

**J. Sargeant Reynolds Community College
Course Content Summary**

Course Prefix and Number: CSC 222 **Credits:** 4

Course Title: Object-Oriented Programming

Course Description:

Introduces the concepts and techniques of object-oriented programming to students with a background in procedural programming and problem solving. Uses a high-level computer language to illustrate and implement the topics. Second course in a three course sequence. (CSC 221-222-223). Lecture 4 hours. Total 4 hours per week. 4 credits.

General Course Purpose:

CSC 221, CSC 222, and CSC 223 comprise the standard sequence of minimal programming content for computer science majors. The course sequence will teach the students to use high-level languages and their applications to problem solving by using algorithms within procedural and object-oriented languages, while ensuring data adheres to a structured model. The Introduction to Object-Oriented Programming course covers the topics of classes, objects, encapsulation, cohesion, inheritance, abstraction, and polymorphism. JAVA is the preferred language for this course, institutions may offer using a different language to align with primary 4-year partner requirements.

Course Prerequisites/Corequisites:

CSC 221 or equivalent or departmental consent

Course Objectives:

Upon completing the course, the student will be able to:

Civic Engagement

- Engage and build technology that responds to human needs and helps people navigate institutional systems

Critical Thinking

- Assess why certain solutions might not work and to save time in coming up with a more efficient approach

Professional Readiness

- Work well with others and display situationally and culturally appropriate demeanor and behavior

Quantitative Literacy

- Perform accurate calculations, interpret quantitative information, apply and analyze relevant numerical data, and use results to support conclusions

Scientific Literacy

- Represent real-world objects and processes virtually by identifying properties, behavior, and operations relevant to solving problems on a computer.

Written Communication

- Develop, convey, and exchange ideas in writing, as appropriate to a given context and audience

Review of Procedural Problem-Solving Concepts

- Describe activities related to program development
- Solve problems using techniques such as pseudocode, flowcharts, UML, and model development.
- Evaluate algorithms for errors
- Discuss the presence of algorithms in various activities

Review of Procedural Programming

- Design programs using appropriate program design techniques.
- Develop programs using sequential and selection operations
- Choose adequate repetition structures based on the type of application
- Solve problems using procedures
- Develop applications using arrays

Object-Oriented Design

- List the members of a class and identify the purpose of each.
- Describe the mechanisms used to provide and restrict access to class members.
- Explain the difference between overloading and overriding
- Explain how to construct and release objects within a program
- Explain cohesion and how to achieve high cohesion
- Compare procedural design to an object-oriented design

Development & Testing Tools

- Apply a variety of tools for program development and testing.
- Apply a version control system in team or multiple revision scenarios.
- Apply the use of an automated debugger to set breakpoints and examine data values.

Abstract data type (ADT) Implementations & Applications

- Design and implement classes
- Design, implement, and manipulate objects belonging to classes
- Explain the difference between data structures that are internal versus external to a class.

Recursion

- Explain the parallels between ideas of mathematical and/or structural induction to recursion and recursively defined structures.
- Create a simple program that uses recursion.
- Describe how recursion is implemented on a computer.

Inheritance & Polymorphism

- Explain the benefits and restrictions of inheritance
- Distinguish between inheritance of implementation and inheritance of design
- Design class hierarchies using inheritance and interfaces.
- Create a class which implements an interface
- Explain how inheritance and virtual functions implement dynamic binding with polymorphism.

Files & Exceptions

- Create programs using file handling techniques
- Describe the use of relative and absolute paths to identify a file.
- Detecting end of input conditions and common error conditions.
- Explain encapsulating exceptions
- Demonstrate throwing and catching exceptions
- Write code to implement try catch and finally blocks
- Write code to create a custom Exception

Major Topics to be Included:

- Review of Procedural Problem-Solving Concepts
- Review of Procedural Programming
- Object-Oriented Design
- Development & Testing Tools
- Abstract data type (ADT) Implementations & Applications
- Recursion
- Inheritance & Polymorphism
- Files & Exceptions

Effective Date/Updated: January 1, 2022