PARKLAND COLLEGE

# Course Information Form (CIF)

**Course Code:** CSC 240 (IAI CS 912)

**Course Title:** Computer Science II (Java)

**Department:** Business/Computer Science and Technologies
**Effective Date:** Summer 2026
**PCS Code:** 1.1 - Baccalaureate/Transfer
**CIP Code:** 11.0301
**Repeatability:** 0

## Credit Hours

**Catalog Notation:** 2-2-3
**Credit Hour Distribution:**

| Lecture: 2 | Lab: 2 | Clinical: 0 | **Total: 3** |
|---|---|---|---|

## General Course Information

### Catalog Description

Advanced topics in computer science, object-oriented programming using Java, inheritance and polymorphism, linked list and tree data structures, stacks and queues, generic data types using good Object Oriented Design.

### General Course Objectives

To become proficient in using advanced computer science concepts such as object oriented design and programming including inheritance and polymorphism, recursion, and beginning data structures to develop large scale programs.

### Minimum Placement Levels

| English | Reading | Math |
|---|---|---|
| None | None | Placement out of MAT 072 |

### Prerequisites

Credit in CSC 140 with a grade of C or higher

### Methods of Evaluation

Weekly programming labs (10-12) will be given to introduce and develop software design and programming skills. 5-7 quizzes and a comprehensive final exam will test the student's retention and comprehension of the theorized material.

### Instructional Materials and Additional Supplies

Java Software Solutions, Foundations of Program Design, 9th edition. Lewis and Loftus. ISBN: 0134462025

## Course Content

### General Learning Outcomes (GLOs)

- Reasoning and Inquiry: Students will demonstrate the ability to solve problems using deductive reasoning and logic, quantitative reasoning, or the scientific method.
- Technology: Students will demonstrate the ability to evaluate, select, and appropriately use current and emerging tools.

### Course Segments and Student Learning Outcomes

| Course Segment | Learning Outcomes | Lecture Hours | Lab Hours | Clinical Hours |
|---|---|---|---|---|
| Course Introduction, Development Environment, and Writing Classes | 1. Identify course requirements, goals, and expectations.<br>2. Set up and test Java programming environment.<br>3. Review and apply fundamental concepts of designing and writing a Java class.<br>4. Explain and use both instance variables and methods, and class variables and methods.<br>5. Write programs that consist of multiple source files. | 1 | 1 | 0 |
| Object-Oriented Design | 1. Review and apply activities involved in software design and development.<br>2. Review and demonstrate the process of identifying classes and objects in problem descriptions.<br>3. Review and demonstrate principal relationships among and between classes (dependency, aggregation, inheritance).<br>4. Use data models with functions in object-oriented design.<br>5. Apply object-oriented principles.<br>6. Write methods for object assignment and copy constructors. | 2 | 2 | 0 |
| Inheritance I | 1. Review principles of inheritance in programming and demonstrate activities involved in creating sub-classes, and differentiate between single and multiple inheritance.<br>2. Explain and demonstrate the principle of overriding class methods, and differentiate between overloading and overriding methods.<br>3. Identify and explain concepts and applications of inheritance. | 2 | 2 | 0 |
| Inheritance II | 1. Explain the principle of class hierarchies in inheritance, and demonstrate activities involved in creating different class hierarchies.<br>2. Explain and demonstrate the concept and design of an abstract class.<br>3. Explain the principle of visibility in inheritance and demonstrate its application in class designs.<br>4. Explain the principles of designing programs with inheritance.<br>5. Apply function overloading and/or operator overloading where language-applicable.<br>6. Design and develop simple class hierarchies. | 2 | 2 | 0 |
| Polymorphism via Inheritance | 1. Explain the principles of polymorphism and late-binding and demonstrate their application among inherited classes. | 2 | 2 | 0 |
| Polymorphism Design | 1. Review and demonstrate the process of identifying polymorphism in problem descriptions. | 2 | 2 | 0 |
| Interfaces | 1. Explain the principle of an interface in Java and demonstrate designing, writing, and using the interface within a program.<br>2. Explain and demonstrate the application of a Java-supplied interface (e.g. Comparable, Iterator) in a program. | 2 | 2 | 0 |

| Course Segment | Learning Outcomes | Lecture Hours | Lab Hours | Clinical Hours |
|---|---|---|---|---|
| Polymorphism via Interfaces | 1. Explain and demonstrate the principles of polymorphism using interfaces. | 2 | 2 | 0 |
| Sorting and Searching | 1. Explain and demonstrate standard sorting and searching algorithms applied to lists of values in computer programs. | 2 | 2 | 0 |
| Exceptions I | 1. Explain the classifications and types of errors within a program including compile-time errors, run-time errors, and logical errors.<br>2. Explain the concept of a Java exception and show the sequence of events when handling exceptions, including generating exception objects, throwing, catching, and handling exceptions.<br>3. Explain and demonstrate the application of the try-catch and finally statements when handling exceptions within a program.<br>4. Explain and show the sequence of event propagation among multiply-nested methods. | 2 | 2 | 0 |
| Exceptions II; I/O Exceptions | 1. Explain the architecture and application of classes within the Java exception class hierarchy.<br>2. Explain and demonstrate writing a custom Java exception to handle both checked and unchecked exceptions.<br>3. Explain the concept of I/O streams in Java and demonstrate the applicability of stream classes for reading and writing information from and to a file. | 3 | 3 | 0 |
| Recursion | 1. Explain the concept of recursion and demonstrate its usefulness in a computer program to solve a problem.<br>2. Apply recursive programming techniques. | 2 | 2 | 0 |
| Collections - Linked Lists I | 1. Explain the topic of data structures and Java collections and how they are used to store, manage, and operate on large amounts of data; differentiate between static and dynamic types of data structures.<br>2. Explain and demonstrate the fundamentals of how pointers are used as links to connect objects within a linked-list data structure.<br>3. Identify fundamental concepts of data structures.<br>4. Apply concepts of dynamic binding and polymorphism (and virtual functions where language-applicable). | 2 | 2 | 0 |
| Collections - Linked Lists II, Stacks, and Queues | 1. Explain and demonstrate the usefulness of creating intermediate nodes to link objects within a linked-list data structure.<br>2. Explain and discuss fundamental linear data structures (stacks, queues) and the unique order in which items are added and removed from each.<br>3. Identify fundamental memory concepts of stack and heap.<br>4. Apply concepts of concrete/abstract classes (and interfaces where language-applicable).<br>5. Implement linked lists, including operations such as insertion, deletion, and traversal.<br>6. Use both arrays (or equivalent structures) and linked structures to implement stacks and queues, including basic operations such as insertion and deletion.<br>7. Implement various sort and search algorithms with an introduction to program verification and complexity. | 2 | 2 | 0 |

| Course Segment | Learning Outcomes | Lecture Hours | Lab Hours | Clinical Hours |
|---|---|---|---|---|
| Collections - Trees, Graphs, and Java API | 1. Explain and discuss fundamental nonlinear data structures (trees, graphs) and the unique order in which items are traversed for each.<br>2. Explain the construction of a binary search tree and demonstrate the different types of manners in which it can be traversed using recursion (inorder, preorder, postorder).<br>3. Explain the organization, contents, and applicability of the data structure classes within the Java Collections API.<br>4. Explain and demonstrate generics and how they are used within collection classes.<br>5. Use both arrays (or equivalent structures) and linked structures to implement binary trees, including basic operations such as insertion, deletion, and traversal.<br>6. Apply and use concepts of templates or generics where language-applicable. | 2 | 2 | 0 |

**Total Contact Hours**

| Lecture Hours | Lab Hours | Clinical Hours |
|---|---|---|
| 30 | 30 | 0 |