IMPERIAL VALLEY COLLEGE

Basic Course Information

Semester:	**Sprint 2023**	Instructor Name:	**Octavio Ortiz**
Course Title & #:	**CS 231**	Email:	**octavio.ortiz@imperial.edu**
CRN #:	**20550**	Webpage (optional):	**Canvas**
Classroom:	**2609**	Office #:	**2767.1**
Class Dates:	**2/13-6/9**	Office Hours:	**Faculty Schedule**
Class Days:	**T/TR**	Office Phone #:	**760-355-5706**
Class Times:	**9:40 – 12:20 PM**	Emergency Contact:	**Silvia Murray**
Units:	**3**	Class Format:	**Face-to-Face (On Ground)**

Course Description

Further training in program design and development. Object-oriented programming to include inheritance, polymorphism, and generic code. Extensive programming in Java. Introduction to data structures: arrays, stacks, queues, linked lists, trees, hash tables, dictionaries, sets and graphs. Standard methods used for sorting, searching and analyzing the relative efficiency of algorithms (Big-O notation) (CSU, UC)

Course Prerequisite(s) and/or Corequisite(s)

CS 221 with a grade of "C" or better.

Student Learning Outcomes

Upon course completion, the successful student will have acquired new skills, knowledge, and or attitudes as demonstrated by being able to:

1. Correctly determine the relative runtimes of different sort algorithms on arrays of different sizes.
2. Correctly implement an abstract data type (ADT) as a Java class.
3. Correctly use recursion to solve a problem with a data structure.

Course Objectives

Upon satisfactory completion of the course, students will be able to:

1. Analyze unstructured problems and design computer solutions.
2. Apply or create appropriate data structures to solve a particular problem.
3. Apply or a create suitable algorithm to solve a particular problem.
4. Analyze the relative efficiency of a particular algorithm.
5. Implement and test the efficiency of a particular algorithm.
6. Design and implement a stack Abstract Data Type (ADT) and queue ADT.
7. Define and implement a binary tree ADT.
8. Perform a runtime analysis of sorting algorithms.
9. Design and code a complete program of 500 lines or more.

IMPERIAL VALLEY COLLEGE

Textbooks & Other Resources or Links

Introduction to JAVA – Programming and Data Structures
Author: Y. Daniel Liang
Edition: 12th
ISBN: 978-0-13-652023-8
Copyright Year: 2020
Publisher: Pearson Prentice Hall

Course Requirements and Instructional Methods

Students will be exposed to various instructional methods. Lectures, both in person and through pre-recorded tutorial videos, will introduce students to fundamental programming concepts. Students will then apply what they learn in lectures to their own programming assignments and applications.

Programming assignments will be relatively short and will assess a student's mastery of a particular programming skill, as well as a student's ability to problem solve. Programming applications, or projects, will be more intricate. To develop an application, students will rely on the various programming and problem-solving skills they have developed up to that point.

There will be short quizzes where students will read code and answer multiple choice, true-false, and free-response questions pertaining to the code segments. A comprehensive semester final exam will assess students' ability to read, debug and rationalize code segments that range in complexity.

Course Grading Based on Course Objectives

ASSIGNMENT	POINTS
Collaborative Assignments	**15%**
Approximately 2-3 per week	
Programming Applications	**20%**
Approximately 3-5 PA's in semester	
Quizzes	**40%**
3 planned quizzes	
Projects/Final Exam	**25%**
Midterm project & comprehensive final	
Total	**100%**

Score	Letter Grade
≥ 90%	**A**
≥ 80%	**B**
≥ 70%	**C**
≥ 60%	**D**
< 60%	**F**

IMPERIAL VALLEY COLLEGE

Course Policies

Attendance:

Attendance is mandatory. Students are expected to attend every class meeting. Lectures will preview programming assignments, programming applications and future assessments.

- Although attendance is not explicitly factored into your grade, failing to complete programming assignments and assessments due to absences will negatively impact your grade.
- Students with excessive absences will be dropped from the course as outlined in AP 5075.

Late Submissions:

Programming assignments are to be completed and submitted by the due date stated on Canvas. Late programming assignments will be accepted and penalized as follows:

- 90% maximum score if submitted within 24 hours past due date
- 80% maximum score if submitted within 48 hours past due date
- 70% maximum score if submitted within 72 hours past due date
- 50% maximum score if more than three days and less than a week past due date
- No credit will be given to assignments that are one week or more past due

Programming applications/projects, quizzes and the final exam will NOT be accepted late.

Make-up Assignments:

There are no make-up assignments.

- Programming applications/projects and quizzes cannot be made up, however, if the material is presented again in future applications or quizzes, then the failed assessment will be reevaluated.

Drop Policy

The instructor reserves the right to drop students who fail to attend the first-class session or fail to complete the first assignment by the assigned due date.

Other Course Information

Resources:

https://www.w3schools.com – Learn Programming

https://docs.oracle.com/en/java/index.html - Java Documentation

IVC Student Resources

IVC wants you to be successful in all aspects of your education. For help, resources, services, and an explanation of policies, visit http://www.imperial.edu/studentresources or click the heart icon in Canvas.

IMPERIAL VALLEY COLLEGE

Course Calendar

The semester calendar is meant to provide an overview of the topics that will be covered throughout the semester. Every effort will be made to adhere to the calendar; however, changes might be necessary.

Week	Date	Topic	Assignment
Week 1	2/14	• **Syllabus & Course Policies** ○ Modules, collaborative notes, programming assignments, etc…	
	2/16	• **Inheritance & Polymorphism (Review)** ○ Superclass and subclass, extending classes, keyword "super"	
Week 2	2/21	• **Exception Handling & File Class** ○ Exception types, keyword "throws"	
	2/23	• **Abstract classes & Interfaces** ○ Abstract classes, implementing an interface	
Week 3	2/28	• **Abstract classes & Interfaces** ○ Overriding abstract methods, comparable interface	
	3/2	• **Event-Driven Programming & Animations** ○ Events & Event Sources ○ Inner-Class Handlers ○ Registering Handlers and Handling Events	
Week 4	3/7	• JavaFX Application	
	3/9	• **Review Chapters 11-13, 15** • **Quiz 1 (Chapters 11-13, 15)** ○ Deadline to submit late assignments for 50% credit.	
Week 5	3/14	• **Recursion** ○ Recursive methods, recursion vs iteration ○ Recursion & stack/heap space	
	3/16	• **Generics** ○ Benefits of generics, syntax for generic methods, syntax for generic classes & interfaces ○ Wildcard generic types	
Week 6	3/21	• **Generics** ○ Implementing generic classes	
	3/23	• **Lists, Stacks, Queues & Priority Queues** ○ Collections	
Week 7	3/28	• **Lists, Stacks, Queues & Priority Queues** ○ Iterators, for-each method, Lists	
	3/30	• **Lists, Stacks, Queues & Priority Queues** ○ Lists	
Week 8	4/4	**Midterm Project**	
	4/6	• **Review Chapters 18-20** • **Quiz 2 (Chapters 18-20)** ○ Deadline to submit late assignments for 50% credit.	

Week	Date	Topic	Assignment
Week 9	4/18	• **Lists, Stacks, Queues & Priority Queues** ○ Comparator Interface, static methods for lists and collections	
	4/20	• **Lists, Stacks, Queues & Priority Queues** ○ Queues and priority queues	
Week 10	4/25	• **Sets & Maps** ○ Sets, performance of sets and lists	
	4/27	• **Sets & Maps** ○ Maps	
Week 11	5/2	• **Developing Efficient Algorithms** ○ Algorithm efficiency and Big-O notation	
	5/4	• **Developing Efficient Algorithms** ○ Algorithm and time complexity, determining Big-O	
Week 12	5/9	• **Sorting** ○ Insertion sort, merge sort	
	5/11	• **Sorting** ○ Quick sort, bubble sort • **Quiz 3 (Chapters 21-23)** ○ Deadline to submit late assignments for 50% credit.	
Week 13	5/16	• **Implementing Lists, Stacks, Queues & Priority Queues** ○ Array Lists, Linked Lists	
	5/18	• **Implementing Lists, Stacks, Queues & Priority Queues** ○ Stacks & Queues, Priority Queues	
Week 14	5/23	• **Binary Search Trees** ○ Binary search trees, deleting elements from a BST	
	5/25	• **Binary Search Trees** ○ Tree visualization and MVC, Iterators	
Week 15	5/30	• **Hashing** ○ Hash functions and hash codes, handling collisions	
	6/1	• **Hashing** ○ Load factor and rehashing • **Final Project**	
Week 16	6/6	• **Review** ○ Review data structures, algorithms and efficiency • **Final Project**	
	6/8	• **Final Project Due** • **Comprehensive Final Exam**	

*****Subject to change without prior notice*****