CE Getting Started

Due Jan 12 by 11:59pm **Points** 10 **Submitting** a text entry box or a file upload
Available until Jan 19 at 11:59pm

This assignment was locked Jan 19 at 11:59pm.

Fundamentals

CE: Getting Started

Learning Objectives

- Set up the programming environment needed for this course.

Overview

This CE consists of two parts.
Part 1 demonstrates how to set up your Java projects to include algs4.jar, the Java programming environment provided by Prof. Sedgewick and his team. Part 2 introduces you to CodePost, an auto-grader that allows you to obtain early feedback on your assignments and to improve your code before submitting it on Canvas.

Instruction

Part 1 Programming Environment

- Create a Java Project called **2420_ProgrammingEnvironment**

You might see a dialog that asked you for a module name.
Click "Don't Create" because we don't need modules in this course.

- Create a package called **test**
- Add a class called **Hello**, that prints Hello World
- Run the program and verify that everything works so far.
- Add a second class called **TestAlgs4** and copy the following code:
  http://algs4.cs.princeton.edu/windows/TestAlgs4.java.html
  (http://algs4.cs.princeton.edu/windows/TestAlgs4.java.html)
  The code doesn't compile because TestAlgs4 uses classes that can't be found. We need to provide a jar file that includes those classes.
- Let's start by adding a folder called **lib** to the project. You can do that by right-clicking the project 2420_ProgrammingEnvironment > new > Folder and call it lib.
- Download **algs4.jar** **(https://slcc.instructure.com/courses/773686/files/126485215/download?download_frd=1)** and copy it into the folder lib. You can drag the file algs4.jar from the download folder in the lib folder (Eclipse). A dialog will ask you whether you want to copy the file. Select copy.
  At this point, you added the folder with the jar file to the project, and yet the squiggly lines are still there. There is still another step we need to do. We need to let the compiler know that algs4.jar needs to be included in the compilation.
- Add algs4.jar to the build path.
  Right-click the project 2420_ProgrammingEnvironment > Build Path > Add External Archives
  Navigate to the algs4.jar file in the lib folder, select it, and click Open.
  At this point, the Package Explorer in Eclipse should show 'Referenced Libraries' with algs4.jar.
  Still, the red squiggly lines are still there.
- Add the missing import statements.
  Roll your mouse over StdOut. Eclipse will provide multiple quick-fix options. Choose the one that imports StdOut from edu.princeton.cs.algs4. If you can't see it, you might need to scroll down for more quick-fix options (even if the scroll bar is not visible right away)
  Do something analogous with other classes that need import statements.
- Run the code.

Part 2 CodePost

- Create a second package inside the project and call it **gettingStarted.**
- Download the class **Calculator.java** **(https://slcc.instructure.com/courses/773686/files/126485196/download?download_frd=1)** and add it to the package gettingStarted.
  Notice that the last method that should calculate the absolute value is not properly

implemented yet. Please leave it as it is. We use it to demonstrate how to access and interpret the auto-generated feedback from CodePost.

- Sign up for CodePost
  Look for an email from CodePost in your bruinmail account. Follow the instructions there to sign up. If you encounter difficulties, read the **student sign-up instructions (https://help.codepost.io/en/articles/3655946-student-signup-instructions)** from suggestions.
- Once you are signed into CodePost, you will see an assignment called TryOutCodePost. Click 'Upload assignment' to submit the file Calculator.java. Look at the auto-generated feedback.
  - How many tests failed?
  - Where in the feedback can you find information about the method(s) that caused the tests to fail?
  - Can you find more detailed information besides the method name that can help you troubleshoot the code?
- Fix the code, then click 'Add/Update files' to resubmit the updated code at CodePost.

Here is a video that introduces CodePost, describes how to sign up, demonstrates how to submit an assignment, and shows how to interpret the test results.
https://youtu.be/lsXHj5RFMcQ (https://youtu.be/lsXHj5RFMcQ)

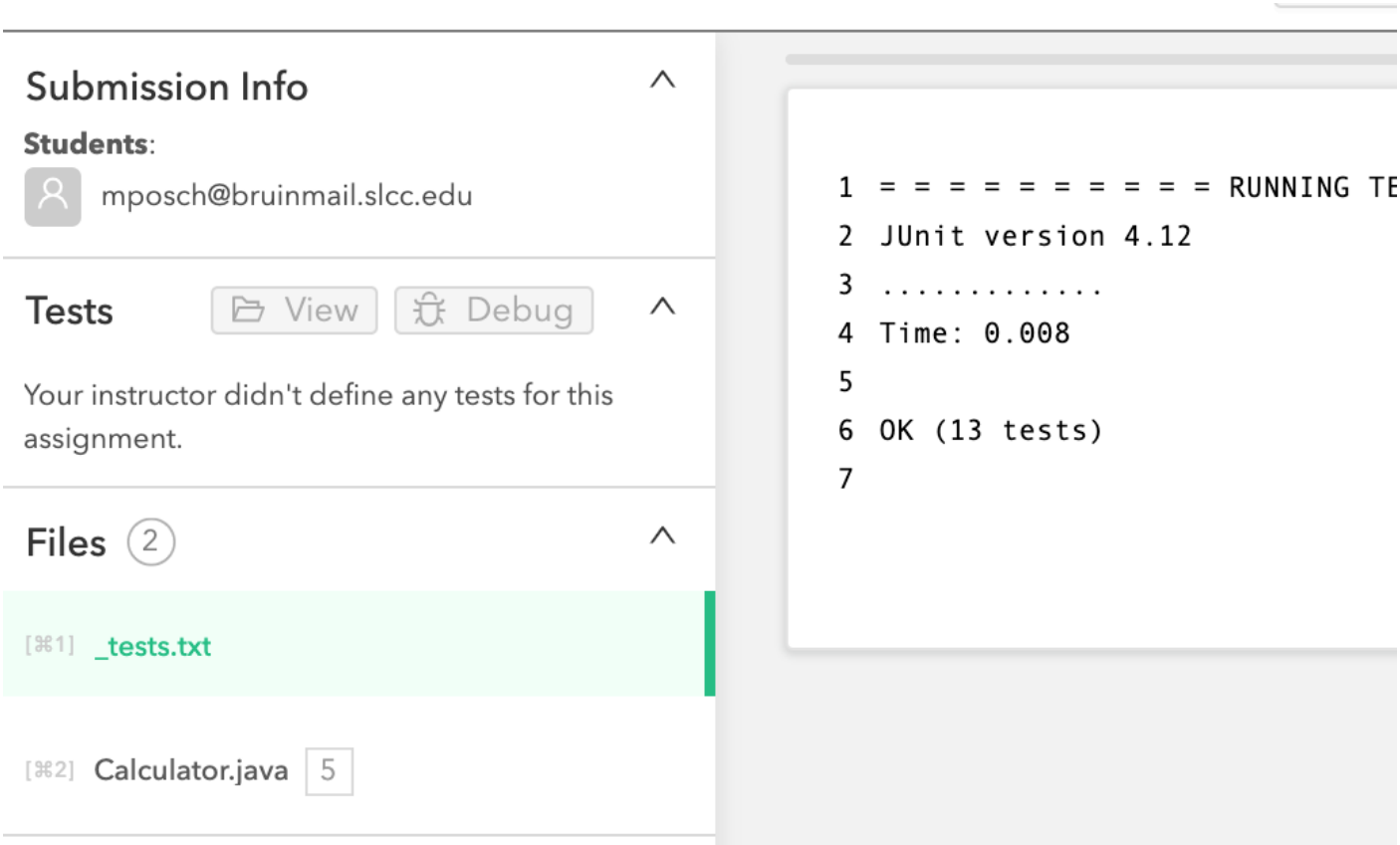
(https://youtu.be/lsXHj5RFMcQ)

Expected Output

Part 1:

Part 2:

After fixing the code, the CodePost feedback should look similar like the screenshot below. Notice, how it shows the bruinmail email and the file names on the left and the auto-grader feedback on the right.



Submission

Create a screen recording following the [guidelines for lab recordings](#).

The video should be **20-40 seconds** long.
Post the video.

(https://slcc.instructure.com/courses/773686/modules/items/16910403)

(https://slcc.instructure.com/courses/773686/modules/items/16910404)

(https://slcc.instructure.com/courses/773686/modules/items/16910407)

(https://slcc.instructure.com/courses/773686/modules/items/16910408)

CE Getting Started						
Criteria	**Ratings**					**Pts**
Code Environment Create a Java Project as described. Add a folder that includes algs4.jar and add it to the build path. Running the program results in the expected output.	**5 pts** **Full Marks**	**4 pts** **Close**	**3 pts** **Getting There**	**2 pts** **More work is needed**	**0 pts** **Missing or insufficient**	5 pts
CodePost Sign up for CodePost, submit the provided Java file, interpret the auto-generated feedback, fix the code, and submit again. The feedback should look similar to the one shown under Expected Output.	**5 pts** **Full Marks**	**4 pts** **Close**	**3 pts** **Getting There**	**2 pts** **More work is needed**	**0 pts** **Missing or insufficient**	5 pts
						Total Points: 10