CS 249: Core Assignment Rules

Programming Assignments

- ALL code files should be in the package **edu.*yourname*.assignXX**, where:
  - *yourname* is replaced with your SITNet ID *in lowercase letters*
  - XX is replaced with the zero-padded assignment number (01, 02, etc.)
- Prompts and printouts match **EXACTLY** to the specification
  - This includes capitalization, punctuation, spaces, and newlines!
- Your java files/class names/ method names MUST match the spelling/capitalization **EXACTLY**!
- ALL data in your class should be **PRIVATE** or at least **PROTECTED**!
- Unless otherwise stated, NONE of the class data in the assignments should be static! All class data is **INSTANCE** data!
- Except for replacing my SITNET ID with your own, **do NOT modify:**
  - The test programs
  - Any **complete** programs I provide

Starting a New Assignment

For each new assignment, you must do the following procedures:

- **Switch to the "main" branch**
  - "Git" → "Branches" → click on "main" → "Checkout"
- **Fetch any changes from remotes**
  - "Git" → "Fetch"
- **Pull changes from "upstream" remote repository (i.e., my repository)**
  - "Git" → "Pull" → select "upstream"
- **Create a new branch for this assignment**
  - "Git" → "Branches" → "New Branch…"
  - Use name "assignXX", where XX is the zero-padded assignment number (01, 02, etc.)
  - Make sure "Checkout branch" is selected
- **Publish your new branch**
  - "Git" → "Push"
- **Under the appropriate subproject "assignXX":**
  - **Create package folders for main code**
    - Right-click on assignXX/src/main/java folder → "New" → "Package"
    - Name should be: **edu.*yourname*.assignXX**
  - **Create necessary Java files in this package folder**
    - Right-click on package folder → "New" → "Java Class"

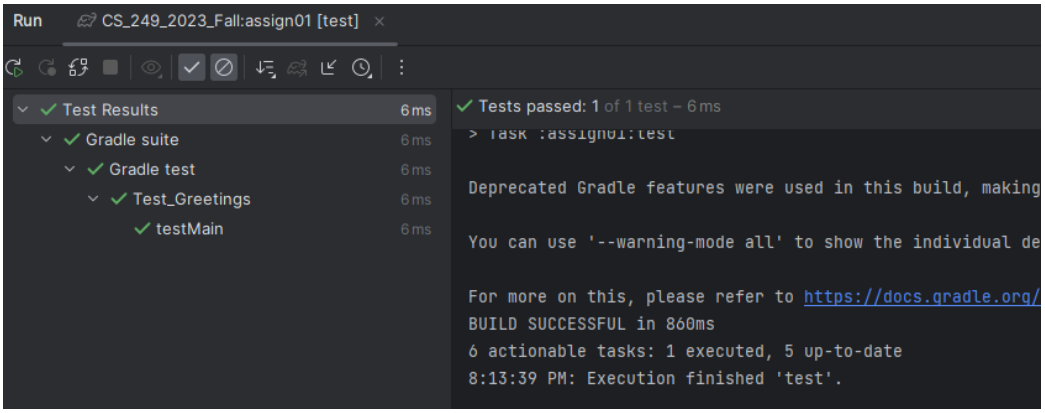
    - **Modify assignXX/src/main/java/module-info.java file**
        - Replace all references to “realemj” with *yourname*
    - **Modify the test code in src/test/java**
        - Change the “realemj” part of package folder to yourname
            - Right-click on package folder → “Refactor” → “Rename…”
            - This should also change the package line in the testing files themselves
        - Do NOT change:
            - import edu.realemj.testing.*;
            - import edu.realem.testing.GeneralTesting;
    - **In build.gradle, change value of mainClass variable to refer to *yourname* instead of “realemj”**
    - **Reload all Gradle projects**
    - **Right-click on the verification:test task for the new assignment in the Gradle menu**
        - Select "Modify Run Configuration…"
        - Under "Modify Options", make sure "Run as test" is checked
        - Hit "Apply" and "OK"
- **Complete assignment code and make commits to this new branch**
    - “Git” → “Commit”
    - Feel free to push commits up to the repository as well: “Git” → “Push”

Testing Screenshot

In addition to the code submission, you MUST submit a screenshot of the test results.

- **You MUST run the test files and send a screenshot of the test results!**
    - Even if your program(s) do not pass all the tests, you MUST send this screenshot!
- This screenshot should show clearly what tests have passed (or not).
- **WARNING: If any code does not compile, tests will NOT run!**
- **Include the screenshot inside the subproject for the assignment!**

Here is an example of an acceptable test screenshot:



UML Diagrams

- **For the UML diagrams, use UMLet: https://www.umletino.com/**
- **You MUST submit the diagram as a .png image file!**
    - **Do not ONLY submit a UMLet .uxf file (although if that is included, that's fine)**
- **Make sure your methods have the EXACT names and parameters as described!**
- **You must use the UML syntax defined in the slides!**
- Unless otherwise stated, submit everything as ONE diagram.
- Unless a method is overridden, you do NOT need to repeat a parent's method in a child class' diagram.
- **Include the UML diagram image inside the subproject for the assignment!**

Completing and Submitting an Assignment

Once you have finished development on your branch, do the following:

- **Commit any changes to your branch**
    - "Git" → "Commit"
- **Checkout the "main" branch**
    - "Git" → "Branches" → click on "main" → "Checkout"
- **Merge your branch into the "main" branch**
    - "Git" → "Merge" → select your branch for the assignment
- **Commit as necessary and push changes to remote repository**
    - "Git" → "Push"

Be sure to do these steps BEFORE the assignment deadline!

Grading

Apart from the specific grading breakdown, I reserve the right to take points off for not meeting the specifications in the assignment description.

Every assignment will have a different grading breakdown. Do NOT assume it is always the same!

For the PROGRAMMING part, these penalties will be in place unless otherwise specified:

Issue	***Penalty (in %)***
Code that does not compile	60
Using static class data where instance data is required	20
Non-private/non-protected class data	10
Printout/prompt format incorrect **(including spaces and capitalization)**	5
Files not in correct package	5
Submission code not merged with main branch (but exists in repository)	5
New branch not created or not used for assignment code development	5
Code files/classes/methods named incorrectly	10
Other stuff submitted (class files, etc.) BUT NO CODE SUBMITTED	100
Nothing submitted at all	100

In general, these are things that will be penalized:

- **Code that does not compile**
- Poor abstraction and/or encapsulation
- Sloppy or poor coding style
- Bad coding design principles
- Code that crashes, does not run, or takes a VERY long time to complete
- Using code from ANY source other than the course materials
- Collaboration on code of ANY kind; this is an INDIVIDUAL PROJECT
- Sharing code with other people in this class or using code from this or any other related class
- Output that is incorrect
- Algorithms/implementations that are incorrect
- Submitting improper files
- Failing to submit ALL required files