CS A170: JAVA PROGRAMMING 1

Item	Value
Curriculum Committee Approval Date	03/12/2025
Top Code	070600 - Computer Science (Transfer)
Units	4 Total Units
Hours	90 Total Hours (Lecture Hours 63; Lab Hours 27)
Total Outside of Class Hours	0
Course Credit Status	Credit: Degree Applicable (D)
Material Fee	Yes
Basic Skills	Not Basic Skills (N)
Repeatable	No
Open Entry/Open Exit	No
Grading Policy	Standard Letter (S), • Pass/No Pass (B)

Course Description

A first Computer Science course taught using the Java programming language. Students will build Java applications. Emphasis will be placed on programming fundamentals such as variables, selection and loops as well as object-oriented programming concepts including classes and inheritance. ADVISORY: CIS A090; and CIS A100 or CIS A111. Transfer Credit: CSU; UC. C-ID: COMP 122.**C-ID:** COMP 122.

Course Level Student Learning Outcome(s)

1. Create, compile, execute, and test Java applications.
2. Create programs that correctly apply object-oriented principles.
3. Implement selection structures (if/else and switch), repetition structures (while, do/while, and for), and one- and two-dimensional arrays.

Course Objectives

- 1. Use Java tools to create, compile and run syntactically correct Java programs.
- 2. Demonstrate proficiency applying the fundamental concept of sequence and input-processing-output (IPO).
- 3. Demonstrate proficiency writing programs that use logic and selection statements.
- 4. Demonstrate proficiency using iteration and loops to process strings and arrays.
- 5. Demonstrate proficiency in calling methods with arguments, and defining methods and parameters of different types.
- 6. Solve accumulation, counting and other fundamental algorithms, using the arrays and lists found in the standard Java Class Libraries.
- 7. Demonstrate the ability to read the Java Class Library documentation, create objects using constructors, and call methods.
- 8. Demonstrate proficiency defining classes, making proper use of encapsulation.
- 9. Apply the concept of inheritance to solve various problems

Lecture Content

Introduction to Programming Java Mechanics: edit, compile, run and test Java programs. Computers and Memory: hardware and how data is stored in memory Software and the CPU: types of software; how the CPU works History of Programming: machine, assembly and high-level languages What about Java? History of Java and why it is important. Data and Output Java Syntax Basics: language rules for Java programs Console Output and escape sequences Variables and Values: how to declare variables and fill them with values Numbers: introduction to the primitive integer and real number types Objects, Input and Processing Interactive Programs: create programs that accept user input Processing Data: use arithmetic operators to perform calculations Math Functions: learn how to use functions, starting with the Math class Objects and methods: create objects of different types. Difference between reference types and value types. Calling accessor and mutator methods on objects Characters, Strings and Functions Side Effects: casting and the short-hand assignment operators Characters and Strings: the char primitive type and the String class type Strings and Methods: use the Java API Javadocs, start with simple String functions Parsing and Formatting: convert between Strings and numbers and format output Procedures and Functions Writing Procedures: learn syntax of procedures, technique of functional decomposition Passing Parameters: write procedures that modify behavior based on passed arguments Writing Functions: learn to write methods that calculate and return a value Syntax Errors: learn techniques for avoiding common compiler errors Style Rules: learn about the conventions used for formatting Java programs Introducing Computer Logic True False: learn how to use Java's boolean type to compare numbers Object String Relationships: learn about different ways to compare objects and Strings Introducing Selection: make decisions with the if, else and conditional operators OOP Concepts: vocabulary and principles underlying Object-Oriented Programming More on Logic Logical Operators: combine Boolean expressions using Java's logical operators Multiple Choice: make decisions involving multiple outputs Using the switch statement and the conditional operator Applying selection to data validation Learning About Loops Writing Loops: learn a strategy for writing loops that are correct. The for Loop: learn how to use Java's for statement to create counter-controlled loops The while Loop: write loops that use a counter and those that test a condition More Indefinite Loops: using do-while, managing necessary and intentional bounds Java Jumps: using break and continue to fine-tune a loop Loops, Arrays Files Iterators and Limits: use iterators to process files and Strings with limit-bound loops Primitive Arrays: how to create and initialize one-dimensional primitive arrays Arrays and Methods: learn how arrays are stored and how to write methods to process them Fundamental array algorithms: counting, accumulating, extreme values Arrays and Application Arrays, Pictures and Sound: use arrays to modify digital images and or to process sound files Partially-filled Arrays: learn how to insert and delete elements from an array Object and 2D Arrays: create arrays that contain rows and columns More Array Topics: process the command-line and variable-length parameter lists User-Defined Types Defining Fields: learn how to create new types and use the data elements they contain Defining Methods: learn to add actions to classes by defining instance methods Encapsulation: using private data and writing methods that provide safe access Constructors: defining constructors to initialize your object's private data Writing Classes: some larger case study Bugs and Testing Scope and Static Methods: writing class methods and implementing shared data elements Runtime Errors: learn to recognize runtime errors, throw exceptions and use assertions Unit Tests: learn how to create and use unit tests (JUnit framework) Debugging: learn how to use your IDE's debugger and a debugging

strategy Inheritance and Composition Introduction to Inheritance: learn how to create new classes by extending existing classes Inheritance at Work: practical examples of using inheritance Composite Objects: learn the technique of combining objects to create new classes Enumerated Types: create user-defined single-value types ArrayLists: use Java's generic ArrayList class to process collections of elements Interfaces and Exceptions Specification Inheritance: create and use abstract classes with abstract methods Interface Inheritance: use pure specification by defining and implementing interfaces Events and Interfaces: learn how to work with event interfaces and event adaptors Inner Classes: create inner classes and anonymous objects Handling Exceptions: learn to use try-catch and to create your own exception types

Lab Content

1. Working with Console Programs a. Simple Console Output (using System.out.print and println) b. Interactive Programs: Produce several programs that receive input, process it and produce output (Examples: FeetToMeters, CelciusToFahrenheit) c. Using Math functions: Write IPO programs that use the functions and constants in the Math class. Examples: CircleStats, FutureValue, PresentValue d. Formatted Output: Write programs that print output formatted into columns, and that control the way that numbers are presented. 2. Working with Graphical Programs a. Graphical Output: Produce text output using graphics b. Fonts and Colors: use different fonts and change colors c. Graphical Shapes and Output: Write programs that draw and fill shapes. (Ex. Pie Chart) d. Basic Animation: Create a PacMan, bouncing ball or similar program 3. Writing Procedures and Functions a. Decompose an output program into different procedures b. Using String Functions: Use the functions in the String class to rearrange a sentence into its individual words c. Writing String Functions: Write different String functions that examine String input and return a new String. 4. Decision Making and Selections a. Logic functions with if and if-else: Write different functions that make use of simple selection (the if statements) b. Combining Decisions: Write functions that use the logical operators to combine decisions. c. Multiway Branching: Write functions that use sequential or nested if statements to make decisions. 5. Iteration and Repetition a. Counter-Controlled For Loops: Write functions that process Strings using counter-controlled for loops. b. Counting and Finding: Write functions that process Strings to count or find some value or category. c. Using the while Loop: Write functions t hat process String and numbers using counter-controlled while loops. d. Sentinel and flag-controlled loops: Write functions that use different kinds of indefinite loops. e.Working with the ArrayList class. Compare and contrast with arrays. 6. Iterators and Processing Files a. Process a file and produce statistics (counting words, adding numbers, etc.) 7. Processing Arrays a.Arrays and Methods: Write functions that take, return and modify arrays. b. Arrays and Loops: Write functions that process arrays using loops c. Arrays and the fundamental algorithms: Write functions that calculate sums, averages, count for a condition and find the extreme and adjacent values d. Working with 2D arrays 8. Defining Classes a. Creating a basic class: Instance variables, accessor and mutator methods, constructors b. Overriding methods in the Object class: Writing toString and Equals c. Testing Classes: Writing manual unit tests (expected and actual) 9. Inheritance, Composition and Enumeration a. Extending a class without constructor chaining b. Extending a class with overriding and constructor chaining

Method(s) of Instruction

- Lecture (02)
- DE Live Online Lecture (02S)
- DE Online Lecture (02X)
- Lab (04)
- DE Live Online Lab (04S)
- DE Online Lab (04X)

Instructional Techniques

Lecture Demonstration Guided in-class exercises Clicker Questions

Reading Assignments

Students will spend a minimum of 4 hours per week reading the textbook and/or other reading material assigned. Students will be expected to follow along with the exercises in the reading material.

Writing Assignments

Students will spend a minimum of 6 hours per week writing code.

Out-of-class Assignments

Students will spend a minimum of 6 hours per week completing weekly programming assignments.

Demonstration of Critical Thinking

Written examinations, programming proficiency examinations, and in-class guided exercises.

Required Writing, Problem Solving, Skills Demonstration

Written examinations, programming proficiency examinations, and in-class guided exercises.

Eligible Disciplines

Computer science: Master's degree in computer science or computer engineering OR bachelor's degree in either of the above AND master's degree in mathematics, cybernetics, business administration, accounting or engineering OR bachelor's degree in engineering AND master's degree in cybernetics, engineering mathematics, or business administration OR bachelor's degree in mathematics AND master's degree in cybernetics, engineering mathematics, or business administration OR bachelor's degree in any of the above AND a master's degree in information science, computer information systems, or information systems OR the equivalent. Note: Courses in the use of computer programs for application to a particular discipline may be classified, for the minimum qualification purposes, under the discipline of the application. Master's degree required. Computer science: Master's degree in computer science or computer engineering OR bachelor's degree in either of the above AND master's degree in mathematics, cybernetics, business administration, accounting or engineering OR bachelor's degree in engineering AND master's degree in cybernetics, engineering mathematics, or business administration OR bachelor's degree in mathematics AND master's degree in cybernetics, engineering mathematics, or business administration OR bachelor's degree in any of the above AND a master's degree in information science, computer information systems, or information systems OR the equivalent. Note: Courses in the use of computer programs for application to a particular discipline may be classified, for the minimum qualification purposes, under the discipline of the application. Master's degree required.

Textbooks Resources

1. Required Horstmann, C.. Big Java, Early Objects, 7th ed. Wiley, 2020 2. Required Horstmann, C.. Java Concepts, Late Objects, Enhance eText, 8th ed. Hoboken NJ: Wiley, 2016