The University of Texas at Tyler  
Bachelor of Science in Computer Science

Syllabus

<table>
<thead>
<tr>
<th>Course Number:</th>
<th>COSC 1337</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Title:</td>
<td>The Object-Oriented Paradigm</td>
</tr>
</tbody>
</table>

Course Description:
Introduces the concepts of object-oriented programming to students with a background in the procedural paradigm. Topics covered include a review of control structures and data types, the object-oriented programming paradigm, object-oriented design, an overview of programming language principles, simple analysis of algorithms, basic searching and sorting techniques, and an introduction to software engineering issues.

<table>
<thead>
<tr>
<th>Pre-requisites:</th>
<th>COSC 1336/1136</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credits:</td>
<td>3 Lecture hours</td>
</tr>
<tr>
<td>Text(s):</td>
<td>Horstman, <em>Big Java</em>, Wiley Publishing</td>
</tr>
</tbody>
</table>

Languages Used:
Java

Topics:
1. Review of object-oriented programming: Object-oriented methodology, object oriented design; software tools.
2. Principles of object-oriented programming: Inheritance; class hierarchies; polymorphism; abstract and interface classes; container/collection classes and iterators
3. Object-oriented design: Concept of design patterns and the use of APIs; modeling tools such as class diagrams, CRC cards, and UML use cases.
4. Virtual machines: The concept of a virtual machine; hierarchy of virtual machines; intermediate languages
5. Fundamental computing algorithms: Searching; sorting; introduction to recursive algorithms
6. Fundamental data structures: Built-in, programmer-created, and dynamic data structures
7. Event-driven programming: Event-handling methods; event propagation; exception handling
8. Foundations of human-computer interaction: Human-centered development and evaluation; principles of good design and good designers; engineering tradeoffs; introduction to usability testing
9. Fundamental techniques in graphics: Hierarchy of graphics software; using a graphics API; simple color models; homogeneous coordinates; affine transformations; viewing transformation; clipping
10. Software engineering issues: Tools; processes; requirements; design and testing; design for reuse; risks and liabilities of computer-based systems

Additional Materials:
COSC 1137 Object-Oriented Paradigm Lab complements this course.

Evaluation Method: (only items in dark print apply)
1. Examination/Quiz  
2. Homework
3. Paper/Report  
4. Computer Program
5. Project  
6. Presentation
7. Class Participation  
8. Peer Review
Course Objectives: By the end of this course students are expected to:

1. Choose the appropriate data structure for modeling a given problem. [1, 2, 4, 5, 6]
2. Determine when a recursive solution is appropriate for a problem. [1, 4]
3. Design, code, test, and debug simple event-driven programs that respond to user events. [5, 6]
4. Develop code that responds to exception conditions raised during execution. [4]
5. Describe the importance and power of abstraction in the context of virtual machines. [1, 2]
6. Demonstrate different forms of binding, visibility, scoping, and lifetime management. [1, 2, 4, 5, 6]
7. Defend the importance of abstractions, especially with respect to programming-in-the-large. [4, 5, 6]
9. Design, implement, and test the implementation of “is-a” relationships among objects using a class hierarchy and inheritance. [4, 5]
10. Compare and contrast the notions of overloading and overriding methods in an object-oriented language. [1, 2, 4, 5, 6]
11. Explain the relationship between the static structure of the class and the dynamic structure of the instances of the class. [1, 4, 5]
12. Describe how iterators access the elements of a container. [1, 2, 4, 5]
13. Describe how the class mechanism supports encapsulation and information hiding. [1, 2]

Numbers in bracket refer to method(s) used to evaluate the course objective.

Relationship to Program Outcomes: (only items in dark print apply)²
This course supports the following Computer Science Program Outcomes, which state that our students at the time of graduation are expected to:

1. Posses knowledge of the fundamentals of mathematics, science, and technology.
2. Be able to use modern computational tools and techniques in the practice of computer science.
3. Be able to develop logically sound and efficient algorithms.
4. Be prepared to implement algorithms in multiple programming languages, on multiple hardware platforms, and in multiple operating system environments.
5. Be able to perform analysis, design, implementation, testing, and maintenance of computer-based systems, stressing software engineering principles.
6. Be prepared to seek continuing professional development, graduate studies, or professional certifications related to computer science.
7. Demonstrate effective written, visual and oral communication skills.
8. Posses an educational background to understand the global context in which computer science is practiced, including:
   a. Knowledge of contemporary issues related to computer science;
   b. The impact of computers on society;
   c. The role of ethics in the practice of computer science.
9. Be able to contribute effectively as members of a project development team.
10. Recognize the need to pursue continued learning throughout their professional careers.

Numbers in brackets refer to course objective(s) that address the Program Outcome.

Prepared By: Kay Pleasant
Date: 10/11/2004
Revised: