COMP90041
Programming and Software Development
2020 - Semester 2
Lab 3

Yuhao(Beacon) Song

yuhsong1@unimelb.edu.au

yuhsong@student.unimelb.edu.au

Introduction

- **Timetable**
  - Tue(18) 16:15-17:15 (Melbourne Time)
    - https://unimelb.zoom.us/j/94854648719?pwd=WUY0NmR6MkI5UVZBUWhGNWFIU216Zz09
  - Wed(19) 17:15-18:15 (Melbourne Time)
    - https://unimelb.zoom.us/j/93723434766?pwd=MGh4QkJuZnhJR21LZ0VqeXhJQU52UT09

- **GitHub Page (Tutorial Materials, Solutions, Additional Resources)**
  - https://github.com/Beaconsyh08/COMP90041-2020SEM2

- **PollEv**
  - https://pollev.com/yuhsong

Outline

- Git
- Lecture Review
- Exercise & demo

Git & GitHub

- **Git** is a distributed **version-control system** for tracking changes in source code during software development.

-- https://en.wikipedia.org/wiki/Git

- **GitHub** provides hosting for software development and version control using **Git**.

-- https://en.wikipedia.org/wiki/GitHub

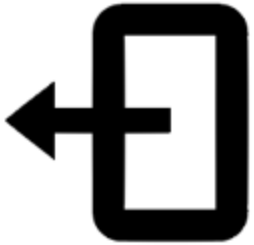
Git Basic

- **add**
  - **Puts** current working files into the **stage** (aka index or cache)
- **commit**
  - **Commits** staged changes to a **local** branch
- **push**
  - **Uploads** changes from all **local** branches to the respective **remote** repositories.

In case of fire

1. git commit

2. git push



3. leave building

Git Tools

- **Git Bash/Terminal**: If you prefer the **Command-line interface** and want to looks more **geek**. (Downloaded with Git)
- **Git GUI**: achieve the operations with **GUI** (Downloaded with Git)

- **GitHub Desktop**: similar to Git GUI, make clone, fork… from **GitHub** more smooth.
  - https://desktop.github.com/
- **IDE VCS**: For example, **IntelliJ** VCS let you connect to git directly. Others IDE could have the same functions
  - https://www.jetbrains.com/idea/download/

- There are lot more tools you could use as well: https://git-scm.com/downloads/guis
- Git: https://git-scm.com/downloads

Tutorial Q1

Exercise 1a: Histogram of temperatures

Write a program that reads in temperatures (in Celsius) for five days, that is, from Monday to Friday and plots a histogram showing the temperatures. The name of your class should be `Temperatures`. Given below is a sample run of the program.

```
Please enter temperature for Monday: 25
Please enter temperature for Tuesday: 33
Please enter temperature for Wednesday: 26
Please enter temperature for Thursday: 28
Please enter temperature for Friday: 20

Histogram of Temperatures
-------------------------
Monday      | **************************
Tuesday     | *********************************
Wednesday   | **************************
Thursday    | ****************************
Friday      | ******************
```

Exercise 1b: Input and Output Redirection

You will be given a sample test input file temperatures.txt and the corresponding sample output file `test0-output.txt`. When you run your program by the following command in a terminal (or Windows command line):

```
java Temperatures <  temperatures.txt > my-output.txt
```

your program should produce a file name `my-output.txt`which should be exactly the same as `test0-output.txt`. In this command, "`<`temperatures.txt" and "`> my-output.txt`" are called "input redirection" and "output redirection." They use the content in temperatures.txtas the command line input, and print the program output into `my-output.txt`.

Exercise 1c: Submission

Once your program produces the correct output on your local machine, go ahead and submit your projects via GitHub.

You should always test your code locally on your machine before pushing it into your repository. Once you are confident that your program compiles correctly you can commit your changes and push your code to your repository. As soon as you submit your code, the GitHub server will start running automated on your code and compare your code's output to what it is expecting. This process can take a few minutes. You can view the results online. Note that you can submit as many times as you like to test your code.

How you edit, compile and run your Java program is up to you. You are free to use any editor or development environment. However, you need to ensure that your program compiles and runs correctly with java's version 1.8.0.

While Statement

- The **while statement** evaluates expression, which must return a **boolean** value. If the expression evaluates to **true**, the while statement executes the statement(s) in the while block. The while statement continues testing the expression and executing its block until the expression evaluates to **false**.

- **Difference**
  - **while** evaluates its expression at the **top** of the loop
  - **do-while** evaluates its expression at the **bottom** of the loop
  - Therefore, the statements within the **do block** are always **executed at least once**

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html

The break and continue Statements

- The **break statement** consists of the keyword **break** followed by a semicolon
  - When executed, the **break statement** ends the **nearest** enclosing **switch** or **loop** statement

- The **continue statement** consists of the keyword **continue** followed by a semicolon
  - When executed, the **continue statement** ends the **current loop** body iteration of the **nearest** enclosing loop statement
  - Note that in a for loop, the **continue statement** transfers control to the **update expression**

- When loop statements are nested, remember that any **break** or **continue** statement applies to the innermost, containing loop statement

Pitfall: Using == with Strings

- The equality comparison operator (==) can correctly test two values of a **primitive type**
- However, when applied to two objects such as objects of the **String class**, == tests to see if they are stored in the same memory location, not whether or not they have the same value → (Skip until lab5)
- In order to test two strings to see if they have equal values, use the method equals, or equalsIgnoreCase

string1.equals(string2)

string1.equalsIgnoreCase(string2)

Primitive Types

Display 1.2 **Primitive Types**

TYPE NAME	KIND OF VALUE	MEMORY USED	SIZE RANGE
`boolean`	`true` or `false`	1 byte	not applicable
`char`	single character (Unicode)	2 bytes	all Unicode characters
`byte`	integer	1 byte	−128 to 127
`short`	integer	2 bytes	−32768 to 32767
`int`	integer	4 bytes	−2147483648 to 2147483647
`long`	integer	8 bytes	−9223372036854775808 to 9223372036854775807
`float`	floating-point number	4 bytes	$-3.40282347 \times 10^{+38}$ to $-1.40239846 \times 10^{-45}$
`double`	floating-point number	8 bytes	$\pm 1.76769313486231570 \times 10^{+308}$ to $\pm 4.94065645841246544 \times 10^{-324}$

Tutorial Q2

Exercise 2: Traffic Infringements

The traffic section of a Police Department wishes to automate the writing of warnings, fines etc. to motorists who exceed the 60km/hr speed limit and whether doing it under influence of liquor or not. Your task is to implement the following warning and fines in the program based on the corresponding conditions:

Condition	*Message(s)*
> 60 and <65	Warning
>60 and <65 and drunk	Warning + Take a shower
65 to <= 70	$5 fine for each km/hr over 60 km/hr
65 to <= 70 and drunk	$7 fine for each km/hr over 60 km/hr + Take a shower
> 70	$10 fine for each km/hr over 60 km/hr
> 70 and drunk	$15 fine for each km/hr over 60 km/hr Spend the day/night in cell until become sober

The program should ask the traffic officer to type in the km/hr speed of the offending driver. It should then ask whether driver is drunk or not. (The officer answers with a ‘y’ or ‘n’ and the appropriate message is then given.) The program should then display the appropriate message and where any fine is applicable, the program should compute and display the fine.

Sample Run 1

```
Please enter speed: 64
Is the driver drunk? ('Y' for drunk, 'N' otherwise): N

**************************************************
Warning

--------------------------------------------------
You have a fine of $0.0
**************************************************
```

Sample Run 2

```
Please enter speed: 64
Is the driver drunk? ('Y' for drunk, 'N' otherwise): Y

**************************************************
Warning + Take a shower

--------------------------------------------------
You have a fine of $0.0
**************************************************
```

Sample Run 3

```
Please enter speed: 85
Is the driver drunk? ('Y' for drunk, 'N' otherwise): Y

**************************************************
$15.0 fine for each km/hr over 60 km/hr

Spend the day/night in cell until become sober.

--------------------------------------------------
You have a fine of $375.0
**************************************************
```

Git Useful Resources

- **Git Tutorial:**
  - https://www.atlassian.com/git/tutorials/what-is-git
- **Git Official Doc:**
  - https://git-scm.com/doc
- **Git For IntelliJ:**
  - https://www.jetbrains.com/help/idea/set-up-a-git-repository.html