Cerro Coso College

Course Outline of Record Report

10/13/2021

CSCIC254 : Object Oriented Programming

General Information

Author:	-
Course Code (CB01) :	CSCIC254
Course Title (CB02) :	Object Oriented Programming
Department:	Business Information Technolog
Proposal Start:	Fall 2013
TOP Code (CB03) :	(0706.00) Computer Science (transfer)
SAM Code (CB09) :	Non-occupational
Distance Education Approved:	Yes
Course Control Number (CB00) :	CCC000547178
Curriculum Committee Approval Date:	05/03/2013
Board of Trustees Approval Date:	06/13/2013
External Review Approval Date:	07/18/2013
Course Description:	This course follows the Introduction to Computer Science course with a focus on object oriented programming and design. A greater emphasis is placed on abstraction and using programming to solve a wide range of problems. Intermediate data structures are also addressed including trees, graphs, stacks, queues and linked lists. Students learn how to use the program development life cycle to design, code, and test programs.
Submission Type:	New Course
Author:	No value

Faculty Minimum Qualifications

Master Discipline Preferred:	• Computer Science
Alternate Master Discipline Preferred:	No value
Bachelors or Associates Discipline Preferred:	No value
Additional Bachelors or Associates Discipline Preferred:	No value

Course Development Options

Basic Skills Status (CB08)

Course is not a basic skills course.

Course Special Class Status (CB13)

Course is not a special class.

Grade Options

- Letter Grade Methods
- Pass/No Pass

☐ Allow Students to Gain Credit by Exam/Challenge

Allowed Number of Retakes

0

Course Prior To College Level (CB21)

Not applicable.

Rationale For Credit By Exam/Challenge

No value

Retake Policy Description

Type:|Non-Repeatable Credit

☑ Allow Students To Audit Course

Course Support Course Status (CB26)

No value

Associated Programs

☑ Course is part of a program (CB24)

Associated Program	Award Type	Active
CC Liberal Arts: Mathematics & Science	A.A. Degree Major	Summer 2018 to Fall 2020
Liberal Arts: Mathematics & Science Associate in Arts Degree	A.A. Degree Major	Fall 2020

Transferability & Gen. Ed. Options

Course General Education Status (CB25)

No value

Transferability	Transferability Status
Transferable to CSU only	Approved

Units and Hours:

Summary

Minimum Credit Units (CB07)	3
Maximum Credit Units (CB06)	3
Total Course In-Class (Contact) Hours	54
Total Course Out-of-Class Hours	108
Total Student Learning Hours	162
Faculty Load	0

Credit / Non-Credit Options

Course Credit Status (CB04)

Course Non Credit Category (CB22)

Non-Credit Characteristic

Credit - Degree Applicable	Credit Course.	No Value

Course Classification Status (CB11)	**Funding Agency Category (CB23)**	□ Cooperative Work Experience Education Status (CB10)
Credit Course.	Not Applicable.	

□ Variable Credit Course

Weekly Student Hours

	In Class	Out of Classs
Lecture Hours	3	6
Laboratory Hours	0	0
Activity Hours	0	0

Course Student Hours

Course Duration (Weeks)	18
Hours per unit divisor	0
Course In-Class (Contact) Hours	
Lecture	0
Laboratory	0
Activity	0
Total	54
Course Out-of-Class Hours	
Lecture	0
Laboratory	0
Activity	0
Total	108

Time Commitment Notes for Students

No value

Faculty Load

Extra Duties: 0

Faculty Load: 0

Units and Hours: - Weekly Specialty Hours

Activity Name	Type	In Class	Out of Class
No Value	No Value	No Value	No Value

Pre-requisites, Co-requisites, Anti-requisites and Advisories

Prerequisite

MATHC151 - Analytic Geometry and Calculus I

Some of the data structures that will be used are based on higher level math concepts.

AND

Prerequisite

CSCIC252 - Introduction to Computer Science

This course requires a student to know the basics of computer programming through creating functions. Students should also be comfortable writing and reading algorithms. This is all information in the CSCI C252 course.

Entrance Skills

Entrance Skills	Description
No value	No value

Limitations on Enrollment

Limitations on Enrollment	Description
No value	No value

Specifications

Methods of Instruction

Methods of Instruction	Problem Solving
Rationale	No value
Methods of Instruction	Project-based learning
Rationale	No value
Methods of Instruction	Skills Development and Performance
Rationale	No value
Methods of Instruction	Peer analysis, critique & feedback
Rationale	No value

Methods of Instruction	Outside reading
Rationale	No value

Methods of Instruction	Lecture
Rationale	No value

Methods of Instruction	Laboratory
Rationale	No value

Methods of Instruction	Discussion
Rationale	No value

Methods of Instruction	Computational Work
Rationale	No value

Assignments

A. Reading Text - Preparing for class by reading the chapters assigned
B. Programming Assignments - Programming assignments week
C. Additional Reading - Supplemental information provided by the instructor to prepare for the class
D. Group work - Group work time for a group project

Methods of Evaluation	Rationale
Homework	Programming assignments demonstrating student's ability to design a application using an object-oriented programming language, data structures, and existing library. Example: Create a grade book program. You will have a student class which will take
Research Paper	Essays demonstrating students' understanding of social computing issues. Example: Using a current event, discuss how computing in changing and what effects it will have on the future of computing.
Tests	Objective tests/quizzes demonstrating student's knowledge of tracing code, data structures, test plans, and object oriented programming. Example: Trace the following code snippet which includes a class and an object. What does the object hold befor

Equipment

No Value

Textbooks

Author	Title	Publisher	Date	ISBN

Kubica, J. (2012) Computational
Fairy Tales, , Kubica

Carrano, F., Henry, T.. (2012)
Data Abstraction & Problem
Solving with C++: Walls and
Mirrors, , Pearson

Other Instructional Materials

Description	Software: Microsoft. Microsoft C++ Express Edition 2010, Express 2010 ed. -Free C++ compiler
Author	
Citation	Object Oriented Programming

Materials Fee

No

Learning Outcomes and Objectives

Course Objectives

No value

CSLOs

Implement, test, and debug simple recursive functions and procedures.

Expected SLO Performance: 70.0

Explain how to use class hierarchies, inheritance, and polymorphism correctly to reuse existing design and code.

Expected SLO Performance: 70.0

Create programming solutions that use existing libraries and data structures including arrays, records, strings, linked lists, stacks, queues, and hash tables.

Expected SLO Performance: 70.0

Discuss significant trends and societal impacts related to computing, software, and the Internet.

Expected SLO Performance: 70.0

Compare and contrast object-oriented analysis and design with structured analysis and design.

Expected SLO Performance: 70.0

Design, implement, test, and debug simple object oriented programs.

Expected SLO Performance: 70.0

Science Liberal Arts: Mathematics & Science AA Degree	Apply algebraic, graphical, numerical, and other methods to solve applied problems in the areas of mathematics, natural sciences, computer graphics, and computer animation.

Evaluate tradeoffs in lifetime management (reference counting vs garbage collection).

Expected SLO Performance: 70.0

Outline

Course Outline

A. Societal and Professional Issues
a. Computing and the Internet
b. Social impact of computing
c. Privacy
B. Programming Languages
a. Object-oriented languages vs procedural languages
b. Effects of scale on programming methodology
C. Basic Algorithmic Analysis
a. Asymptotic analysis of upper and average complexity bounds
b. Best; average; and worst case behaviors
c. Big O and little o notations
d. Standard complexity classes
e. Empirical measurements of performance
f. Time and space tradeoffs
D. Language Translation
a. Comparison of interpreters and compilers
b. Machine-dependent/independent aspects of translation
E. Programming Constructs
a. Cohesion and decoupling
b. Assertions; including pre/post conditions and loop invariants
c. Software reuse
d. Self-documentation
e. Object oriented analysis and design
f. Component level design
F. Software Lifecycle
a. Requirements analysis and design modeling tools
b. Testing tools
c. Configuration management
G. Object Oriented Principles
a. Abstraction
b. Objects
c. Classes
d. Encapsulation
e. Inheritance
f. Polymorphism
H. Object Oriented Programming
a. Class constructors and destructors
b. ADTs
c. Reusable software components
d. APIs
e. Modeling tools
f. Class diagrams
g. Encapsulation and information hiding
h. Class hierarchies
i. Abstract and interface classes
j. Templates
I. Abstraction Mechanisms
a. Procedures; functions and iterators as abstraction mechanisms
b. Parameterization mechanisms
J. Recursion
a. Recursive mathematical functions
b. Simple recursive procedures
c. Divide-and-conquer strategies
d. Recursive backtracking
e. Implementation of recursion
K. Computing algorithms
a. Searching
b. Sorting
i. Quadratic sorting algorithms
ii. O(N log N) sorting algorithms
L. Data Structures

a. Pointers and references
b. Stacks
c. Static; stack and heap allocation
d. Queues
e. Linked lists
f. Hash tables
g. Runtime storage management
M. Graphs and Trees
a. Trees
b. Undirected graphs
c. Directed Graphs
d. Binary search trees
N. Declarations and types
a. Declaration models
b. Garbage collection
O. Software security
a. Buffer overflows
b. Memory leaks
c. Malicious code
d. Unauthorized and back-door access
e. Security-aware exception handling
P. Virtual Machines
a. Concept of virtual machine
b. Hierarchy of virtual machine
Q. Human-Computer Interaction
a. Universal principals
b. Human-centered considerations
c. Usability testing and verification
d. Design trade-offs
e. Standard API graphics
R. Event-Driven programming
a. Graphics API
b. Event Creation
c. Event-handling methods
d. Exception handling
e. Debugging in the API environment
A. Basic Algorithmic Analysis
a. Asymptotic analysis of upper and average complexity bounds
b. Best; average; and worst case behaviors
c. Big O and little o notations
B. Programming Constructs
a. Assertions; including pre/post conditions and loop invariants
b. Self-documentation
c. Object oriented analysis and design
C. Software Lifecycle
a. Requirements analysis and design modeling tools
b. Testing tools
c. Configuration management
D. Object Oriented Programming
a. Objects
b. Classes
c. Class constructors and destructors
d. ADTs
e. Reusable software components
f. APIs
g. Modeling tools
h. Class diagrams
i. Encapsulation and information hiding
j. Class hierarchies
k. Abstract and interface classes
l. Templates
E. Recursion
a. Recursive mathematical functions
b. Simple recursive procedures
c. Divide-and-conquer strategies
d. Recursive backtracking
e. Implementation of recursion

F. Computing algorithms
a. Searching
b. Sorting
i. Quadratic sorting algorithms
ii. O(N log N) sorting algorithms
G. Data Structures
a. Pointers and references
b. Stacks
c. Static; stack and heap allocation
d. Queues
e. Linked lists
f. Hash tables
g. Runtime storage management
H. Graphs and Trees
a. Trees
b. Undirected graphs
c. Directed Graphs
d. Binary search trees
I. Declarations and types
a. Garbage collection
J. Software security
a. Buffer overflows
b. Memory leaks
c. Malicious code
d. Unauthorized and back-door access
e. Security-aware exception handling
K. Human-Computer Interaction
a. Usability testing and verification
b. Standard API graphics
L. Event-Driven programming
a. Graphics API
b. Event Creation
c. Event-handling methods
d. Exception handling
e. Debugging in the API environment

Delivery Methods and Distance Education

Delivery Method: Please list all that apply -Face to face -Online (purely online no face-to-face contact) -Online with some required face-to-face meetings (“Hybrid”) -Online course with on ground testing -iTV – Interactive video = Face to face course with significant required activities in a distance modality -Other

Face 2 Face
Online
Hybrid
Interactive

Rigor Statement: Assignments and evaluations should be of the same rigor as those used in the on-ground course. If they are not the same as those noted in the COR on the Methods of Evaluation and out-of-class assignments pages, indicate what the differences are and why they are being used. For instance, if labs, field trips, or site visits are required in the face to face section of this course, how will these requirements be met with the same rigor in the Distance Education section?

No Value

Effective Student-Instructor Contact: Good practice requires both asynchronous and synchronous contact for effective contact. List the methods expected of all instructors teaching the course. -Learning Management System -Discussion Forums -Moodle Message -Other Contact -Chat/Instant Messaging -E-mail -Face-to-face meeting(s) -Newsgroup/Discussion Board -Proctored Exam -Telephone -iTV - Interactive Video -Other (specify)

contact_moodle_forums
contact_moodle_message
contact_email
contact_face2face
contact_itv
contact_other

Software and Equipment: What additional software or hardware, if any, is required for this course purely because of its delivery mode? How is technical support to be provided?

No Value

Accessibility: Section 508 of the Rehabilitation Act requires access to the Federal government's electronic and information technology. The law covers all types of electronic and information technology in the Federal sector and is not limited to assistive technologies used by people with disabilities. It applies to all Federal agencies when they develop, procure, maintain, or use such technology. Federal agencies must ensure that this technology is accessible to employees and the public to the extent it does not pose an "undue burden". I am using -iTV—Interactive Video only -Learning management system -Publisher course with learning management system interface.

s508_itv
s508_moodle
s508_publisher

Class Size: Good practice is that section size should be no greater in distance ed modes than in regular face-to-face versions of the course. Will the recommended section size be lower than in on-ground sections? If so, explain why.

No Value