Tennessee Coding II Course Syllabus

High School - One Year (150 hours)

Course Overview and Goals

The CodeHS Tennessee Coding II curriculum builds on foundational programming skills to develop students into well-rounded software developers across two major platforms. Students master object-oriented programming in Java—covering variables, methods, selection and iteration, class design, data collections, and inheritance—before transitioning to mobile application development with React Native. Throughout the course, students apply the software development life cycle, design thinking, and professional practices such as version control and documentation. Upon completion, students will be able to design, build, test, and present complete Java programs and original mobile apps, demonstrating the computational thinking and engineering skills needed for advanced coursework and careers in software development.

Learning Environment

The course utilizes a blended classroom approach. The content is fully web-based, with students writing and running code in the browser. Teachers utilize tools and resources provided by CodeHS to leverage time in the classroom and give focused 1-on-1 attention to students. Each unit of the course is broken down into lessons. Lessons consist of video tutorials, short quizzes, example programs to explore, and written programming exercises, adding up to over 100 hours of hands-on programming practice in total. Several units end with a comprehensive unit test that assesses students' mastery of the material from that unit as well as challenge problems where students can display their understanding of the material.

Development Environment

Students write and run Java and JavaScript (React Native) programs in the browser using the CodeHS editor.

Prerequisites

Following state requirements, we recommend students complete the CodeHS Tennessee Computer Science Foundations course and CodeHS Tennessee Coding I before taking Tennessee Coding II. No prior experience with Java or mobile app development is required.

Technology Requirements

To complete all activities and exercises in this course, students must have access to the 3rd party sites and tools listed here: Tennessee Coding II Whitelist

More Information

Browse the content of this course at https://codehs.com/course/25295/explore

Course Breakdown

Module 1: Software Fundamentals (2 weeks / 10 hours)

Students will explore software development environments, data processing, and the software development life cycle while connecting each concept to real-world engineering workflows.
[codehs](https://codehs.com/course/28401/edit)

Topics Covered	• Choosing a Programming Language • Data Processing Methods • Comparing Software Development Environments • The Software Creation Process in an SDE • Software Development Life Cycles (SDLC) • Version Control and Requirement Management • Introduction to Algorithms, Programming, and Compilers

Example Assignments	• **Choosing the Right Tool**
	  ○ Students analyze real-world scenarios and defend whether Java or React Native is better suited for each task.
	• **Scenario Analysis**
	  ○ Students identify the data processing method used in several scenarios and explain their reasoning.
	• **Comparing IntelliJ IDEA and VS Code**
	  ○ Students compare two development environments and identify shared and distinguishing features.
	• **Choosing and Defending an SDLC**
	  ○ Students match scenarios to software development life cycle models and write a recommendation for a technical audience.

Module 2: Java Basics (2 weeks / 10 hours)

Students will deepen their understanding of Java by writing programs that use variables, data types, expressions, and input/output to solve practical problems.

Topics Covered	• Variables and Data Types • Expressions and Output • Assignment Statements and Input • Compound Assignment Operators • Documentation with Comments • String Manipulation • Escape Sequences and Formatting

Example Assignments	• **Signature with Strings**
	  ○ Students declare string variables and use print statements to produce a formatted signature.
	• **Variables About You**
	  ○ Students declare and initialize variables, choosing the best data type and a meaningful name for each.
	• **Weight of a Pyramid**
	  ○ Students write a program that estimates the weight of the Great Pyramid of Giza using arithmetic and variables.
	• **Debugging: Escape Sequences**
	  ○ Students fix a program's output formatting using only escape

	sequences.

Module 3: Using Objects and Methods (2 weeks / 10 hours)

Students learn the fundamentals of writing and running Java programs, including how to use variables, data types, expressions, and input/output. They are also introduced to classes, and learn how to create objects and call methods, including those from built-in classes such as Math and String.

Topics Covered	• Application Program Interface (API) and Libraries • Method Signatures • Objects: Instances of Classes • Object Creation and Storage (Instantiation) • Calling Class Methods • Calling Instance Methods • The Math Class

Example Assignments	• **Exploring the Java String API**
	  ○ Students complete a scavenger hunt through the Java String class documentation to learn its methods.
	• **Houses**
	  ○ Students complete provided method headers to build ASCII-art houses, practicing method calls.
	• **Averages**
	  ○ Students finish overloaded methods that calculate and return the average of two, three, and four values.
	• **Creating a Zoo**
	  ○ Students plan the structure of a zoo animal class and its associated objects.

Module 4: Selection and Iteration (5 weeks / 25 hours)

Students learn how to control the flow of a program using conditional statements and loops. They develop Boolean expressions, use relational and logical operators, and write selection statements with if, if-else, and if-else-if. Students also explore iteration through while and for loops, including nested and compound iterations.

Topics Covered	• Algorithms with Selection and Repetition • Boolean Expressions • if Statements and Nested if Statements • Compound Boolean Expressions • Comparing Boolean Expressions • while Loops and for Loops • Implementing Selection and Iteration Algorithms • Implementing String Algorithms • Nested Iteration • Informal Run-Time Analysis

Example Assignments	• **Everyday Algorithms**
	  ○ Students represent an everyday task as an algorithm that uses both selection and repetition, then flowchart it.
	• **Triple-Double**
	  ○ Students write a program that uses compound boolean logic to determine whether a basketball player earned a triple-double.

	• **Meeting Goals**
	  ○ Students build a program that checks user input against a personal goal.
	• **Debugging: If Statements**
	  ○ Students find and fix logic errors in a program that classifies a number as positive, negative, or zero.

Module 5: Class Creation (3 weeks / 15 hours)

Students learn how to design and implement classes using principles of object-oriented programming. They explore encapsulation, access control, and the structure of classes with instance variables, constructors, and methods. Students write accessor and mutator methods, define behaviors using parameters and return values, and understand how object references are passed and returned.

Topics Covered	• Abstraction and Program Design • Impact of Program Design • Anatomy of a Class • Constructors • Methods: How to Write Them • Passing and Returning Object References • Class Variables and Methods • Scope and Access

Example Assignments	• **Public Library System**
	  ○ Students outline and diagram a class-based program to manage a community's public library branches, applying abstraction.
	• **Smart Lights**
	  ○ Students complete a SmartLight class that tracks on/off state and brightness level.
	• **Letter Grades**
	  ○ Students complete an Exam class that stores a score and computes its corresponding letter grade.
	• **Debugging: Circle Access**
	  ○ Students correct privacy and access settings in the Circle and CircleRunner classes to follow best practices.

Module 6: Data Collections (4 weeks / 20 hours)

Students learn how to work with collections of data using arrays, ArrayList objects, and 2D arrays. They develop and analyze algorithms that involve searching, sorting, traversing, and manipulating elements in these structures. Students also explore reading data from text files and using wrapper classes to convert between primitive types and objects.

Topics Covered	• Ethical and Social Issues Around Data Collection • Introduction to Using Data Sets • Array Creation, Access, and Traversals • Implementing Array Algorithms • Using Text Files • Wrapper Classes • ArrayList Methods and Traversals • Implementing ArrayList Algorithms • Searching Algorithms

	• Sorting Algorithms

Example Assignments	• **Encapsulation and Secure Coding**
	  ○ Students modify a UserAccount class to enhance the security of private data through encapsulation.
	• **FitBit Data Set: Sleep Analysis**
	  ○ Students analyze a real fitness data set to draw conclusions about sleep patterns.
	• **Everyday Life Data Set**
	  ○ Students collect their own data set, write a guiding question, and design an algorithm to analyze it.
	• **Data Breach Current Events**
	  ○ Students examine a real-world data breach and respond to questions about data collection risks.

Module 7: Inheritance and Polymorphism (2 weeks/10 hours)

Students learn how to recognize common attributes and behaviors that can be used in a superclass and create hierarchies by writing subclasses to extend a superclass.

Topics Covered	• Inheritance • Writing Constructors for Subclasses • Overriding Methods • The super Keyword • Creating References Using Inheritance • Polymorphism • The Object Superclass

Example Assignments	• **Books**
	  ○ Students refactor a Book superclass with Encyclopedia and Mystery subclasses to optimize the hierarchy.
	• **Computers**
	  ○ Students design a Computer superclass with Laptop and Desktop subclasses.
	• **Instruments**
	  ○ Students build an Instrument hierarchy with Wind and Strings subclasses.
	• **Electric Cars**
	  ○ Students create an ElectricCar subclass that overrides methods of an existing Car class.

Module 8: Applied Object Oriented Programming (3 weeks/15 hours)

Students will apply object-oriented programming concepts to build and refine larger Java projects that integrate multiple classes, features, and data structures in realistic scenarios.

Topics Covered	• Planning Your Development Strategy • Designing and Implementing the Virtual Pet Class • Interacting Classes • Steganography and the Picture Lab (Exploring Color) • Hiding and Revealing Pictures and Messages • The Celebrity Lab • Managing Your Software's Development Process

	• Writing End-User Documentation

Example Assignments	• **Virtual Pets: Implementing a Class**
	  ○ Students implement a VirtualPet class that tracks happiness and energy and responds to feeding and play.
	• **Steganography Project**
	  ○ Students manipulate pixel color values to hide and reveal images and text messages within pictures.
	• **Celebrity Lab**
	  ○ Students build and extend the Celebrity guessing-game classes across a multi-part lab.
	• **Writing End-User Documentation**
	  ○ Students write clear documentation that helps end users understand and operate their program.

Module 9: React Native Foundations (3 weeks/15 hours)

Students are introduced to the mobile apps course and the React Native framework and its program structure and syntax. Students also preview some of the tools and technologies they will use to build and run their apps.

Topics Covered	• Introduction to React Native and Expo • Introduction to Components • The Stylesheet API • Styling View and Text Components • TouchableHighlight and the onPress Function • TextInput • Flex Layouts • Using Dimensions to Control Size

Example Assignments	• **Rainbow Background**
	  ○ Students use the Stylesheet to build a full-width rainbow background with evenly sized color strips.
	• **Andy Warhol Image**
	  ○ Students use nested View components to recreate a pop-art style grid layout.
	• **My Favorites List**
	  ○ Students build an app that displays a styled list of five of their favorite things.
	• **Challenge: Scorekeeper**
	  ○ Students build a scorekeeping app across multiple parts, adding layout, styling, and responsive design.

Module 10: Building Mobile Apps (3 weeks / 15 hours)

Students learn how to use state values and how to update the state of their app in various ways to create quick, dynamic programs. They will learn how mathematical equations and string methods can be used to alter values as the user interacts with their programs.

Topics Covered	• Using State to Control Values

	• Updating State with onPress • Using Mathematical Equations to Update State • Intro to Design Thinking • Empathy and Inclusivity • Define and Ideate • Prototype and Test

Example Assignments	• **Full Counter App**
	  ○ Students build an app with increase and decrease buttons that update a displayed value using state.
	• **Easy Calculator**
	  ○ Students create an app with text inputs and operation buttons that performs basic arithmetic.
	• **User Interface Scavenger Hunt**
	  ○ Students evaluate the user interfaces of real apps to identify good and poor design choices.
	• **Design Thinking Project**
	  ○ Students interview a classmate, define a problem, and ideate and prototype an inclusive solution.

Module 11: End of Course App Build (1 week / 5 hours)

Students use everything they have learned in this course to design, prototype, and code their own custom app and present it to their class In this project-facing module.

Topics Covered	• App Planning and Requirements • Planning with Pseudocode • Coding the Layout and Styles • Implementing App Functionality • Presenting the Finished App

Example Assignments	• **Build Your Own App**
	  ○ Students plan, design, and code an original mobile app that meets a defined set of requirements.
	• **App Presentation**
	  ○ Students create and deliver a presentation showcasing their app and the work behind it.