ORACLE ACADEMY

Java Fundamentals

Learning Objectives

This course lays the foundation for students with little or no programming experience to learn the Java programming language. The course introduces fundamental programming concepts and terminology in an easy, engaging manner. Students will:

- Learn how to define and animate a story using the Alice 3 development environment.
- Create a game and apply Java programming constructs using the Greenfoot development environment, further enhancing a student's understanding of Java programming.
- Work with Eclipse to understand data types and operators, literals, variable initialization, scope rules, casts, and expressions.
- Examine features that relate to methods and classes such as public and private access specifiers, passing objects to a method, returning objects from a method, overloading, recursion static class members, and nested/inner classes.
- Progress into encapsulation, inheritance, and polymorphism.

Section 1: Welcome and Introduction

In this section, students will become oriented with the course structure, the hardware and software requirements, the folder structure necessary to support effective programming activities, and how scenarios produce a programmatic result. Teamwork, team roles, team tasks, and team assessment topics will be presented, along with skills required to give an effective presentation. This section will also cover journaling and how to collect artifacts for student journals.

Lesson 1: Welcome

- State the goal of the course
- Explain the course map
- Describe the teaching format used in class
- Become familiar with the computer lab, accounts, and an IDE
- Describe the computer software and folder configuration used in class
- Describe importance of teamwork and introduce a member of the course

Lesson 2: Introduction

- Describe the purpose for using Alice3 and Greenfoot tools to learn Java
- Describe the skills used to generate an animation or create a game
- Describe the components of a team project
- Create a teamwork assessment rubric
- Describe the purpose for creating a journal to document programming projects
- Describe code of ethics and cyber security

Section 2: Storytelling with Alice 3

In this section, students use storytelling techniques to develop an animation using Java. Problem solving and writing a storyboard prepares a student to define a scene and animate actors within the scene using programming constructs. The animation of actors introduces procedure definition, an understanding of algorithms, functions, and conditional statements. Presentations of students' animations, as well as journal entries, will allow them to articulate the components of Java, and the fundamental language, mastered throughout this process.

Lesson 1: Telling a Story Visually

- Compare the definitions for animation and scenario
- Write an example of using four problem-solving steps to storyboard an animation
- Write a visual and textual storyboard for an animation
- Flowchart a storyboard
- Describe an algorithm

Lesson 2: Developing an Animation Scene

- Identify all components of a scene for a given scenario
- Define a gallery using Java programming terminology
- Define classes and instances and give examples of both
- Describe three-dimensional positioning axes
- Describe the difference between precise positioning and drag-and-drop positioning
- Use a procedural method to precisely position an object in a scene
- Describe the value of saving multiple versions of an animation scene

Lesson 3: Animation Movement

- Toggle and describe the visual difference between the scene editor and the code editor
- Correlate storyboard statements with program execution tasks
- Describe actor orientation in 3D modeling
- Create programming comments
- Use procedures to move objects in a scene

- Write Java programming procedures
- Demonstrate how procedure values can be altered

Lesson 4: Controlling Animation Movement

- Define multiple control statements to control animation timing
- Recognize programming constructs to invoke simultaneous movement
- Understand variables and how they are used in programming
- Create functions to control movement based on a return value
- Define the value of a variable based on a math calculation

Lesson 5: Manipulating Animation Motion

- Create control structures to effect execution of instructions
- Create an expression to perform a math operation
- Use keyboard controls to manipulate an animation
- Create a conditional loop for repetitive behavior
- Use random numbers to randomize motion
- Complete an animation
- Test an animation

Lesson 6: Correlating Java Fundamentals with Alice 3

- Describe variables
- Describe Java simple types
- Define arithmetic operators
- Describe relational and logical operators
- Describe assignment operators
- Describe a method, class, and instance
- Describe a scenario where an IF control structure would be used
- Describe a scenario where a WHILE control structure would be used
- Recognize the syntax for a method, class, function, and procedure
- Describe input and output

Section 3: Using Greenfoot

This section builds on students' understanding of scenes, actors, procedures, and programming to develop a simple game using the Greenfoot development environment. Developing a game, and actually seeing the results of programming statements, provides continued excitement and motivation for students to learn more fundamental concepts in Java. In this section, students create instances of a class within a game, program the instances to do a task, and then compile, debug, and execute the program successfully. Games will be enhanced with animated images, sounds, and conditions that will cause a game to stop. Game boards will be created automatically with all actors and programming necessary to launch the game from a web site.

Lesson 1: Getting Started with Greenfoot

- Describe the components of the Greenfoot interactive development environment
- Create an instance of a class
- Describe classes and subclasses
- Recognize Java syntax used to correctly create a subclass

- Define parameters and how they are used in methods
- Describe how inheritance is impacted by subclass and superclass
- Describe properties of an object
- Describe the purpose of a variable
- Describe programming concepts and define terminology

Lesson 2: Examining Greenfoot Source Code

- Demonstrate source code changes to invoke methods programmatically
- Demonstrate source code changes to write an IF decision statement
- Describe a procedure to display object documentation
- Demonstrate program testing strategies
- Recognize phases for developing a software application

Lesson 3: Adding Control Statements

- Create randomized behaviors
- Define comparison operators
- Create IF ELSE control statements
- Create an instance of a class
- Recognize and describe dot notation
- Describe effective placement of methods in a super or subclass
- Simplify programming by creating and calling defined methods
- Write programming statements to include sound in a program
- Write programming statements to include keyboard movements in a program

Lesson 4: Creating a World, Animating Actors, and Ending a Game

- Construct a world object using a constructor method
- Write programming statements to use the new keyword
- Define the purpose and syntax of a defined variable
- Recognize the syntax to define or test variables
- Write programming statements to switch between two images
- Write programming statements to end a game

Lesson 5: Automating World Creation with Programming

- Define abstraction and provide an example of when it is used
- Create a while loop in a constructor to build a world
- Describe an infinite loop and how to prevent one from occurring
- Use an array to store multiple variables used to create a world
- Create an expression using logic operators
- Describe the scope of a local variable in a method
- Use string variables to store and concatenate strings

Lesson 6: Creating an Inventory of Java Fundamentals

- List and define Java programming terminology
- Recognize and label Java programming constructs
- Identify incorrect Java programming syntax
- List programming tasks to design and implement a game using Java
- List five variations to include in programming Q/A tests

Section 4: Getting Started with Eclipse

This section introduces the Eclipse environment and ensures that students can begin writing programs using control flow statements, keywords, and blocks of code to review and increase their understanding of Java programming syntax. Students will review their knowledge of data types and operators which are at the core of almost all computer languages. Literals, variable initialization, scope rules, casts, and expressions will complete the data type and operator topics.

Lesson 1: Installing and Compiling with Eclipse

- Identify components of Eclipse
- Download and install Eclipse
- Set the classpath environment variable
- Compile a sample application
- Test to ensure installation is complete
- Write the code for GalToLit.java
- Continue coding until error free compilation and execution is reached
- Modify the program to compute a different number of gallons into its equivalent number of liters

Lesson 2: Control Flow Statements and Blocks of Code

- Describe the general form of a Java program
- Use variables and apply the IF and FOR statements
- Create a block of code
- Recognize Java keywords

Lesson 3: Programming with Data Types and Operators

- Use simple types in Java code
- Specify literals for the simple types and for strings
- Demonstrate two or more ways to initialize variables
- Describe the scope rules of a method
- Understand type conversion in expressions
- Apply casting in Java code
- Use the arithmetic operators
- Use the relational and logical operators
- Use the assignment operator
- Understand type conversion in expressions
- Write a program that utilizes at least one math method
- Begin to understand and navigate the Java API
- Understand a small sample of Java classes and methods

Section 5: Control Statements, Classes, Objects and Methods

This section covers control statements which are at the heart of any programming language because they govern a program's flow of execution. Java implements control statements in a logical, cohesive manner that makes learning them straightforward. This section expands the use of keyboard input which requires a solid understanding of both classes and exceptions. Classes, objects, and methods are also at the core of Java's object model. Students will solidify their

understanding of classes and objects and expand their knowledge of methods by learning more about return types, parameters, the new operator, references, and garbage collection.

Lesson 1: Using Program Control Statements

- Create a WHILE loop
- Create a DO-WHILE loop
- Input characters from the keyboard during program execution
- Use the full forms of the IF and FOR statements
- Apply switch in Java code
- Use break effectively in Java code
- Rewrite a Java program to prompt the user for input and perform a mathematical calculation

Lesson 2: Using Classes, Objects, and Methods

- Recognize the correct general form of a class
- Create an object of a class
- Describe object references
- Create methods that compile with no errors
- Return a value from a method
- Use parameters in a method
- Add a constructor to a class
- Apply the new operator
- Describe garbage collection and finalizers
- Apply the THIS reference
- Modify a constructor to pick a random valued card and calculate it's point value

Section 6: Arrays and Strings

This section continues the examination of data types and operators by discussing arrays and string variables. Understanding the fundamental use of arrays, and how they are an integral part of classes and objects, allows students to write effective and efficient Java programs. Understanding errors and describing how they are handled allows students to write code to effectively handle an exception.

Lesson 1: Using Arrays

- Write a single-dimensional array in a Java program using primitive data types
- Write a single-dimensional array in a Java program using reference (Object) types
- Write a 2-dimensional array in a Java program using primitive data types
- Write a 2-dimensional array in a Java program using reference (Object) types
- Declare an array, initialize an array, and traverse the array
- Describe array initialization
- Use command-line arguments
- Rewrite a Java program to store integers into an array, perform a mathematical calculation, and display the result
- Write code to perform a simple Bubble Sort of integers
- Use alternative array declaration syntax

Lesson 2: Using String Objects

- Instantiate a String
- Create a reference to a String
- Use the + and += operators for concatenating Strings
- Interpret escape sequences in String literals
- Recognize the difference between a String and a primitive char data type
- Test Strings with the compareTo and equals method
- Describe why the == operator does not always work with Strings
- Write code to use String methods (length, substring, indexOf, charAt)
- Distinguish between the String method length() and an array's length value
- Create a program that will take in 5 Strings and store them in an array of Strings, sort the array in alphabetical order, and display the Strings back to the user
- Write code to use the Ternary operator
- Use the appletviewer executable and run an applet

Lesson 3: Handling Errors

- Describe the different kinds of errors that can occur and how they are handled in Java
- Describe what exceptions are used for in Java
- Determine what exceptions are thrown for any foundation class
- Write code to handle an exception thrown by the method of a foundation class

Section 7: Recursion, Abstraction, and Inheritance

This section examines several features that relate to methods and classes. It discusses the public and private access specifiers, passing objects to a method, returning objects from a method, overloading, recursion static class members, and nested/inner classes. Controlling access to class members is a fundamental aspect of encapsulation. Inheritance, the means by which hierarchical class structures are created, is described. Inheritance is the mechanism that underpins Java's support for run-time polymorphism.

Lesson 1: Passing Objects and Overloading Methods

- Use the public and private access specifiers
- Pass objects to methods
- Return objects from methods
- Overload methods
- Use variable argument methods
- Overload constructors
- Write a class with specified arrays, constructors, and methods

Lesson 2: Understanding Recursion and Static Modifier

- Create static variables
- Use static variables
- Create static methods
- Use static methods
- Create static classes

- Use static classes
- Create linear recursive methods
- Create non-linear recursive methods

Lesson 3: Understanding Inheritance

- Demonstrate and explain UML (Unified Modeling Language) class diagrams
- Use the extends keyword to inherit a class
- Compare and contrast superclass and subclass
- Explain how inheritance affects member access
- Use super to call a superclass constructor
- User super to access superclass members
- Create a multilevel class hierarchy
- State when constructors are called in a class hierarchy
- Apply superclass references to subclass objects
- Demonstrate understanding of inheritance through the use of applets
- Modify an existing applet and change the parameters

Lesson 4: Understanding Polymorphism

- Write code to override methods
- Use dynamic method dispatch to support polymorphism
- Create abstract methods and classes
- Override methods from the object class
- Demonstrate the use of final
- Explain the purpose and importance of the Object class
- Write code for an applet that displays two triangles of different colors