class libraries, their components and function.
15. **Acquired** a working knowledge of important namespaces in a class library and relevant classes in them.
16. **Acquired** a working knowledge of an integrated programming environment; and **effectively** uses online documentation.
17. **Debugs** programs using a command line or GUI debugger.
18. **Defines** what an algorithm is and the means to specifying them; **articulates** the characteristics of a good algorithm.
19. **Demonstrates** understanding of various parameter passing techniques.
20. **Demonstrates** understanding of type casting concepts and **uses** them in programs.
21. **Demonstrates** programming techniques for implementing relationships between classes.
22. **Explains** differences between structures and classes and **programs** using structures.
23. **Recognizes** situations that require enumeration types and **uses** them programmatically.

Course Assessment

The course assessment components include: homeworks (10%), programming assignments (30%), two exams (40%), and a final (20%). 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* (3rd 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 all the up to date information about the course. It hosts all the course materials including homeworks, handouts, lecture notes, and reading materials. Also, you will use the Vista for submitting your homework and programming projects.