CSC 201J  Software Design and Programming  I

Catalog description:

This course introduces a set of fundamental design principles and problem-solving techniques for the development of computer algorithms and their implementation as programs. Problem solutions are developed with the help of an appropriate modeling language and then coded in an object-oriented programming language. (Consult the Computer Science Department for the languages and tools currently in use.) Topics such as problem specification, object-oriented analysis and design, standard data types, control structures, methods and parameter passing, and design for reuse are presented through a study of specific example problems and solutions. Style, documentation, solution robustness, and conformance with specifications are emphasized throughout. Three lecture hours and three hours of scheduled laboratory per week plus extensive programming work outside of class.

Prerequisites: High school algebra I & II; experience with a window-based operating system and the use of email and a word processor.

Goals:

The purpose of this course is to develop students’ understanding of a coherent set of tools and techniques for creating computer solutions to simple problems in data manipulation. Upon completion of the course, a student should be able to do the following:

CG01: analyze a problem statement for completeness and clarity;
CG02: use the methodology of object-oriented design to develop class diagrams (data descriptions and methods) for a problem solution;
CG03: convert this solution into source code in the designated high-level programming language in accordance with a well-defined set of style rules;
CG04: debug and test the program;
CG05: provide clear documentation for the result.

Objectives:

Upon successful completion of the course, a student will have:

CO01: demonstrated knowledge of the syntax elements of an object-oriented programming language;
CO02: gained experience in analyzing problem statements for completeness and consistency;
CO03: practiced standard techniques of problem analysis;
CO04: applied the fundamentals of object-oriented design methodology;
CO05: learned and utilized simple techniques for validation and verification of programs;
CO06: created full documentation for several completed projects.

<table>
<thead>
<tr>
<th></th>
<th>PO1</th>
<th>PO2</th>
<th>PO3</th>
<th>PO4</th>
<th>PO5</th>
<th>PO6</th>
<th>PO7</th>
<th>PO8</th>
<th>PO9</th>
<th>PO10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO01</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO02</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO03</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO04</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO05</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

Topics (using Java and UML):

- the “problem-solving universe”
  - operational definition of computer (specifically, electronic stored-program digital computer)
  - components of a typical computer
  - fundamental computer capabilities
- strategies and tools for problem-solving
  - formulating precise specifications for a problem and its solution
  - algorithms
  - modular design of user requirements in measurable units
  - preconditions and postconditions
  - specification of user requirements in measurable terms
• programming languages and programming language paradigms
  PL1(0.5)
• brief history and overview of the Java language
  PL2(0.5), PL3(0.5)
• data types
  ° basic data types: integer, real, character, boolean
  PL4(1.0), PF3(1.5)
  ° literals of each type
• variables and constants
  PF1(2)
• reference types (including String and pre-defined wrapper classes)
  PL4(1.0), PF3(2)
• console output
  PF1(0.5)
• console input
  PF1(0.5)
• simplified graphical user interfaces (GUIs)
  ° JOptionPane
  HC1(1)
• object-centered problem analysis
  PL6(0.5), SE1(0.5)
• object-centered design and implementation
  PL6(0.5), SE1(0.5)
• introduction to UML (Unified Modeling Language)
  SE3(0.5)
• classes
  ° overview
  PF3(1), PL5(2), PL6(1.5), SE2(0.5)
  ° attributes
  ° methods
    • parameter and argument lists, return values, signatures
    • use of modular design in creating methods
    • visibility rules (scope, context)
  ° objects (instances)
    • handles
    • copying objects (shallow vs. deep copies, clones)
• selection control structures
  PF1(1.5)
• testing and verification
  SE6(0.5)
• debugging
  PF2(1), SE2(0.5)
• repetition control structures (loops)
  ° while and do while structures
  ° for structure
  PF1(1), PL4(0.5)
• object-oriented program design techniques
  PL6(1), SE1(1), SE7(0.5)
• collections (conceptual discussion)
  PF2(0.5)
• arrays of one dimension
  PF2(1), PF3(1)
  ° syntax rules
  ° static nature of arrays; physical vs. logical size of an array
  ° common algorithms:
    • storing a value in an array
    • removing a value from an array
    • linear traversal
    • linear search

Note: Programming language features, techniques, and aspects of object-orientation not mentioned above (for example: sorting, binary search, multidimensional arrays, stream and file I/O, inheritance, derivation) are normally covered in the second course in the sequence, CSC202J.

Programming assignments: Ten to twelve programming assignments are given. One or more of these may be group projects. Each programming assignment normally involves the design, writing, testing and debugging of a program and the submission of an appropriate laboratory report. Each assignment has a specific due date, with a short grace period during which the assignment may be submitted for reduced credit. When the grace period has expired, the assignment will no longer be accepted.
All programs must be coded in the programming language currently used for instruction in the CSC201J/202J sequence; no exceptions will be allowed. The version of the language being used will be the currently accepted standard version: any extensions or variations in student-owned compilers must be approved in advance by the instructor, who may choose to forbid their use.

**Laboratory exercises:** There will be short programming exercises to be completed during weekly scheduled laboratory sessions. Each exercise focuses on a specific language feature or programming technique presented in recent lectures. Performance on these exercises will be incorporated into the course grade.

**Exams and quizzes:** There will be two examinations, two shorter quizzes, and a comprehensive written two-hour final examination.

**Grading:** Final grades will be determined on the basis of the following approximate weights: examinations and quizzes - 40%, programming assignments and lab exercises - 60%.

<table>
<thead>
<tr>
<th></th>
<th>Test / Quiz Questions</th>
<th>Homework Problems</th>
<th>Programming Projects</th>
<th>Lab Exercises</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO01</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>CO02</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CO03</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>CO04</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>CO05</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>CO06</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bibliography:**


