Additional documentation for the Computer Science Project 2022

Disclaimer

This document serves as an **Unofficial** supplementary information document for the Computer science project. This does not override the information on the official website and any unofficial scripts and information will not be valid to acquire additional marks. To clarify, if your project works after following the information and scripts contained here but does not work with the official scrips, you will not be granted the marks.

Contents

The official documentation is available on the website here.

Overview

The first set you need to take in order to get the testers up and running is downloading the tester files. This can be done on the official website or via this link or you can download the modified files from here. Once you have downloaded the files you must extract them into the same file directory as your project. Below is an example of how your directory might look.

```
your-project-folder-name-here
├── files
│   ├── 1_1_moves.txt
│   ├── 1_1_out.txt
│   ├── 1_1_params.txt
│   ├── ....
│   ├── 5_5_moves.txt
│   ├── 5_5_out.txt
│   └── 5_5_params.txt
├── runone.sh
├── StdAudio.java
├── StdDraw.java
├── StdIn.java
├── StdOut.java
├── SUxxxxxxxx.java    < This is your project
├── testall.sh
└── testone.sh
```

Please be aware that if the bash (.sh) files are not in the same folder as your program (SUXXXXXXXX.java) then they will not work.

For details on individual test cases please consult the website. This document will only serve to explain how to use the scripts.

The modified files

The modified files by the one and only Kirbs can be found here. They are not majorly different from the original testing scripts but contain some quality of life features. They will automatically compile your program for you and should be able to work on all machines as long as you follow the methods listed in the OS sections. You will still have to elevate their privileges and manually enter in your project name but after that you're sorted for good.

On the GitHub repository click on Code and Download .zip then install it normally.

Linux

After following the steps in the overview you can proceed with the following.

You will need to open the files in a text editor and edit the first line from `SUXXXXXXXX` to your .java project name.

In order to successfully run the scripts from the command line you will need to give them the required permissions. First open a terminal using "Ctrl+Alt+T" or by right clicking in your folder and selecting open terminal. Then you need to make sure you are in the correct directory. Use the `cd` command to navigate to the correct location like this `cd ~/path/to/directory`

You will then need to give the files adequate permissions in order to run the properly. Use the command: `chmod +x testone.sh testall.sh runone.sh` Note this only needs to be done once.

At this point you will be able to execute the testing scripts by using their name as the command like `./testone.sh 2 3`.

Understanding the arguments for the scripts
understanding the outputs if the scripts

Windows

After following the steps in the overview you can proceed with the following.

I would highly recommend you read this web page on how to run bash scripts on windows. This would allow you to easily follow the Linux steps.

You may also be required to add the "shebang" `#!/bin/bash/` to the beginning of your test scripts.

At this point you will be able to execute the testing scripts by using their name as the command like `./testone.sh 2 3`.

Understanding the arguments for the scripts
understanding the outputs if the scripts

Mac

After following the steps in the overview you can proceed with the following.

I would highly recommend you read this web page on how to run bash scripts on windows. This would allow you to easily follow the Linux steps.

You will be required to add the "shebang" `#!/bin/bash/` to the beginning of your test scripts and you will have to rename them from `.sh` to `.bash`

At this point you will be able to execute the testing scripts by using their name as the command like `./testone.bash 2 3`.

Understanding the arguments for the scripts
understanding the outputs if the scripts

Understanding the arguments for the scripts

Note: In order to use the scripts you need to compile your program first, use the `javac` command to do so.

There are three different scripts, each allowing you to do something different. Runone allows you to run the program with the inputs for the given arguments. Testone performs a test on your program for te given arguments. Testall takes no arguments and tests all categories.

The command line arguments for the scripts may seem unintuitive at first but are really quite simple. The first argument details the Category of testing and the second argument details the case for said category.

The link to the catagories on the web page can be found here.

- First hand in categories
    1. Argument validation
    2. Input validation
    3. Blockade detection
    4. Win
    5. Row deletion (Bonus Mark)

Details on the cases can be found on the website. Each category tests a different aspect of your program, as listed above. This allows you to target a specific section of your code until you get it right.

Understanding the output of the scripts

The output of the tester seems very daunting at first and is very unintuitive to read but here are some tips to help you understand what it means.

You will need to recompile your program every time that you make changes so I would recommend downloading the modified test files here which will automatically do that for you.

At the beginning the output will first tell you what category and case is being tested, then what the arguments are that are being used. Following that are the moves, this will be a list of all of the moves that the tester will make (Eg. 2 - Quit). Finally the comparison will be shown.

The best case

> $ ./testone.sh 1 1

```
Category 1, case 1
Arguments:
  3 0 2 3
Moves:
  2
Comparison:
Files - and files/1_1_out.txt are identical
```

Running a test case and seeing this is the best case scenario, this means that the given output and the expected output are identical and that case was successful.

The normal case

> $ ./testone.sh 1 1

```
Category 1, case 1
Arguments:
  3 0 2 3
Moves:
  2
Comparison:
5,12c5,12
< ,*******
< ,*******
< ,*******
< ,*******
< ,*******
< ,*******
< ,*******
< ,*******
---
> .*******
> .*******
> .*******
> .*******
> .*******
```

```
  > .*******
  > .*******
  > .*******
```

The < left arrow denotes the output that your program produced, the > right arrow denotes the expected output. In this case the given output was a comma instead of a full stop.