COURSE INFORMATION FORM

DISCIPLINE: Computer Science/Information Systems

COURSE TITLE: Object-Oriented Programming

CR.HR: 3  LECT HR: 2  LAB HR: 2  CLIN/INTERN HR: 0  CLOCK HR: 0

CATALOG DESCRIPTION
Introduction to object-oriented programming for students with procedural programming background. Data encapsulation, information hiding, built-in classes and libraries, inheritance, polymorphism, simple graphical user interfaces, user-defined classes and event-driven programming. Basic object-oriented design, maintainable software, software reuse, class hierarchies, design patterns and Universal Modeling Language. Uses object-oriented language.

PREREQUISITES
MATH 110 Intermediate Algebra or equivalent placement test score
CSIS 123 Programming Fundamentals

EXPECTED STUDENT OUTCOMES IN THE COURSE (ESO)
Upon completion of this course, the student will be able to:

1. Use the terminology associated with object-oriented design and programming.
2. Use built-in classes and libraries.
3. Design classes and class hierarchies based upon software engineering principles.
4. Code classes, sub-classes, driver programs and applications.
5. Design and implement a simple graphical user interface.
6. Apply tools to support program development, testing and debugging.
7. Code and document programs following a guideline.
8. Interpret class-relationship and use-case diagrams in UML

GENERAL EDUCATION OUTCOMES (ESO)
Specify which general education outcomes, if any, are substantially addressed by the course. Numbers in parentheses identify the Expected Student Outcomes linked to the specific General Education Outcome.

<table>
<thead>
<tr>
<th>Outcomes</th>
<th>ESO</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Critical Thinking</td>
<td>2.</td>
</tr>
<tr>
<td>B. Define, analyze, and evaluate information, materials and data</td>
<td>B.</td>
</tr>
<tr>
<td>3. Unambiguously define problems with issues</td>
<td>3,5</td>
</tr>
<tr>
<td>4. Integrate information and see relevant relationships that broaden and deepen understanding</td>
<td>3,4,5</td>
</tr>
</tbody>
</table>
PROGRAM-LEVEL OUTCOMES

CAREER AND TECHNICAL EDUCATION PROGRAM OUTCOMES
Specify which Career and Technical program outcomes, if any, are substantially addressed by the course by completing the “Career and Technical Education template” to show the relationship between course and program outcomes to assessment measures.

1. Use industry specific software and/or apply troubleshooting skills to solve problems.
2. Create and define solutions to real life business challenges.
3. Recognize the need for continued professional development.

CLASS-LEVEL ASSESSMENT MEASURES
Student accomplishment of expected student outcomes will be assessed using the following measures. (Identify which measures are used to assess which outcomes.)

Exercises (1-8)
Individual or group design projects (1-8)
Individual or group programming problems (1-8)
Quizzes and examinations (1-8)
Written/Oral reports (1, 8)
Individual instructors may order this outline as fits the needs of their individual courses. In addition, they may place more emphasis on some areas than on others. What is assured is that this particular list is covered in the course. Other topics may be added to a course as the instructor sees fit, and as time and interest allow. An *asterisk can be used to mark an item as optional.

I. Review of Procedural Programming
   A. Using an integrated development environment to enter, debug and test programs
   B. Compiled versus interpreted
   C. Primitive data types
   D. Basic input/output operations
   E. Control structures coding
   F. Functions (Methods)
   G. Call-by-value versus call-by-reference
   H. Scope and duration of variables and constants
   I. Primary data structures of strings, single-dimensioned arrays and records
   J. Sequential file handling
   K. Principles of structured, modular design
   L. Coding style techniques

II. Object-Based Programming
   A. References and pointers
   B. Class mechanisms that support encapsulation and information-hiding
   C. Class components
   D. Separation of data elements and methods
   E. Access specifiers
   F. Constructor methods
   G. Destructor methods
   H. Overloading methods
   I. Built-in classes
   J. User-defined classes
   K. Instantiation of a class object
   L. Runtime storage management

III. Object-Based Design
    A. Information-hiding and encapsulation
    B. Abstract data type
    C. Attributes and behaviors of a class
    D. Has-a relationships
    E. Class diagram in Universal Modeling Language (UML)

IV. Object-Oriented Programming
    A. Class mechanisms that support inheritance and multiple inheritance
    B. Class mechanisms that support polymorphism
    C. Implementation of friendly or package access
    D. Built-in class hierarchies
    E. User-defined class hierarchies
    F. Separation of class definitions into multiple files within a project
    G. *Collection classes and iteration protocols
    H. *Internal representations of objects and method tables

V. Object-Oriented Design
    A. Is-a relationships
B. Is-a versus has-a relationships  
C. Single inheritance versus multiple inheritance  
D. Class hierarchies  
E. Overriding methods  
F. Friendly or package access  
G. Polymorphism  
H. *Class design using Class, Responsibilities, Corroboration (CRC) cards  
I. Class relationship diagrams in UML  
J. Use-case diagrams in UML  

VI. Simple Graphical User Interfaces (GUI)  
A. Use of simple graphic components to build an interface  
B. Human-centered development  
C. Principles of good screen design  
D. Component-based programming in an API environment  
E. Class browsers and related tools  
F. Event-driven programming versus command-line programming  
G. Label  
H. Text field  
I. Push button  
J. *Dialog box  
K. *Image and icon  
L. *Check box  
M. *Radio button  
N. *Drop-down list  

VII. Simple Graphics API  
A. Graphics Applications Programming Interface  
B. 2D versus 3D  
C. Coordinate system  
D. Graphic objects  
E. Draw lines  
F. Draw rectangles  
G. Draw ovals  
H. Color controls  
I. Fonts controls  

VIII. Fundamental Data Structures  
A. Strings and string processing  
B. Multiple-dimensional arrays with searching and sorting algorithms  
C. *Introduction to collections/templates  

IX. Software Engineering Principles  
A. System requirements and specifications  
B. Functional and nonfunctional requirements  
C. Composition versus Inheritance  
D. Maintainable software  
E. Design for reuse  
F. Top-down versus bottom-up design  
G. Object-oriented analysis and design  
H. Basic concepts of formal specification techniques  
I. Software validation  
J. Prototyping  
K. Software evolution  
L. Design patterns  
M. Model, Viewer, Control (MVC) design pattern example  
N. *Component-level design  

X. Fundamental Computing Algorithms  
A. Simple numerical algorithms  
B. Sequential and binary search  
C. Selection and insertion sorting algorithms
D. Quicksort, heapsort and mergesort algorithms

XI. Introduction to Database Systems
   A. History and motivation for database systems
   B. Components of database systems
   C. Database architecture and data independence
   D. Database Management System functions
   E. Basic use of a database query language to retrieve information

XII. Event-Driven Programming
    A. Events in a GUI program
    B. Event-handling methods
    C. Event propagation
    D. Exception handling