All Guides

Welcome to the TCSS 305 learning guides. These guides supplement assignment instructions with deeper explanations of concepts, tools, and best practices.

Browse by Topic

Getting Started

Essential setup and workflow guides for the course.

Environment Setup

Install JDK 25, IntelliJ IDEA, and configure your development environment.

→ Environment Setup

IDE Basics

Navigate IntelliJ IDEA, run code, debug, and use essential features.

→ IDE Basics

Java Packages

Organize code with packages, understand naming conventions and imports.

→ Java Packages

Git Workflow

Version control fundamentals, branching, and collaborative workflows.

→ Git Version Control
→ Git Branching & PRs

Code Quality

Tools and practices for writing clean, maintainable code.

Linters and Code Quality

Why linting matters and how to use Checkstyle and IntelliJ inspections.

→ Linters and Code Quality

Defensive Programming

Validate inputs, fail fast, and write robust code that handles edge cases.

→ Defensive Programming

Logging

Use `java.util.logging` for debugging and runtime diagnostics.

→ Logging

Core Concepts

Fundamental Java and OOP concepts used throughout the course.

Interface Contracts

Design by contract, preconditions, postconditions, and invariants.

→ Interface Contracts

Inheritance Hierarchies

Design class hierarchies, use `extends`, and understand polymorphism.

→ Inheritance Hierarchies

Polymorphism

Same method call, different behavior. The core of object-oriented programming.

→ Polymorphism

Java Enums

Type-safe constants and enumeration types with methods and fields.

→ Java Enums

Comparable and Comparator

Natural ordering and external comparison strategies for objects.

→ Comparable and Comparator

Sealed Types & Records

Modern Java features: sealed classes, records, and pattern matching.

→ Sealed Types & Records

BigDecimal & BigInteger

Arbitrary precision arithmetic for financial and scientific calculations.

→ BigDecimal & BigInteger

Object Methods: equals, hashCode, toString

Override `Object` methods correctly and understand the contract.

→ Implementing equals/hashCode/toString
→ The equals/hashCode Contract

Testing

Write effective tests and practice test-driven development.

Introduction to Unit Testing

Why we test, what unit tests are, and testing fundamentals.

→ Intro to Unit Testing

Test-Driven Development

Red-green-refactor cycle and writing tests before implementation.

→ Test-Driven Development

Writing JUnit 5 Tests

Annotations, assertions, parameterized tests, and JUnit 5 features.

→ Writing JUnit 5 Tests

Testing Complex Logic

Test random behavior, preference hierarchies, and seeking algorithms.

→ Testing Complex Logic

GUI & Events

Build graphical user interfaces and handle user interactions.

Event-Driven Programming

Understand event loops, the Hollywood Principle, and how GUIs respond to user actions.

→ Event-Driven Programming

Swing API Basics

Java GUI history, JFrame/JPanel hierarchy, and core Swing patterns.

→ Swing API Basics

Swing Layout Managers

Arrange components with BorderLayout, FlowLayout, GridLayout, and BoxLayout.

→ Swing Layout Managers

Custom Painting with Java 2D

paintComponent, Graphics2D, coordinate system, shapes, and mapping grid coordinates to pixels.

→ Custom Painting with Java 2D

Building Menus with JMenuBar

JMenuBar, JMenu, JMenuItem, Exit pattern, and JOptionPane.

→ Building Menus with JMenuBar

Adding Event Handlers

ActionListeners on buttons — from inner classes to lambdas.

→ Adding Event Handlers

λ Introduction to Lambda Expressions

Lambda syntax, functional interfaces, and method references.

→ Introduction to Lambda Expressions

Handling Mouse Events

MouseListener, MouseAdapter, and tracking mouse interactions on a canvas.

→ Handling Mouse Events

Handling Key Events

KeyListener, KeyAdapter, key bindings, and responding to keyboard input.

→ Handling Key Events

Animation with javax.swing.Timer

Timer-driven animation, game loops, and periodic UI updates.

→ Animation with javax.swing.Timer

Design Patterns

Proven solutions to recurring software design problems.

Introduction to Design Patterns

What design patterns are, why they matter, and how to apply them.

→ Introduction to Design Patterns

The Observer Pattern

Decouple subjects from observers with publish-subscribe notification.

→ The Observer Pattern

Model-View-Controller (MVC)

Separate data, presentation, and user interaction into distinct roles.

→ Model-View-Controller (MVC)

The Strategy Pattern

Encapsulate interchangeable algorithms behind a common interface.

→ The Strategy Pattern

Advanced Topics

Optional deep-dives for exploring beyond requirements.

Exploring the Road Rage Codebase

Deep-dive into design patterns in the starter code (Strategy, Factory, Observer, Sealed Types).

→ Exploring Road Rage Codebase

Quick Reference

Lookup materials for specific rules and inspections. Consult as needed.

Checkstyle Reference

Complete reference for all Checkstyle rules used in TCSS 305.

→ Checkstyle Reference

IntelliJ Inspections Reference

Reference for IntelliJ inspections enabled in course projects.

→ IntelliJ Inspections

Creating Custom Maps

Map file format reference for the Road Rage simulation.

→ Creating Custom Maps

By Assignment

Guides are introduced progressively throughout the course. Each assignment below lists the **new** guides for that assignment.

Assignment 1a: Bookstore Inventory

→ Environment Setup
→ IDE Basics
→ Java Packages
→ Git Version Control
→ Linters and Code Quality
→ Checkstyle Reference
→ IntelliJ Inspections Reference
→ Interface Contracts
→ Logging
→ Intro to Unit Testing

Assignment 1b: Testing the Bookstore

→ Test-Driven Development
→ Writing JUnit 5 Tests
→ The equals/hashCode Contract

{} Assignment 1c: Implementing the Bookstore

→ Git Branching & PRs
→ Inheritance Hierarchies
→ Sealed Types & Records
→ Implementing equals/hashCode/toString

→ BigDecimal & BigInteger
→ Defensive Programming

Assignment 2: Road Rage

→ Java Enums
→ Polymorphism
→ Comparable and Comparator
→ Testing Complex Logic
→ Creating Custom Maps
→ Exploring the Road Rage Codebase

Assignment 3: Sketch Pad

→ Event-Driven Programming
→ Swing API Basics
→ Swing Layout Managers
→ Adding Event Handlers
→ Introduction to Lambda Expressions
→ Handling Mouse Events

Group Project: Tetris (Sprint 1)

→ Building Menus with JMenuBar
→ Custom Painting with Java 2D

Group Project: Tetris (Sprint 2)

→ Handling Key Events
→ Animation with javax.swing.Timer
→ Introduction to Design Patterns
→ The Observer Pattern
→ Model-View-Controller (MVC)
→ The Strategy Pattern

Reading Order

For those working through the guides systematically, here's a recommended sequence organized by topic area.

Part I: The Workshop

Set up your development environment and learn your tools.

#	Guide	Description
1	Environment Setup	Install JDK 25, IntelliJ IDEA, and configure your system
2	IDE Basics	Navigate IntelliJ, run code, and debug
3	Java Packages	Organize code with packages and imports

Part II: Version Control

Track changes and collaborate with Git.

#	Guide	Description
4	Git Version Control	Commits, pushing, authentication
5	Git Branching & PRs	Branches, pull requests, collaboration

Part III: Code Quality

Write clean, maintainable code that follows professional standards.

#	Guide	Description
6	Linters and Code Quality	Why linting matters

#	Guide	Description
7	Defensive Programming	Validate inputs, fail fast
8	Logging	Runtime diagnostics with `java.util.logging`

Part IV: Object-Oriented Foundations

Core OOP concepts and Java features used throughout the course.

#	Guide	Description
9	Interface Contracts	Design by contract, pre/postconditions
10	Inheritance Hierarchies	Class design and polymorphism
11	Polymorphism	Same method call, different behavior — the core of OOP
12	Java Enums	Type-safe constants and enumeration types
13	Comparable and Comparator	Natural ordering and comparison strategies
14	Sealed Types & Records	Modern Java features
15	BigDecimal & BigInteger	Arbitrary precision arithmetic
16	Implementing equals/hashCode/toString	Override Object methods correctly
17	The equals/hashCode Contract	Understand and test the contract

Part V: Testing

Verify your code works and practice test-driven development.

#	Guide	Description
18	Introduction to Unit Testing	Why we test, testing fundamentals
19	Test-Driven Development	Red-green-refactor cycle
20	Writing JUnit 5 Tests	Annotations, assertions, JUnit 5 features
21	Testing Complex Logic	Test random behavior, preferences, and seeking algorithms

---

Part VI: GUI & Events

Build graphical interfaces and handle user interactions.

#	Guide	Description
22	Event-Driven Programming	Event loops, the Hollywood Principle, GUI event model
23	Swing API Basics	JFrame/JPanel hierarchy, core Swing patterns
24	Swing Layout Managers	BorderLayout, FlowLayout, GridLayout, BoxLayout
25	Custom Painting with Java 2D	paintComponent, Graphics2D, coordinate system, shapes, and pixel mapping
26	Building Menus with JMenuBar	JMenuBar, JMenu, JMenuItem, Exit pattern, and JOptionPane
27	Adding Event Handlers	ActionListeners — from inner classes to lambdas

#	Guide	Description
28	Introduction to Lambda Expressions	Lambda syntax, functional interfaces, method references
29	Handling Mouse Events	MouseListener, MouseAdapter, mouse tracking
30	Handling Key Events	KeyListener, KeyAdapter, key bindings
31	Animation with javax.swing.Timer	Timer-driven animation and game loops

Part VII: Design Patterns

Proven solutions to recurring software design problems.

#	Guide	Description
32	Introduction to Design Patterns	What design patterns are and why they matter
33	The Observer Pattern	Publish-subscribe notification between objects
34	Model-View-Controller (MVC)	Separate data, presentation, and interaction
35	The Strategy Pattern	Encapsulate interchangeable algorithms

Part VIII: Advanced Topics

Optional deep-dives for those wanting to explore beyond requirements.

#	Guide	Description
36	Exploring the Road Rage Codebase	Design patterns in the starter code

Appendices

Reference materials to consult as needed. Not meant to be read cover-to-cover.

Ref	Title	Description
A	Checkstyle Reference	All Checkstyle rules explained
B	IntelliJ Inspections Reference	All IntelliJ inspections explained
C	Creating Custom Maps	Map file format reference for Road Rage