assignment # group-project

Sprint 0 – Setup & Familiarization

School of Engineering and Technology, University of Washington Tacoma
TCSS 305 Programming Practicum, Winter 2026
Value: 5% of the course grade

Due Date

Sunday, 22 February 2026, 23:59:59

📄 Description

Sprint 0 prepares your team to build a Tetris game with a graphical user interface. You will hold your first group meeting, set up a shared GitHub repository, learn the provided back-end API by building a simple console application, and complete required reading on collaborative Git workflows.

The back-end/model is provided as a library; your team will build the front-end GUI and connect it to the back-end using the Observer Design Pattern in later sprints. This sprint is about getting organized and understanding the tools you will use.

🎯 Learning Objectives

By completing this sprint, you will:

- Conduct a structured team meeting and document the agenda and outcomes
- Set up a shared GitHub repository using GitHub Classroom's group assignment workflow
- Practice collaborative Git workflows: branching, merging, and pushing
- Explore a provided Java API by reading documentation and writing a console application
- Understand Git collaboration models (shared repository model)

✅ Before You Begin

Ensure you have completed:

- JDK 25 installed and configured
- IntelliJ IDEA installed with Checkstyle plugin
- Git installed and configured
- GitHub account created
- Completed all prior individual assignments

✏️ Requirements

Requirement 1: First Group Meeting

All group members must attend the first group meeting.

1. Use When2Meet to find a time when everyone can meet
2. Use the meeting agenda template linked below — this template **must** be followed and the meeting documented. This is part of the grade.
3. Do not rush this meeting — estimate about an hour

Meeting Agenda Template

Google Doc Template

Caution

You must be logged into your **UW Google account** to access the meeting agenda template. If you receive a permissions error, switch to your `@uw.edu` account in Google.

Requirement 2: Repository Setup

All members should be present for this work (remote is acceptable).

Guide

- Git Version Control — Git fundamentals: commits, branches, and remote repositories
- Git Branching and Pull Requests — How to work with branches and merge changes

Accept the Assignment

This project uses **GitHub Classroom** to distribute starter code and manage team repositories.

GitHub Classroom Assignment

Accept the Group Project

FIRST TEAM MEMBER

1. Click the GitHub Classroom assignment link provided by your instructor
2. **First time only:** Select your name from the course roster to link your UW identity to your GitHub username
3. You will be prompted to **create a new team** — name your team `Group N` (where N is your group number)
4. Click **Accept this assignment**. GitHub Classroom creates a shared team repository.

REMAINING TEAM MEMBERS

1. Click the same GitHub Classroom assignment link
2. **First time only:** Select your name from the course roster
3. Find your team name in the list and click **Join**
4. GitHub Classroom adds you to the existing team repository — all members have push access automatically

Important

All team members must accept the assignment through GitHub Classroom. Do **not** skip this step — your instructor uses GitHub Classroom to track team membership and access repositories.

Set Up Branches

In your team repository:

1. **Code** → **Branches** → **Create a new branch** from `main` called `sprint0`

2. **Code → Branches → Create a new branch** from `sprint0` called `testingGit`
3. **Code → Code Button → Copy HTTPS URL**

IntelliJ Setup

> **Guide**
>
> IDE Basics — Getting started with IntelliJ IDEA

1. **New Project from Version Control**
2. Paste your GitHub repository HTTPS URL and follow prompts
3. Once the project is open, check out both `sprint0` and `testingGit` branches to your local repo
4. In the `testingGit` branch, correct all Checkstyle warnings in `Sandbox.java`
5. Commit (with a commit comment) the changes to your local repo, then push to the remote repo
6. Switch to `sprint0` branch and merge `testingGit` into it. Push the merge to the remote repo.
7. Switch to `main` branch and merge `sprint0` into it. Push the merge to remote repo.

Practice with Git/GitHub

Repeat the branching and merging process. Each team member should practice creating branches, making changes, committing, pushing, and merging. Spend an hour or so on this.

> **Tip**
>
> This practice time is valuable. The more comfortable your team is with Git workflows now, the fewer merge conflicts and lost work you will encounter in later sprints.

Requirement 3: Learn the Back-End API

Individual Work

Read and learn the provided back-end/model API documentation:

Back-End API Documentation

Tetris API Javadoc

Group Discussion

As a group, discuss, review, and strategize how to use the API.

Review the Starter Project

Individually review the starter project in IntelliJ:

- Notice the back-end is provided via a library `.jar` file
- In the `edu.uw.tcss.app` package, there is a short application that instantiates a back-end object and calls several methods
- Run it to improve your understanding

Console UI Practice Application

Create a new application file with a `main` method. This will be a simplified Console User Interface (CUI) version of Tetris using the `toString()` method provided in the `TetrisGame` class.

The CUI must provide options to:

- Start a new Tetris game
- Move the movable piece down one, left, and right
- Rotate the movable piece clockwise and counterclockwise
- Drop the movable piece
- Pause and unpause the Tetris game
- End the Tetris game

Note

- When the game is paused, attempt to move, drop, and rotate the movable piece to see what happens
- The `toString()` does not inform when the game ends programmatically; there is no requirement to handle this
- There is no requirement for the board to auto-advance (the piece only moves when the user interacts)
- **This code will NOT be used in later sprints** — the goal is familiarization with the API

Requirement 4: Required Reading

Read all of the following GitHub documentation:

- About Commits
- Comparing Commits
- Commit Exists on GitHub but Not in My Local Clone
- Shared Repository Model
- About Branches
- Creating and Deleting Branches
- About Pull Requests
- Comparing Branches in Pull Requests

> **Important**
>
> We are **NOT** using the Fork and Pull model. This course uses the **Shared Repository Model**.

Requirement 5: Executive Summary

Your project includes a file called `executive-summary.md`. This file documents your team's contributions, meeting notes, and commentary for each sprint.

The `.md` file extension stands for Markdown. Markdown is a markup language used to format plain text. You may already know a markup language — the M in HTML and XML stands for markup. (Note: markup languages are NOT programming languages but instead are tools for formatting text.)

> **External Resource**
>
> Markdown Cheat Sheet

Edit the `executive-summary.md` file and complete the following for Sprint 0:

- **Team Members** — list all group members with GitHub usernames
- **Meeting Agenda** — add the link to your Google Doc meeting agenda

- **Sprint 0 Contributions** — for each group member, write 2–3 paragraphs describing their contribution to Sprint 0 (include details about what they worked on, what they learned, and how they supported the team)
- **Sprint 0 Comments** — document any issues, ideas, code weirdness, etc.

> **Important**
>
> The executive summary is a living document. You will update it each sprint with new contributions, meeting notes, and comments. Keep previous sprint entries intact.

📚 Guide Reference

Guide	Description
Git Version Control	Git fundamentals: commits, branches, and remote repositories
Git Branching and Pull Requests	How to work with branches, merge changes, and submit via pull request
IDE Basics	Getting started with IntelliJ IDEA

🚀 Submission and Grading

> **Important**
>
> Start early — repository setup and Git practice require coordination among all group members.

Please see the rubric in Canvas for a breakdown of the grade for this sprint.

Submitting Your Work

1. Ensure all repository setup is complete (branches created, merged, all members have accepted via GitHub Classroom)
2. Ensure your Console UI practice application is committed and pushed

3. Ensure your `executive-summary.md` is complete with all required content for Sprint 0
4. Submit the GitHub repository link to the assignment in Canvas

> **Tip**
>
> You can commit and push multiple times. Only your final push before the deadline will be graded.