LSC Use Only Proposal No: LSC Action-Date:	*UWUCC Use Only* Proposal No: 11-124h. UWUCC Action-Date: App 4/3/12 Senate Action Date: App 4-17-12

Curriculum Proposal Cover Sheet - University-Wide Undergraduate Curriculum Committee

Contact Person(s) David T. Smith	Email Address dtsmith@iup.edu
Proposing Department/Unit Computer Science	Phone 7-4478

Check all appropriate lines and complete all information. Use a separate cover sheet for each course proposal and/or program proposal.

1. Course Proposals (check all that apply)

☐ New Course ☐ Course Prefix Change ☐ Course Deletion
☑ Course Revision ☐ Course Number and/or Title Change ☑ Catalog Description Change

Current course prefix, number and full title: COSC 210 Object Oriented and GUI Programming

Proposed course prefix, number and full title, if changing:

2. Liberal Studies Course Designations, as appropriate
This course is also proposed as a Liberal Studies Course (please mark the appropriate categories below)

☐ Learning Skills ☐ Knowledge Area ☐ Global and Multicultural Awareness ☐ Writing Across the Curriculum (W Course)

☐ Liberal Studies Elective (please mark the designation(s) that applies – must meet at least one)

☐ Global Citizenship ☐ Information Literacy ☐ Oral Communication
☐ Quantitative Reasoning ☐ Scientific Literacy ☐ Technological Literacy

3. Other Designations, as appropriate

☐ Honors College Course ☐ Other: (e.g. Women's Studies, Pan African)

4. Program Proposals

☐ Catalog Description Change ☐ Program Revision ☐ Program Title Change ☐ New Track
☐ New Degree Program ☐ New Minor Program ☐ Liberal Studies Requirement Changes ☐ Other

Current program name:

Proposed program name, if changing:

5. Approvals	Signature	Date
Department Curriculum Committee Chair(s)	[illegible]	2/6/12
Department Chairperson(s)	[illegible]	2/10/2012
College Curriculum Committee Chair	[illegible]	3/7/12
College Dean	[illegible]	3/12/12
Director of Liberal Studies (as needed)		
Director of Honors College (as needed)		
Provost (as needed)		
Additional signature (with title) as appropriate		
UWUCC Co-Chairs	Gail S Sechrist	4/3/12

Course Revision –Computer Science Curriculum

Part II. Description of Curriculum Change

1. New Syllabus of Record

I. Course Description

COSC 210 Object Oriented and GUI Programming 3c-01-3cr

Prerequisite: COSC 108 or COSC 110

An in-depth introduction to the Object Oriented Programming (OOP) paradigm, including encapsulation, inheritance, and polymorphism. The focus will be on designing, implementing, and using objects. This course will also include an introduction to Graphical User Interface (GUI) design and programming.

II. Course Outcomes

Upon successful completion of the course, the student will be able to:

1. Demonstrate an understanding of the fundamental concepts of the OOP paradigm including encapsulation, inheritance, polymorphism, interfaces, and overloading/overriding.
2. Demonstrate an understanding of the fundamental concepts of GUI programming.
3. Design and implement small applications (e.g. involve around a dozen source files) using an object oriented programming language. A few applications will use a GUI interface.
4. Demonstrate an understanding of object oriented programming syntax and semantics including; class definitions, interface definitions, IO library, GUI library, collections library, and exception handling.
5. Implement programs according to standard (i.e. industry accepted) programming practices.
6. Use fundamental object oriented data structures (e.g. arrays of objects, lists, and iterators) within an application.
7. Test and debug object oriented programs using an interactive development environment (IDE).
8. Interpret the design of a set of classes using a subset of Unified Modeling Language (UML)

III. Course Outline:

1. Review of Programming Fundamentals and the Stage for Object Oriented 3 hours
   - What is a computer, computer organization, and history of programming languages
   - Programming basics; statements, data, variables, arrays, functions, calculations, etc.
   - Object oriented programming, what and why
   - History of object oriented languages
   - Compilation and execution
   - Programming conventions part 1

2. Object Basics 3 hours
   - Defining classes; instance variables and methods
   - Primitive data types and Strings (intro)
   - Object instantiation, object variables, state, and purpose of methods
   - Encapsulation, information hiding, set-*er* and get-*er* methods
   - Initializing objects; constructors
   - Integrated development environment and debugging
   - UML class diagram basics.
   - Programming conventions part 2

3. Control Statements — 3 hours
   - Algorithms
   - if, then, while, switch, break, continue, and for
   - Expressions and assignments re-visited; operators, precedence.
   - Data type handling, object vs. primitive, identity vs. equality
   - Exception handling
   - Programming conventions part 3

4. More on Object Definitions — 3 hours
   - Static fields and methods
   - Overloading
   - Packages (or namespaces) and encapsulation revisited
   - Intro to API libraries/packages
   - Parameter passing
   - Argument promotion and casting

5. Arrays — 3 hours
   - Declaring and creating arrays
   - Initializing an array
   - Arrays of Objects
   - For each statement
   - Multidimensional arrays
   - Command line arguments
   - Strings revisited and string buffer

6. Files and IO — 4.5 hours
   - Input streams, output streams, and print streams
   - Input and output of text files
   - Files and directories
   - Handling binary data
   - Object serialization

7. Test — 1.5 hours

8. Inheritance — 3 hours
   - Superclassses and subclasses - generalization and specialization
   - Class hierarchies
   - Is-A vs. Has-A.
   - Constructors in subclasses
   - Object class
   - Encapsulation revisited (protected members)
   - Assignments. determining type, and down casts
   - Expressing Inheritance in UML class diagrams
   - Multiple inheritance

9. Polymorphism — 3 hours
   - Polymorphism
   - Overriding

- Polymorphism behavior using arrays.
- Final attribute on methods and classes
- Abstract classes vs. concrete classes

10. Interfaces	3 hours

- Definition of an interface, a contract
- Implementation vs. use
- Programming to an interface
- Polymorphism behavior using arrays.
- Final attribute on methods and classes
- UML class diagrams for interfaces

11. Class Hierarchies and the Collections API	3 hours

- Revisit exceptions
- Revisit IO
- Collections APIs
- List, Iterator, and Map interfaces
- List and Map implementations
- Sorting and binary search

12. Low Level Graphics	3 hours

- Frame and canvas
- Coordinate system
- Graphics context, fonts, and colors.
- Drawing process
- Captures mouse and keyboard events
- Event processing thread

13. Lightweight Graphical User Interface - Swing	3 hours

- Model View Controller
- Components and containers
- Buttons and Menus
- Event listeners and adapters
- Text Components
- Checkboxes, radio boxes, and lists
- Layout managers
- Dialogs, modal and non model.
- Tables

14. Miscellaneous.	3 hours

- Generics
- Dates, Arrays, Comparator and other useful classes and interfaces

Total	42 hours
Final	2 hours

IV. Evaluation Method

Grade distribution:

Quizzes	10%
Mid Term (1 hour 15 min)	15%
Final Exam (Cumulative)	20%
Around 8 Programming Projects	50%
Participation	5 %

Grade Scale:

90 – 100%	A
80 – 89%	B
70 – 79%	C
60 – 69%	D
< 60%	F

Attendance Policy:

Attendance is crucial to success in this course. To encourage class attendance, the following policy will be used: Attendance will be taken at every class. For each unexcused absence, starting with the fourth, 2% will be deducted from the overall class grade. Generally, excused absences involve illness with a doctor's excuse, verifiable family emergencies, or conflicting university activity.

Samples of projects assigned to help students fulfill course objectives:

Project 1. Develop an application to produce a receipt indicating that a customer has purchased a given item. Basic class definitions are implemented for each type of object – Customer, Item, and Receipt. Program must use encapsulation.

Project 2. Develop a program to produce a receipt with a one to many relationship, such as a phone bill.

Project 3. Develop a program to load data for project 2 from text and binary files.

Project 4. Develop a hierarchical set of classes such as accounts with checking and savings specializations. This project will include inheritance, polymorphism, overriding, and overloading.

Project 5. Develop using low level graphics a set of classes from rendering (and possible updating) shapes. This project will introduce the concepts of low level graphics as well and reinforce class hierarchies, inheritance, and polymorphism.

Project 6. Develop a Graphical User Interface for project 3 or 4. This project will introduce the concepts of GUI design and development. The project will use of a library of predefined classes.

V. Textbook(s)

Deitel & Deitel, "Java How to Program", Eight Edition, Pearson/Prentice Hall, 2008. ISBN: 0-13-605306-8

VI. Special Resource Requirements

None.

VII. Bibliography

Course Revision –Computer Science Curriculum

1. Anderson, Julie and Franceschi, Herve'. *Java 6 Illuminated*, 2nd ed., Jones and Bartlett, Sudbury, MA. 2008.
2. Deitel, H.M. and Deitel, P.J., *C++ How to Program.* 7th ed., . Prentice Hall PTR, Upper Saddle River, NJ. 2004.

3. Friedman, Frank L. and Koffman, Elliot B. Problem Solving, Abstraction, and Design Using C++, 6th ed., Addison Wesley Longman, Inc, Reading, Mass. 2011.

4. Horstmann, Cay, *Big Java*, 4th ed., John Wiley and Sons, RRD Jefferson City. 2008.

2. Summary of Proposed Revisions

Prerequisites have been changed to COSC 108 or COSC 110.

The course outcomes are essentially the same, but have been enhanced to provide clarification and use measures targeting higher levels in Bloom's taxonomy. Two outcomes were added in relational to adapt to current trends in the industry. The additional outcomes relate to industry programming practices (outcome 5) and Unified Modeling language (UML) (outcome 8).

Course content is essentially the same, but has been reworded and reorganized. Dependency on the C++ programming language has been removed.

3. Justification for Revision

The prerequisite was changed to list the new course COSC 108 as an alternative prerequisite. COSC 108 accomplishes similar course outcomes as COSC 110, only via different pedagogy. Therefore, students completing COSC 108 will have sufficient knowledge and skill required to enroll in this course. Although COSC 108 cannot be counted toward the major, it can be taken by Computer Science minors as an alternative to COSC 110. Therefore, the prerequisite option allows entry by both majors and minors.

The addition of the industry practices outcome (5) is to instill good programming practices used by the industry early in a student's development as a programmer.

The addition of interpreting UML class diagrams outcome (8) lays the foundation for elaboration in subsequent courses. UML is now a well established design tool used in computer science.

The course content was updated to remove references to the C++ programming language (this language is no longer used in this course). In the process the content was reworded and reorganized to be in line with the selected text. However wording of the content in the syllabus is independent of any programming language.

4. The Old Syllabus of Record

I. Course Description

COSC 210 Object Oriented and GUI Programming

Prerequisite: COSC110

An in-depth introduction to the Object Oriented Programming (OOP) paradigm. The focus will be on designing, implementing, and using objects. We will cover function and operator overloading, templates, inheritance, and polymorphism. This course will also include an introduction to Graphical User Interface (GUI) design and programming.

II. Course Objectives

The student will:

1. Learn the fundamental concepts of the OOP paradigm.
2. Implement object definitions.
3. Incorporate objects and arrays of objects in application programming.
4. Use overloading, templates, and inheritance when designing, implementing, and using objects.
5. Design Object Oriented applications.
6. Learn the fundamental concepts of GUI design.
7. Develop and test object-oriented GUI programs.

III. Course Outline

A. Introduction to the Object Oriented Programming Paradigm — 2 hours
1. Overview of the fundamental concepts of OOP
2. Object model terminology
3. Introduction to the language and programming environment

B. (C++) Programming Basics — 4 hours
1. Data representation and standard types
2. Literals and literal types
3. Namespaces
4. Input and output (I/O)
5. Operators and expressions
6. Control flow constructs
7. Functions and return types
8. User defined data structures
9. Static and dynamic arrays
10. Structs and unions
11. Memory allocation
12. Debugging techniques

C. Defining Objects — 4 hours
1. Encapsulation
2. Constructors and destructors
3. Member variable and functions
4. Accessor and modifier functions
5. Public, protected, and private declarations
6. Object initialization

D. Overloading — 3 hours
1. Function overloading
2. Overloading resolution

3. Operator overloading	
4. Friend functions	
E. Using Objects	4 hours
1. Scoping	
2. Static and dynamic objects	
3. Arrays of objects	
F. Object Design	3 hours
1. Goals of Object Oriented Software	
G. Templates	4 hours
1. Template functions	
2. Simple sorting techniques (exchange, insertion, selection)	
3. Template object definitions	
H. Inheritance	4 hours
1. Derived types	
2. Virtual functions	
3. Declarations	
4. Single and multiple inheritance	
5. Public vs. private inheritance	
6. Virtual Derivations	
I. Run-Time Type Identification	2 hours
1. Polymorphism	
J. Exception Handling and Debugging	3 hours
1. Exception handling functions	
2. Expected and unexpected exceptions	
3. Using a debugger	
K. Graphical User Interface Design	2 hours
1. Goals of GUI software	
L. Implementing GUIs using Object Oriented Programming	5 hours
1. Concepts and terminology	
2. Dialogue Interface	
3. Single Document Interface	
4. Multiple Document Interface	

IV. Evaluation Methods

Evaluation:

Exams: 3 (50-60%) (including final)
Projects: 5-6 (30-40%)
Quizzes, Homework, and Lab Exercises: (10-20%)

Grading Scale:

The standard grading scale will be used.
90-100% : A; 80-89% : B; 70-79% : C; 60-69% : D; below 60% : F.

Attendance policy:

The attendance policy will conform to the University wide attendance criteria.

Samples of projects assigned to help students fulfill course objectives:

Project 1. Develop a new numerical class such as rational, complex, or Roman. This project will introduce the students to the class syntax, private and public attributes and methods.

Project 2. Expand the class developed for project 1 to incorporate operator overloading. This project will include the syntax for operator overloading and friend functions.

Project 3. Develop a template container class. This project will include template syntax, and the use of dynamic arrays.

Project 4. Develop a hierarchical set of classes such as quadrilaterals. This project will include inheritance, polymorphism, protected attributes and methods and overloading.

Project 5. Develop a Dialog Based Graphical User Interface application. This project will introduce the concepts of GUI design and development. The project will also include the use of a large library of predefined classes.

V. Textbook(s)

Deitel, H.M. and Deitel, P.J.
C++ How to Program. Second Edition.
Prentice Hall PTR, Upper Saddle River, NJ. 1998.

Deitel, H.M., Deitel, P.J., Nieto, T.R., and Strassberger, E.T.
Getting Started with Microsoft Visual C++ 6 with an Introduction to MFC
Prentice Hall PTR, Upper Saddle River, NJ. 1998

VI. Special Resource Requirements

VII. Bibliography

1. Anderson, Paul and Anderson, Gail. Navigating C++ and Object-Oriented Design, Prentice Hall PTR, Upper Saddle River, New Jersey. 1998.

2. Dale, Nell, Weems, Chip, and Headington, Mark, Programming in C++, Jones and Bartlett Publishers, Sudbury, Mass. 2001.

3. Dattatri, Kayshav. C++ Effective Object-Oriented Software Construction, Prentice Hall PTR, Upper Saddle River, NJ. 2000.

4. Eckel, Bruce. Thinking in C++, *2nd Edition.* Prentice Hall PTR, Upper Saddle River, New Jersey. 2000.

5. Friedman, Frank L. and Koffman, Elliot B. Problem Solving, Abstraction, and Design Using C++, Addison Wesley Longman, Inc, Reading, Mass. 2000.

6. Gregory, Kate. Using Visual C++ 6, QUE, Indianapolis, Indiana. 1998. (Optional text)

7. Gurewich, Ori and Gurewich, Nathan. Teach Yourself Visual C++ in 21 Days, SAMS Publishing, Indianapolis, Indiana. 1998.

8. Main, Michael and Savitch, Walter. Data Structures and Other Objects Using C++, Addison Wesley Longman Publishing Company, Reading, Massachusetts. 1997.

9. Murray, William H. and Pappas, Chris H.. MFC Programming in C++ with the Standard Template Libraries, Prentice Hall PTR, Upper Saddle River, NJ. 2000.

10. Perry, Jo Ellen and Levin, Harold D. An Introduction to Object-Oriented Design in C++, Addison Wesley Publishing Company, Reading, Massachusetts. 1996.

11. Savitch, Walter. Problem Solving with C++, Addison Wesley Publishing Company, Menlo Park, California. 1996.

12. Weiss, Mark Allen. Algorithms, Data Structures, and Problem Solving with C++, Addison Wesley Publishing Company, Menlo Park, California. 1996.

Part II. Letters of Support

Not Applicable. This is an internal change to the Computer Science program. Affected programs are all tracks in the Computer Science major, the Computer Science minor, and the Information Assurance Minor

.