Ammar Rafiqui (218931410)

MD Ahnaf Hyder (218207159)

Rigzin Shawootsang (217242124)

Kevin Dao (218118190)

Yash Dani (218462531)

EECS 3311

Alvine Boaye Belle

2024-04-06

Report of Comparison Between Test Cases

Overall Code Coverage:

We wrote our test cases to cover specific functions and or edge cases in mind so that they can cover the majority of the critical points of the code but in the process of doing so, our manual test cases can pass over more general cases that are often overlooked which can be fixed by employing Randoop generated test cases to cover any possible inputs that do not come to mind when creating specific test cases. As such randoop is used to maximize code coverage by systematically going through each possible input. We wrote our test cases to assess specific features of each class based on realistic user-based scenarios, as such they can be easily understandable and useful for system testing. An example of the comparison between the manual and generated test cases can be found in the SystemInventory class where a manual test case is written specifically to evaluate the functionality of adding and removing items from the class in a flow similar to a real user interaction with the system. On the other hand, the Randoop test case doesn't showcase a user interaction or any functionality but rather the system's technical ability to handle operations without losing any integrity.

Overall Mutation Score:

The mutation score can vary significantly between manually written test cases and the ones generated by Randoop. Since they are used to recognize the system's ability to find faults, the manual tests must be written by an experienced tester who deeply understands the system as otherwise rare and critical issues can be left unchecked. Randoop however can fill in these gaps through its generated scenarios that can cover various complex mutations leading to a much higher score than manual tests.

Overall Readability:

As an actual person writes manual test cases, they will always be more readable and more understandable as they have a clear structure to them defined by comments, goals and fitting names that are descriptive of their purpose. This in the long term will make them easier to maintain and repurpose if needed. In contrast, the Randoop-generated test cases will often be generic and will not employ good code structure, making understanding its purpose and the content it is testing hard for users. Our SystemInventory class has clear names such as testSearchItem and testManageItemDetailsWithExistingItemID while randoop test case names are "test01".

```
@Test
void testManageItemDetailsWithExistingItemID() throws Exception {
```

```
@Test
public void test03() throws Throwable {
```

Overall Usefulness:

As discussed in the code coverage and readability section, manual written test cases will always be the most useful method of testing as it will take in the tester's understanding of the system to cover the most critical and frequented areas of the system. For long-term planning, Randoop-generated test cases that are generic and hard to read will be difficult for testers to understand while manual written test cases will have clear documentation about their purpose.

Effectiveness Comparison:

In terms of bug detection, manually written tests are usually more effective at catching complex, business-logic-related bugs because they are written with specific scenarios in mind. Manual testing is superior at identifying user interface and user experience issues, including layout problems, usability issues, and accessibility barriers as it often requires a human perspective to evaluate effectively. On the other hand, Randoop-generated tests can more effectively catch a broad range of regressions and edge cases that human testers might not consider. It is superior in terms of coverage and more effective in revealing errors, and identifying flaky methods and non-deterministic behavior.

Direct Comparison:

```
@Test
void testRemoveItem() {
   SystemInventory si = new SystemInventory(new ArrayList());
   PhysicalItemDetails item = new PhysicalItemDetails();
   item.setItemID(1);
   si.addItem(item);
   si.removeItem(1);
   Assertions.assertTrue(SystemInventory.itemList.isEmpty());
}
```

```
@Test
public void test04() throws Throwable {
    if (debug)
        System.out.format("%n%s%n", "RegressionTest0.test04");
    java.lang.Object obj0 = new java.lang.Object();
    java.lang.Class<?> wildcardClass1 = obj0.getClass();
    org.junit.Assert.assertNotNull(wildcardClass1);
}
```

Code Coverage: Manual test case tests adding and removing functionality while generated test case covers creating objects and getting its class offering little coverage.

Mutation Score: The manual test case has a higher score as it tests the actual specific functionality of the system than the generated test case.

Readability: Both are readable but the manual test case is straightforward and clear.

Usefulness: A manual test case is more useful as it targets a user-orientated scenario while the generated one doesn't target anything useful for testing the system.

Combining Their Strengths:

Both manual-written test cases and Randoop-generated test cases have their strengths and weaknesses which is why it's oftentimes better to integrate them to create better overall code coverage. Any system functionality and specific scenarios can be captured by manual testing while niche and rare edge cases overlooked by testers can be generated through randoop. More often than not, the generated test cases can offer insights into new test cases.

Conclusion:

Both methods of test cases have their advantages and challenges, manual test cases feature more clarity and deep coverage of user-oriented scenarios but may overlook the simpler features. Generated test cases will broaden the overall coverage to find mutations and other hidden edge cases but at the cost of readability and direct usability. Using them both results in your test cases being stronger without any of their weaknesses attached to your system.

Name of the team member	Tasks completed by the team member	Participation of the team member
Ammar Rafiqui	Wrote the test cases for all the classes underneath the package UserData and SystemInventory .	20% (for a team of 5)
MD Ahnaf Hyder	Wrote the report for all of the reports about the test cases.	20% (for a team of 5)
Rigzin Shawootsang	Wrote test cases for classes under PaymentSystem, RentingVisitorData, and ItemsData.	20% (for a team of 5)
Kevin Dao	Created test cases for classes under NewsLetterSystem, MenuOptions and NotificationSystem, got code coverage for project.	20% (for a team of 5)
Yash Dani	Generated Randoop test cases for all classes in the system and contributed in the comparison reports.	20% (for a team of 5)