2021

CSCI5308 Assignment 1

DHRUMIL AMISH SHAH (B00857606)
DH416386@DAL.CA

Initial Steps Performed

1. Cloned the repository https://git.cs.dal.ca/courses/2021-summer/csci-5308/assignment1/ronnie.git in my local machine.
2. Changed the folder name from "**ronnie"** to "**assignment_1"**.
3. Made minor changes
    - Added CityTest.java file (Test class for City.java).
    - Override compareTo(Link o) in Link.java file.
    - Formatted all the files. (IntelliJ – Ctrl + Alt + L).
4. Changed the remote origin to my repository using command
    - git remote set-url origin https://git.cs.dal.ca/courses/2021-summer/csci-5308/assignment1/dashah
5. Added all changes to the staging area.
    - git add .
6. Committed all changes
    - git commit -m "<commit_message>"
7. Pushed all the code to my remote repository (i.e., to dashah)
    - git push

Testing and Commit Approach

For commits, I followed a simple strategy.

1. Start
2. Select a method to test.
    - Example: getLength() method of the Link class.
3. Write down the required test cases for that method in the respective file (TDD Approach).
    - Example: For the getLength() method of the Link class, write test cases in LinkTest class.
4. Test the actual method. If the test succeeds, go to step 5 otherwise, go to step 6.
5. Commit changes to the local repository and push the changes to the remote repository and go to step 7.
6. If the test failed, improve the logic, and go to step 4.
7. End.

⇨ Repeat this approach for all the methods in all the classes. It ensures thorough testing of all methods.

Insights

Classes **City** and **CityComparator** have one method in common.
i.e., public int compare(City c1, City c2). According to the comments provided, the working of these methods is the same.

The compare(City x, City y) method in the **CityComparator** class calls the compare(City c1, City c2) method in the **City** class. It creates confusion. It is not sure whether the **City** class should implement the **Comparator<City>** interface and override the compare() method just like the **CityComparator** class or extend the **CityComparator** class and override the compare method or keep it standalone as it is now. Moreover, the test cases for these methods are also the same.

What I did

I kept it standalone as it is now. I have neither implemented the **Comparator<City>** interface nor extended the **CityComparator** class. For individual comparison, this method is called directly like c1.compare(c1, c2), where c1 and c2 are instances of **City** class. For **CityComparator** class, this method is called like this compare(x, y). i.e., for use in sorting or data structures.

Suggestion
Calling the method c1.compare(c1, c2) of the **City** class like this creates confusion. We are passing c1 twice. i.e., using the "this" keyword and the other using "c1" explicitly. Instead, we can change the method definition to compareByDistance(City c2) and call it like, c1.compareByDistance(c2).

So, in **City** class, code looks like

```
public int compareByDistance(City c2) {
   return this.distance - c2.distance;
}
```

In **CityComparator** class, code looks like

```
public int compare(City x, City y) {
   return x.compareByDistance(y);
}
```

It is now easy to understand. From the method name itself, it is clear that we are comparing cities based on their distance.

Improvements

- The access modifier of all variables is currently **public**. It is better to have variables **private** instead of **public** and provide **getter** and **setter** methods. It ensures data encapsulation and results in a strong object-oriented approach.
  - i.e., Access modifiers of variables in class **City** and **Link** should be changed to **private** from the **public**. Also, appropriate **getters** and **setters** should be provided for these variables.

Code Documentation

- I used Java documentation comment to write the comments. It improves the readability of the code.