A.I. For Software Testing and Reverse Engineering CS4110

This document provides instructions on to setup the Dev Container for Visual Studio Code (VS Code). The Dev Container allows you to attach VS Code onto the docker container and you can directly develop and run your solution from VS Code. For this setup, you would only need to build the JAR once. Please read the instructions carefully in this document. If there are any questions regarding the setup, please drop by during lab hours or send us a message on Mattermost.

Prerequisites

To get the Dev Container working on VS Code, you will need the following installed on your machine:

- Latest version of Docker for your OS - https://www.docker.com/
- VS Code for your OS - https://code.visualstudio.com/Download
- Remote Extension Pack for VS Code - Remote Extension Pack
- (Optional) Docker image used by the course - AISTR Docker Image

The last requirement is optional, as VS Code will automatically download the image if it not already downloaded on your machine.

NOTE: We have tested this setup on all three OSes (Linux, MacOS and Windows) and confirmed that it was working. We also tried this setup on GitHub's codespaces (cloud-based development environment).

Starting and Running the Dev Container

Step 1. Cloning the repository
To get the Dev Container running, first clone the JavaInstrumentation repository to your host machine by running the following command:

```
git clone https://github.com/apanichella/JavaInstrumentation.git
```

If you have already cloned the repository, you can pull the latest changes.

2. Starting container with VS Code
Once you have cloned the repository (or pulled the latest changes), you can open the JavaInstrumentation folder using VS Code as shown in Figure 1 (screenshot from MacOS).

Once the folder is opened, VS Code will recognize that there is a Dev Container configuration for the repository and a pop-up will appear on the right-bottom corner asking whether you want to open the container in a container. An example of a pop-up that is shown in MacOS can be seen in Figure 2.

Click on the “Reopen in Container” and VS Code will create a new docker container using the docker image of this course. Don’t worry about specifying the right image, we have already handled this for you ☺. **You do** have to make sure that Docker is running in the background so that VS Code can start the container. If docker is not running, you will see a pop-up from VS Code.

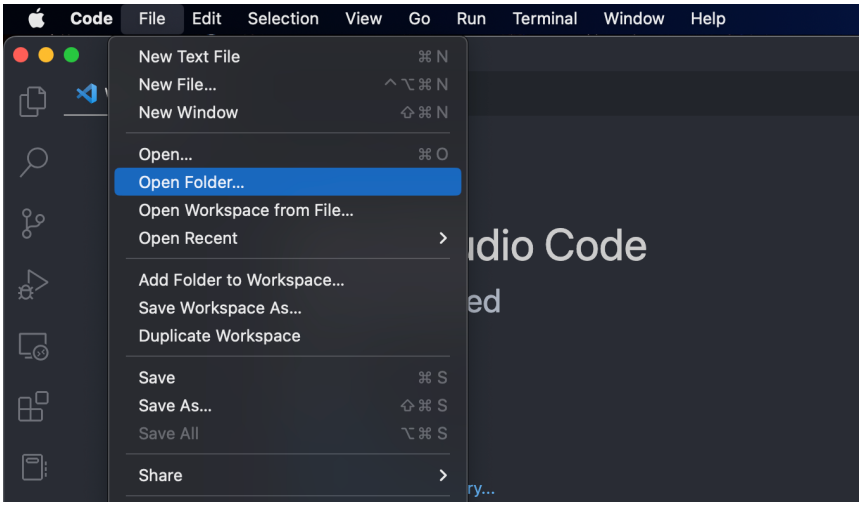

Figure 1. Opening Folder in VS Code.

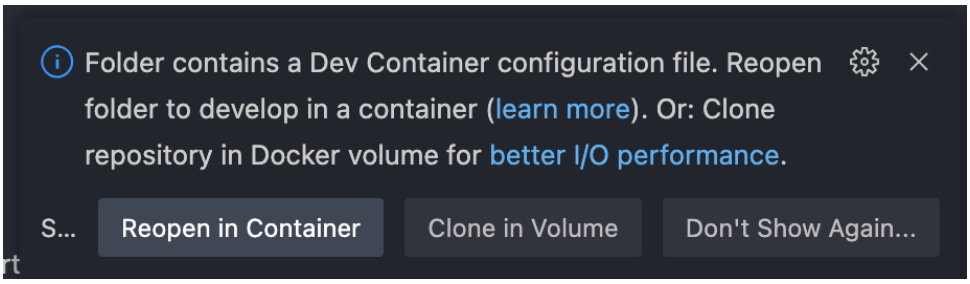

Figure 2. Example pop-up asking whether you want to open folder in a container.

Step 3. Run task to instrument RERS Problem files

Once the container has been started (please be patient), you can run the task of instrumenting the RERS problem file. Don’t worry about creating the task yourself, we have already done this too for you ☺. To run a task, go to Terminal > Run Task… (see Figure 3).

Clicking on “Run Task…” will give you a bunch of options, select the one that says “Instrument all” (see Figure 4). After selecting “Instrument all”, there will be a prompt that ask you which type of instrumentation you would like to use (see Figure 5). After making a selection, VS Code will run “mvn clean package”, instrument the RERS problem files using the selected instrumentation type and also compile the instrumented problem files. The instrumented problem files are listed in the “instrumented” folder.

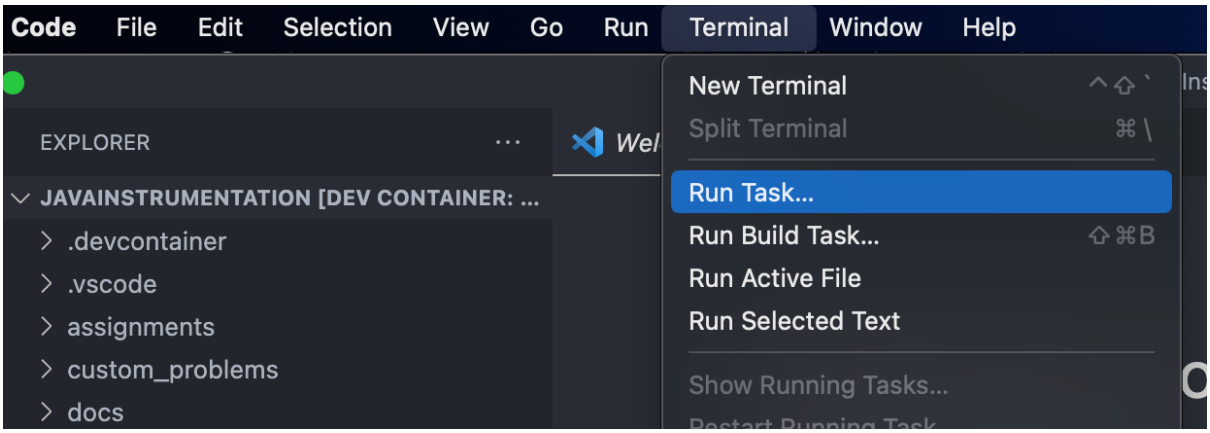

Figure 3. Running a task on VS Code.

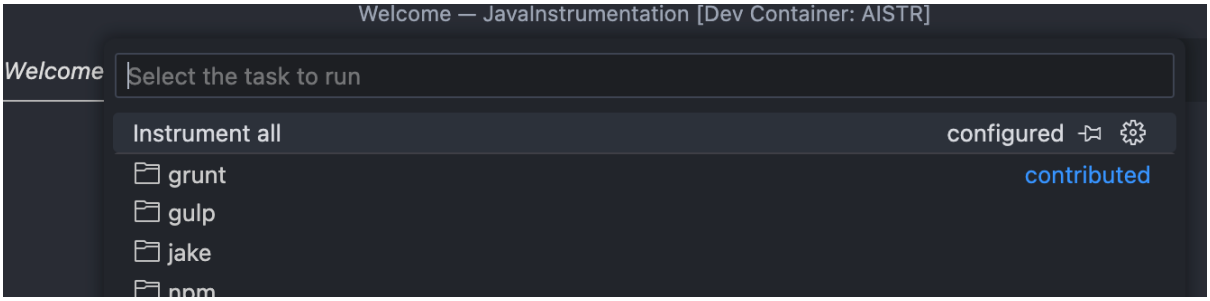

Figure 4. Prompt showing which tasks can be run. You need to select “Instrument all”.

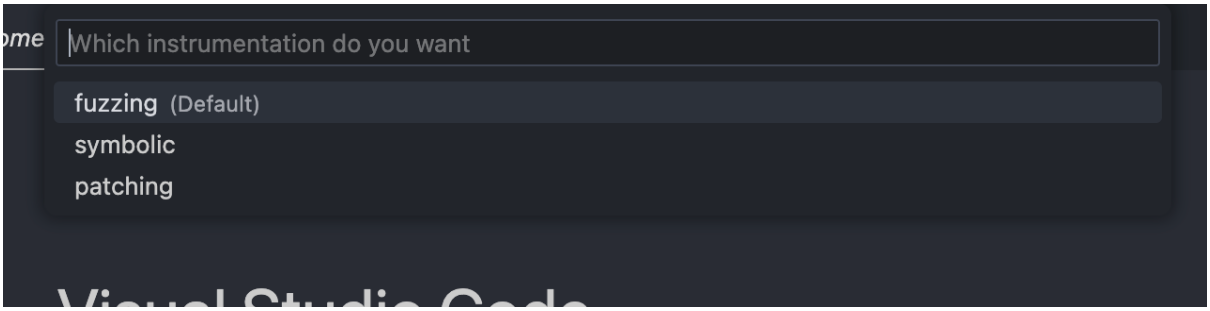

Figure 5. Prompt asking which type of instrumentation should be run.

Running an instrumented RERS Problem

Step 1. Navigate to the Run and Debug tab

To run an instrumented RERS Problem, first navigate to the the “Run and Debug” in the menu shown on the left of VSCode. The icon of this tab is shown in Figure 6.

Step 2. Select Launch Problem or Launch Problem 11

On top next to the run button, you should see a dropdown menu. You can select either "Launch Problem" or "Launch Problem 11" (see Figure 6). The latter only runs Problem 11 and the former provides you the option to select a problem that you want to run (see Figure 7).

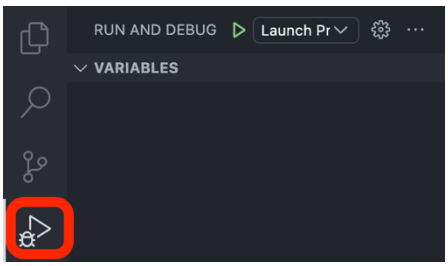

Figure 6. "Run and Debug" Tab.

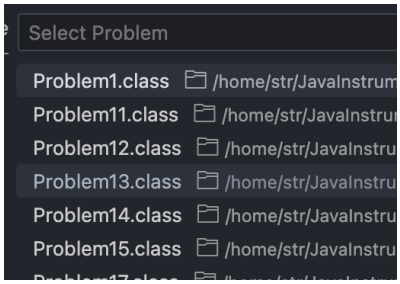

Figure 7. Selecting problem to run.

Clicking on the green arrow would start running the selected problem. You should see the output on the terminal of VS Code.

Syncing changes with Dev Container

The advantage of using a Dev Container is that you VS Code automatically creates a volume for you, so there's no need to create yourself by running any commands.

Debugging on Dev Container

The Dev Container allows you to add breakpoints in your program to debug your code. Adding a breakpoint can be done in the same manner as with any IDE. The only difference is that you would need to declare the variables that you want to inspect in the "Watch" tab on the left. Figure 8 and 9 shows adding a breakpoint and inspecting the "currentTrace" variable, respectively.

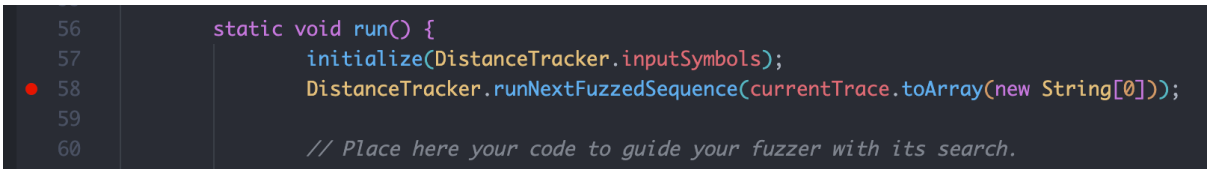

Figure 8. Adding a breakpoint to inspect "currentTrace" variable.

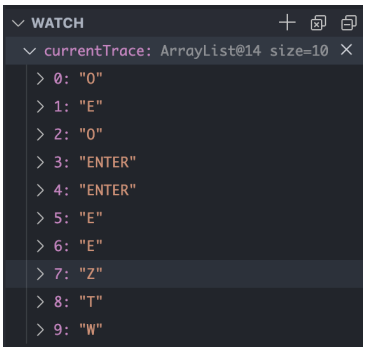

Figure 9. Inspecting "currentTrace" variable.