**First name: John Stephen**

**Lastname: Gutam**

**SWE637 -Take Home Final GNumber: G01413212**

1. **Faults, failures, and errors.**

**Vehicle/Truck classes**

****

A computer code with black text

Description automatically generated with medium confidence

1. **Explain what is wrong with the given code. Describe the fault precisely by proposing a modification to the code.**

The fault in the given code is that clone() method in the Truck class is throwing a ClassCastException when trying to cast the cloned object(Vehicle) to Truck. This is because the clone() method in the Vehicle class is returning an Object of type Vehicle instead of Truck.

To fix this, we can modify the clone() method in the Vehicle class to return a Vehicle object and override the clone() method in the Truck class to return a Truck Object. This can be done by using the super.clone() by casting it to appropriate type.

1. **If possible, give a test case that does not execute the fault. If not, briefly explain why not.**

Vehicle a = new Vehicle (6);

Vehicle b = (Vehicle) a.clone(); //return as expected

1. **If possible, give a test case that executes the fault, but does not result in an error state. If not, briefly explain why not.**

If Super calls clone then no error. If Sub class clones then it’s a error.

1. **If possible give a test case that results in an error, but not a failure. If not, briefly explain why not. Hint: Don’t forget about the program counter.**

It is not possible to have a test case that results in an error but not a failure. The ClassCastException is an error and would result in a failure.

**(f) Implement your repair and verify that the given test now produces the expected output. Submit a screen printout or other evidence that your new program works.**

1. **(A)**

**(e) In the given code, describe the first error state. Be sure to describe the complete state.**

**Test: Truck suv = new Truck(4);**

**Truck co = suv.clone();**

PC : ((Truck)result).y = this.y;

y = 4;

result: Super object

The clone() method in the Vehicle class is returning an Object of type Vehicle instead of Truck which leads to ClassCastException.

**(f) Implement your repair and verify that the given test now produces the expected output. Submit a screen printout or other evidence that your new program works.**

**A screenshot of a computer

Description automatically generated**

**Point/ColorPoint classes**

A screenshot of a computer program

Description automatically generated

1. **Explain what is wrong with the given code. Describe the fault precisely by proposing a modification to the code.**

The fault in the given code is that equals() method in the ColorPoint class violates the symmetry property of the equals() method. According to the equals() contract, if a.equals(b) is true, then b.equals(a) is also true. However in the given code p.equals(cp1) return true, but cp1.equals(p) returns false. super.equals(cp) && (cp)

To fix this, we can modify the equals() method in the ColorPoint class to check if the Object being compared is an instance of the Point class. If it is we can create the ColorPoint object with the same coordinates and default color, and then compare it with the o object using equals.

1. **If possible, give a test case that does not execute the fault. If not, briefly explain why not.**

Point p = new Point(1,2);

ColorPoint cp1 = new ColorPoint(1,2,RED);

p.equals(cp1); // Test: Result = true;

1. **If possible, give a test case that executes the fault, but does not result in an error state. If not, briefly explain why not.**

ColorPoint cp1 = new ColorPoint(1,2,RED);

ColorPoint cp2 = new ColorPoint(1,2,BLUE);

cp1.equals(cp2); //Test: Result= false;

1. **If possible give a test case that results in an error, but not a failure. If not, briefly explain why not. Hint: Don’t forget about the program counter.**

It is not possible to have a test case that results in an error but not a failure.

1. **In the given code, describe the first error state. Be sure to describe the complete state.**

PC : Location B

cp1: [1,2,RED]

p: [1,2]

result: cp1.equals(p) gives false.

1. **Implement your repair and verify that the given test now produces the expected output. Submit a screen printout or other evidence that your new program works.**

To fix this, we can modify the equals() method in the ColorPoint class to check if the Object being compared is an instance of the Point class. If it is we can create the ColorPoint object with the same coordinates and default color, and then compare it with the o object using equals.

A screen shot of a computer program

Description automatically generated

**(B)**

**A screenshot of a computer code

Description automatically generated**

**The problem is that in BigDecimal, equals() and compareTo() are inconsistent. For the sake of this quiz, suppose we decide that equals() is correct, and that compareTo() is faulty.**

1. **At the end of this computation, does set t have the correct value?**

No. t does not have the correct value.

1. **At the end of this computation, does set s have the correct value?**

Yes. s does have the correct value.

1. **Describe the state after the first call to t.add(). Hint: Don’t forget about the program counter.**

PC = t.add(x), x = 1.0, t = [1.0]

1. **Is this state an error state? If so, describe what is wrong.**

No. As per the assumption made that compareTo() is faulty, TreeSet ‘t’ stores t = [1.0]

1. **Describe the state after the second call to t.add().**

PC = t.add(y), x = 1.0, y = 1.00, t = [1.0]

1. **Is this state an error state? If so, describe what is wrong.**

Yes. As per the assumption made that compareTo() is faulty, TreeSet ‘t’ is not storing 1.0 and 1.00. Hence t contains t = [1.0]

1. **Test Automation.**
2. **Number 3**

**A screenshot of a computer

Description automatically generated**

**Number 8**

A screenshot of a computer

Description automatically generated

1. **TakeHomeFinal-Gutam.java**

A screenshot of a computer

Description automatically generated

1. **Flaky Tests**
3. The test cBasicProxyAuthentication is flaky. Because it is connecting through the proxy. If proxy fails the test also fails.

**A screenshot of a computer

Description automatically generated**

1. I observed the test to be failed by changing the password from ‘p4ssw0rd’ to ‘p4ssw0r’. Below is the screenshot of the test failure and the exception.

**A screenshot of a computer

Description automatically generated**

**Exception:**

**A screenshot of a computer code

Description automatically generated**

1. The specific test method in HttpRequestTest.java that is flaky is cBasicProxyAuthentication().

This test method is flaky because it relies on an external proxy server to be running on the localhost. If the proxy server is not running or if there are any issues with the proxy server, the test may fail. Therefore, the test results may vary depending on the availability and stability of the proxy server.

The other two test methods, aCustomConnectionFactory() and bNullConnectionFactory(), do not appear to be flaky as they do not rely on external dependencies and should consistently produce the expected results**.**

HttpRequestTest.java falls under the category of Test Order Dependency. It refers to tests that rely on a specific order of execution or are affected by the order in which other tests are run. In this case, the cBasicProxyAuthentication() test may be affected by the availability and stability of the external proxy server, which can vary depending on the order in which tests are executed or the state of the proxy server.

2. Test t2() is flaky.
3. Test t1() is the polluter for t2().
4. Test t3() is the cleaner for t2().
5. (t1 t2 t3), (t3 t1 t2).
6. The probability for a random order of TestClassOne is 1/3