straighterline

Syllabus

CS103 | **Introduction to Programming in Java**

Course Text

There is no text for this course. All materials are included in the course fee.

Course Description

This course provides a comprehensive foundation in Java programming and computer science fundamentals. Students begin with basic syntax, variables, and control structures including branches and loops, then progress to arrays and user-defined methods. The course covers object-oriented programming principles through classes, objects, and inheritance, alongside essential topics like memory management and input/output operations. Advanced concepts include recursion, exception handling, generics, and collections. Students also explore graphical user interfaces using GUI frameworks and JavaFX, culminating in practical algorithms for searching and sorting. This course establishes the programming skills and problem-solving techniques necessary for continued study in computer science.

Learning Outcomes

After completing this course, students will be able to:

1. Apply fundamental Java syntax and programming concepts to solve basic computational problems
2. Implement conditional logic and branching structures to control program flow
3. Construct iterative solutions using various loop structures to process data efficiently
4. Design and manipulate arrays to store and process collections of data
5. Create user-defined methods to promote code reusability and modular programming
6. Develop object-oriented programs using classes, objects, and inheritance principles
7. Integrate file input/output operations to read from and write to external data sources
8. Analyze recursive algorithms and implement recursive solutions for appropriate problem domains
9. Evaluate and implement exception handling mechanisms to create robust, error-resistant programs
10. Synthesize advanced Java concepts including generics, collections, and GUI development to build comprehensive applications

Course Prerequisites

There are no prerequisites for this course.

Academic Integrity Statement

Academic integrity is the pursuit of scholarly activity in an honest, truthful and responsible manner. Violations of academic integrity include, but are not limited to, plagiarism, cheating, fabrication and academic

misconduct. Failure to comply with the Academic Integrity Policy can result in a failure and/or zero on the attempted assignment/examination, a removal from the course, disqualification to enroll in future courses, and/or revocation of an academic transcript.

Course Completion Policy

In order for a course to be considered complete, **all required coursework must be attempted, submitted, and graded.** Required coursework consists of graded assignments. Any Academic Integrity Policy violations may prevent a course from being considered complete.

Assessment Types

StraighterLine courses may include any combination of the assessment types described below. Review the descriptions to learn about each type, then review the Course Evaluation Criteria to understand how your learning will be measured in this course.

Benchmarks
Benchmarks test your mastery of course concepts. You have 3 attempts, and your highest score counts.
Note: Cumulative Benchmarks (final exams) only allow 1 attempt.

Capstones
Capstones are project-based assessments that help you apply concepts to real-world scenarios. You have 2 attempts, and your highest score counts.

Checkpoints
Checkpoints are quick knowledge checks on important course concepts. All are open-book, and most have 1-3 attempts.

AI Use-Case Policies

StraighterLine Capstone assessments operate under one of three AI Use-Case Policies. These designations are selected intentionally to support learners in developing digital literacy, ethical reasoning, and authentic communication skills. Each model requires students to engage meaningfully with the course outcomes while adhering to academic standards.

Independent Work Requirement: Capstones with this designation must be completed independently without using AI tools. The goal is for learners to showcase their own understanding and skills without AI assistance. Students are expected to generate and submit original work developed solely through their own reasoning and effort.

AI-Assisted Planning Option: Capstones with this designation may allow AI tools to support brainstorming and assessment planning. If allowed, students will be asked to document any AI assistance by noting how it informed their work. Documentation must be included within the assignment or in a designated reflection field. Examples include describing how an AI tool helped organize an outline, generate ideas, or surface sources for further exploration.

AI-Integration Requirement: Capstones with this designation require AI tools as part of the learning process. Students will be asked to reflect upon their AI interactions and AI contributions to the assessment.

Reflections must include which tools were used, how they were used, and what insights students gained from the process. This promotes transparency, ethical use, and metacognitive skill-building.

Course Evaluation Criteria

Your score provides a percentage score and letter grade for each course. A passing percentage is 70% or higher.

There are a total of 1000 points in the course:

Assessment	Points	Learning Outcomes
Checkpoint 1: Introduction to Java	20	1
Benchmark 1: Introduction to Java	42	1
Checkpoint 2: Variables/Assignments	20	1
Benchmark 2: Variables/Assignments	42	1
Checkpoint 3: Branches	20	2
Benchmark 3: Branches	42	2
Checkpoint 4: Loops	20	3
Benchmark 4: Loops	42	3
Checkpoint 5: Arrays	20	4
Benchmark 5: Arrays	42	4
Checkpoint 6: User-Defined Methods	20	5
Benchmark 6: User-Defined Methods	42	5
Checkpoint 7: Objects and Classes	20	6
Benchmark 7: Objects and Classes	42	6
Checkpoint 8: Memory Management	20	6
Benchmark 8: Memory Management	42	6
Checkpoint 9: Input/Output	20	7
Benchmark 9: Input/Output	42	7
Checkpoint 10: Inheritance	20	6
Benchmark 10: Inheritance	42	6
Checkpoint 11: Recursion	20	8
Benchmark 11: Recursion	42	8
Checkpoint 12: Exceptions	20	9
Benchmark 12: Exceptions	42	9
Checkpoint 13: Generics	20	10
Benchmark 13: Generics	42	10

Assessment	Points	Learning Outcomes
Checkpoint 14: Collections	20	10
Benchmark 14: Collections	42	10
Checkpoint 15: GUI	35	10
Checkpoint 16: JavaFX	35	10
Checkpoint 17: Searching and Sorting Algorithms	20	10
Benchmark 15: Searching and Sorting Algorithms	42	10
Total	1000	

Course Roadmap

This roadmap provides an overview of the checkpoints and lessons covered in this course.

Checkpoint 1: Introduction to Java

- Programming (general)
- Programming basics
- Comments and whitespace
- Errors and warnings
- Computers and programs (general)
- Integrated development environment
- Computer tour
- Language history
- Problem solving
- Why programming
- Why whitespace matters and precision matter
- Java example: Married-couple names

Checkpoint 2: Variables/Assignments

- Variables and assignments (general)
- Variables (int)
- Identifiers
- Arithmetic expressions (general)
- Arithmetic expressions (int)
- Example: Health data
- Floating-point numbers (double)
- Scientific notation for floating-point literals
- Constant variables
- Using math methods
- Integer division and modulo
- Type conversions
- Binary
- Characters
- Strings
- Integer overflow

- Numeric data types
- Random numbers
- Reading API documentation
- Debugging
- Style guidelines
- Local variable type face
- Java example: Salary calculation
- Java example: Salary calculation with variables
- Java example: Married-couple names with variables

Checkpoint 3: Branches

- If-else branches (general)
- Detecting equal values with branches
- Detecting ranges with branches (general)
- Detecting ranges with branches
- Detecting ranges using logical operators
- Detecting ranges with gaps
- Detecting multiple features with branches
- Common branching errors
- Example: Toll calculation
- Order of evaluation
- Switch statements
- Boolean data type
- String comparisons
- String access operations
- Character operations
- Finding and replacing text in a string
- Conditional expressions
- Floating-point comparison
- Short circuit evaluation
- Java example: Salary calculation and sale discount with branches
- Java example: Search for name using branches

Checkpoint 4: Loops

- Loops (general)
- While loops
- More while examples
- For loops
- More for loop examples
- Loops and strings
- Nested loops
- Developing programs incrementally
- Break and continue
- Variable name scope
- Enumerations
- Java example: Salary calculation with loops
- Java example: Domain name validation with loops

Checkpoint 5: Arrays

- Array concept (general)
- Arrays

- Array iteration drill
- Iterating through arrays
- Multiple arrays
- Swapping two variables (General)
- Loop-modifying or copying/comparing arrays
- Debugging example: Reversing an array
- Two-dimensional arrays
- Enhanced for loop: Arrays
- Java example: Annual salary tax rate calculation with arrays
- Java example: Domain name validation with arrays

Checkpoint 6: User-Defined Methods

- User-defined method basics
- Print methods
- Reasons for defining methods
- Writing mathematical methods
- Methods with branches
- Methods with loops
- Unit testing (methods)
- How methods work
- Methods: Common errors
- Array parameters
- Scope of variable/method definitions
- Method name overloading
- Parameter error checking
- Using Scanner in methods
- Perfect size arrays
- Oversize arrays
- Methods with oversize arrays
- Comparing perfect size and oversize arrays
- Using references in methods
- Returning arrays from methods
- Common errors: Methods and arrays
- Java documentation for methods
- Java example: Salary calculation with methods
- Java example: Domain name validation with methods

Checkpoint 7: Objects and Classes

- Objects: Introduction
- Using a class
- Defining a class
- Mutators, accessors, and private helpers
- Initialization and constructors
- Choosing classes to create
- Defining main() in a programmer-defined class
- Unit testing (classes)
- Constructor overloading
- Objects and references
- The ‘this’ implicit parameter
- Primitive and reference types
- Wrapper class conversions
- ArrayList
- Classes and ArrayLists

- ArrayList ADT
- Java documentation for classes
- Parameters of reference types
- Static fields and methods
- Using packages
- Java example: Salary calculation with classes
- Java example: Domain name availability with classes

Checkpoint 8: Memory Management

- Introduction to memory management
- A first linked list
- Memory regions: Heap/Stack
- Basic garbage collection
- Garbage collection and variable scope
- Java example: Employee list using ArrayLists

Checkpoint 9: Input/Output

- Output and input streams
- Output formatting
- Streams using Strings
- File input
- File output

Checkpoint 10: Inheritance

- Derived classes
- Access by members of derived classes
- Overriding member methods
- The Object class
- Polymorphism
- ArrayLists of Objects
- Abstract classes: Introduction (generic)
- Abstract classes
- Is-a versus has-a relationships
- UML
- Interfaces
- Java example: Employees and overriding class methods
- Java example: Employees and instantiating from an abstract class

Checkpoint 11: Recursion

- Recursion: Introduction
- Recursive methods
- Recursive algorithm: Search
- Adding output statements for debugging
- Creating a recursive method
- Recursive math methods
- Recursive exploration of all possibilities
- Stack overflow
- Java example: Recursively output permutations

Checkpoint 12: Exceptions

- Handling exceptions
- Throwing exceptions
- Exception with files
- Exceptions with methods
- User-defined exceptions
- Java example: Generate number format exception

Checkpoint 13: Generics

- Generic methods
- Generic methods and type bounds
- Generic classes
- Generic classes with type bounds
- Comparable Interface: Sorting an ArrayList
- Java example: Map values using a generic method

Checkpoint 14: Collections

- Enhanced for loop
- List: LinkedList
- Map: HashMap
- Set: HashSet
- Queue interface
- Deque interface
- SortedSet: TreeSet
- Queue: PriorityQueue
- Stack

Checkpoint 15: GUI

- Basic graphics
- Introduction to graphical user interfaces
- Positioning GUI components using a GridBagLayout
- GUI input and ActionListeners
- GUI input with formatted text fields
- GUI input with JSpinners
- Displaying multi-line text in a JTextArea
- Using tables in GUIs
- Using sliders in GUIs
- GUI tables, fields, and buttons: A seat reservation example
- Reading files with a GUI

Checkpoint 16: JavaFX

- Introduction to graphical user interfaces with JavaFX
- Positioning GUI components using a GridPane
- Input and event handlers
- Basic graphics with JavaFX

Checkpoint 17: Searching and Sorting Algorithms

- Searching and algorithms
- Binary search
- O notation
- Algorithm analysis
- Sorting: Introduction
- Selection sort
- Insertion sort
- Quicksort
- Merge sort

Related Courses

IT101
IT Fundamentals
View Course →

MAT102
Quantitative Reasoning
View Course →

MAT202
Introduction to Statistics
View Course →