Setting Up Your Java Development Environment

COP2250: Java Programming

Professor Pyatt

Week of January 13, 2026

Today's Objectives

- Install and configure JDK 21 (LTS)
- Set up VS Code with Java extensions
- Learn terminal-based compilation workflow
- Configure GitHub for code sharing
- Introduction to AI-assisted development with Claude

Installing the JDK

- Download JDK 21 (LTS) from `https://adoptium.net`
- Run the installer for your operating system
- Verify installation from terminal:

```
java --version
javac --version
```

- Set `JAVA_HOME` environment variable if needed
- LTS = Long Term Support (stable, production-ready)

VS Code + Java Extensions

- Download VS Code from `https://code.visualstudio.com`
- Install the **Extension Pack for Java** (Microsoft)
- This pack includes:
  - Language Support for Java
  - Debugger for Java
  - Test Runner for Java
  - Maven/Gradle support
- VS Code gives you IDE features while staying lightweight
- You can still use terminal for compilation

Compiling from Terminal

The terminal workflow shows you what the IDE does under the hood:

```
# Create project structure
mkdir -p MyProject/src
cd MyProject/src

# Create a Java file
# (use VS Code or any text editor)

# Compile
javac HelloWorld.java

# Run
java HelloWorld
```

Understanding this process helps you debug IDE issues and work on any system.

Project Structure

Organize your projects consistently:

```
MyProject/
  src/              # Source files (.java)
    Main.java
    MyClass.java
  bin/              # Compiled files (.class)
  README.md         # Project documentation
```

- Keep source and compiled files separate
- Use `javac -d bin src/*.java` to compile to bin folder
- Run with `java -cp bin Main`

GitHub for Code Sharing

- Create account at `https://github.com` (if you don't have one)
- Install Git from `https://git-scm.com`
- Configure your identity:

```
git config --global user.name "Your Name"
git config --global user.email "your@email.com"
```

- Each project = one repository
- You'll share your code by pushing to GitHub
- Portfolio-ready: employers look at GitHub profiles

Basic Git Workflow

```
# Initialize a new repo
git init

# Stage your changes
git add .

# Commit with a message
git commit -m "Initial commit"

# Connect to GitHub (first time only)
git remote add origin https://github.com/user/repo.git

# Push to GitHub
git push -u origin main
```

We'll practice this workflow in lab today.

Using Claude for Development

- Claude is an AI assistant that can help you write and debug code
- It's a conversation, not a magic answer machine
- Effective prompting:
  - Be specific about what you're trying to do
  - Share your code and error messages
  - Ask for explanations, not just solutions
  - Iterate: "That didn't work because..."
- Use it to learn, not to bypass learning
- You're still responsible for understanding your code

Claude Demo

Live demonstration:

- Starting a new Java project
- Asking Claude to scaffold a class
- Debugging a compilation error
- Asking "why" questions to understand the code

Key principle: Claude is a pair programmer, not a replacement for thinking.

Lab: Environment Setup

Complete the following setup checklist:

1. Install JDK 21 and verify with `java --version`
2. Install VS Code + Extension Pack for Java
3. Create a project folder with proper structure
4. Write, compile, and run a simple Java program from terminal
5. Initialize a Git repo and make your first commit
6. Create a GitHub repo and push your code
7. Share your repo link in the class discussion

Deliverable: GitHub repo link with your first Java program

Next Class: Thursday

- We'll dive into Classes, Objects, and Methods
- You'll build a complete project using OOP concepts
- Make sure your environment is fully set up before Thursday
- Bring questions about anything that didn't work today