

SWE637 -Take Home Final**1. Faults, failures, and errors.****(A)****Vehicle/Truck classes**

```

class Vehicle implements Cloneable
{
    private int x;
    public Vehicle (int y) { x = y;}
    public Object clone()
    {
        Object result = new Vehicle (this.x);
        // Location "A"
        return result;
    }
    // other methods omitted
}
class Truck extends Vehicle
{
    private int y;
    public Truck (int z) { super (z); y = z;}
    public Object clone()

    {
        Object result = super.clone();
        // Location "B"
        ((Truck) result).y = this.y; // throws ClassCastException
        return result;
    }
    // other methods omitted
}
//Test: Truck suv = new Truck (4); Truck co = suv.clone()
//      Expected: suv.x = co.x; suv.getClass() = co.getClass()

```

Note: Relevant to Bloch, Item 11 page 54.

Book website: Vehicle.java, Truck.java, CloneTest.java

(a) 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.

(b) 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

(c) 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.

(d) 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

(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.

```
1 package com.assignment;
2
3 class Vehicle implements Cloneable {
4     protected int x;
5
6     public Vehicle(int y) {
7         x = y;
8     }
9
10    public Object clone() {
11        try {
12            return super.clone();
13        } catch (CloneNotSupportedException e) {
14            throw new AssertionError();
15        }
16    }
17 }
18 class Truck extends Vehicle{
19     private int y;
20
21    public Truck(int z) {
22        super(z);
23        y = z;
24    }
25
26    public Object clone() {
27        Truck result = (Truck) super.clone();
28        result.y = this.y;
29        return result;
30    }
31
32    public static void main(String[] args) {
33        Vehicle a = new Vehicle(5);
34        System.out.println(a.x);
35        Vehicle b = (Vehicle) a.clone();
36        System.out.println(b.x);
37        Truck c = new Truck(7);
38        System.out.println(c.x);
39        System.out.println(c.y);
40        Truck d = (Truck)c.clone();
41    }
42 }
```

Markers Properties Servers Data Source Explorer Snippets Terminal Console ×

<terminated> AllTests (1) [Java Application] C:\Program Files\Java\jdk1.8.0_202\bin\javaw.exe (Nov 28, 2023, 4:23:19 PM – 4:23:21 PM) [pid: 2136]

5
5
7
7

Point/ColorPoint classes

```
class Point
{
    private int x; private int y;
    public Point (int x, int y) { this.x=x; this.y=y; }

    @Override public boolean equals (Object o)
    {
        // Location A
        if (!(o instanceof Point)) return false;
        Point p = (Point) o;
        return (p.x == this.x) && (p.y == this.y);
    }
}
class ColorPoint extends Point
{
    private Color color;
    // Fault: Superclass instantiable; subclass state extended

    public ColorPoint (int x, int y, Color color)
    {
        super (x,y);
        this.color = color;
    }
}
```

```
@Override public boolean equals (Object o)
{
    // Location B
    if (!(o instanceof ColorPoint)) return false;
    ColorPoint cp = (ColorPoint) o;
    return (super.equals(cp) && (cp.color == this.color));
}
// Tests:
Point p = new Point (1,2);
ColorPoint cp1 = new ColorPoint (1,2,RED);
ColorPoint cp2 = new ColorPoint (1,2,BLUE);
p.equals (cp1); // Test 1: Result = true;
cp1.equals (p); // Test 2: Result = false;
cp1.equals (cp2); // Test 3: Result = false;
// Expected: p.equals (cp1) = true; cp1.equals (p) = true,
//           cp1.equals (cp2) = false
```

Note: Relevant to Bloch, Item 17 page 87.

Book website: Point.java, ColorPoint.java, PointTest.java

- (a) 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.

- (b) 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;
```

- (c) 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;
```

- (d) 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.

- (e) 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.
```

- (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.**

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.

```
class ColorPoint extends Point {
    private Color color;

    public ColorPoint(int x, int y, Color color) {
        super(x, y);
        this.color = color;
    }

    @Override
    public boolean equals(Object o) {
        if (o instanceof ColorPoint) {
            ColorPoint cp = (ColorPoint) o;
            return super.equals(cp) && (cp.color == this.color);
        } else if (o instanceof Point) {
            Point p = (Point) o;
            ColorPoint cp = new ColorPoint(p.x, p.y, Color.DEFAULT); // Create a ColorPoint obj
            with default color
            return super.equals(cp);
        }
        return false;
    }
}
```

8 pts.) Consider the code:

```
BigDecimal x = new BigDecimal("1.0");
BigDecimal y = new BigDecimal("1.00");
// Fact: !x.equals(y), but x.compareTo(y) == 0

Set <BigDecimal> s = new HashSet <BigDecimal> ();
s.add(x);
s.add(y);
// HashSet uses equals(), so s now has 2 elements

Set <BigDecimal> t = new TreeSet <BigDecimal> ();
t.add(x);
t.add(y);
// TreeSet uses compareTo(), so t now has 1 element

System.out.println("s = " + s);
System.out.println("t = " + t);
```

You may find it useful to know that the contract for the `add()` method in the `Set` interface states:

Adds the specified element to this set if it is not already present (optional operation). More formally, adds the specified element `e` to this set if the set contains no element `e2` such that `(e==null ? e2==null : e.equals(e2))`. If this set already contains the element, the call leaves the set unchanged and returns `false`. In combination with the restriction on constructors, this ensures that sets never contain duplicate elements.

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.

(2.) At the end of this computation, does set `s` have the correct value?

Yes. `s` does have the correct value.

(3.) 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]`

(4.) 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]`

(5.) Describe the state after the second call to `t.add()`.

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

(6.) 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]`

2. Test Automation.

A. Number 3

The screenshot displays an IDE window with a Java test class named `BoundedQueueTest.java`. The code includes imports, a class definition, and several test methods. The `testEnQueueFull()` method is currently selected. Below the code editor, the `JUnit` tab shows the execution results of the tests. A green progress bar indicates that all tests passed. The list of tests includes `testEnQueue`, `testEnQueueFull`, `testEnQueueNull`, `testToString`, `testDeQueueEmpty`, `testDeQueue`, `testIsFull`, and `testIsEmpty`, all with a duration of 0.000 s.

```
1 import org.junit.Assert;
2
3 /*
4  * Fullname: John Stephen Gutam
5  * Gnumber: G01413212
6  */
7
8 public class BoundedQueueTest {
9
10     private BoundedQueue queue;
11
12     @Before
13     public void setUp() {
14         queue = new BoundedQueue(3);
15     }
16
17     @Test
18     public void testEnQueue() {
19         queue.enqueue("test");
20         Assert.assertEquals("[test]", queue.toString());
21     }
22
23     @Test(expected = NullPointerException.class)
24     public void testEnQueueNull() {
25         queue.enqueue(null);
26     }
27
28     @Test(expected = IllegalStateException.class)
29     public void testEnQueueFull() {
30         queue.enqueue("test1");
31         queue.enqueue("test2");
32         queue.enqueue("test3");
33         queue.enqueue("test4");
34     }
35
36     @Test
37     public void testDeQueue() {
38
39     }
40 }
```

JUnit Results:

- testEnQueue (0.002 s)
- testEnQueueFull (0.000 s)
- testEnQueueNull (0.000 s)
- testToString (0.000 s)
- testDeQueueEmpty (0.000 s)
- testDeQueue (0.000 s)
- testIsFull (0.000 s)
- testIsEmpty (0.000 s)

Number 8

```
1 import java.util.*;
2 public class Min
3 {
4
5 public static <T extends Comparable<? super T>> T min (List<? extends Comparable<?>> list)
6 {
7     if (list.size() == 0)
8     {
9         throw new IllegalArgumentException ("Min.min");
10    }
11
12    Iterator<? extends T> itr = (Iterator<? extends T>) list.iterator();
13    T result = itr.next();
14
15    if (result == null) throw new NullPointerException ("Min.min");
16
17    while (itr. hasNext())
18    {
19        T comp = itr.next();
20        if (comp.compareTo (result) < 0)
21        { // throws NPE, CCE as needed
22            result = comp;
23        }
24    }
25
26    return result;
27 }
28
29 }
30
31 }
32
33 }
34
35 }
```

Markers Properties Servers Data Source Explorer Snippets Terminal Console JUnit ×

Finished after 0.016 seconds

Runs: 4/4 Errors: 1 Failures: 0

MinTest [Runner: JUnit 4] (0.000 s)

- > [0] (0.000 s)
- > [1] (0.000 s)
- > [2] (0.000 s)
- > [3] (0.000 s)
- testMin[3] (0.000 s)

Failure Trace

- java.lang.IllegalArgumentException: Min.min
- at Min.min(Min.java:9)
- at MinTest.testMin(MinTest.java:31)

B. TakeHomeFinal-Gutam.java

MinTest.java Minjava TakeHomeFinal.java ×

```
1 import static org.junit.Assert.assertFalse;
2 import static org.junit.Assert.assertTrue;
3 import static org.junit.Assume.assumeTrue;
4
5 import java.util.Set;
6 import java.util.HashSet;
7
8 import org.junit.Before;
9 import org.junit.Test;
10 import org.junit.experimental.theories.Theory;
11 class EH {
12     private int x;
13     public EH(int x) { this.x = x; }
14
15     @Override public boolean equals(Object obj) {
16         if (!(obj instanceof EH)) return false;
17         return ((EH) obj).x == this.x;
18     }
19
20     // @Override public int hashCode() { return x; }
21 }
22 public class TakeHomeFinal/*-LName*/ {
23     private EH eh1; private EH eh2; private EH eh3;
24     private Set<EH> s;
25
26     @Before public void setUp() { eh1 = new EH(3); eh2 = new EH(5); eh3 = new EH(3); s = new HashSet<EH>(); }
27
28     @Test
29     public void NoNPE() { // Question 2
30         EH obj = new EH(3);
31         assertFalse(obj.equals(null));
32     }
33
34     @Test
35     public void equalsFalse() { // Question 2
36         // ...
37     }
38 }
```

Markers Properties Servers Data Source Explorer Snippets Terminal Console JUnit ×

Finished after 0.028 seconds

Runs: 4/4 Errors: 0 Failures: 1

TakeHomeFinal [Runner: JUnit 4] (0.000 s)

- basicHash (0.000 s)
- equalsFalse (0.000 s)
- NoNPE (0.000 s)
- equalsTrue (0.000 s)

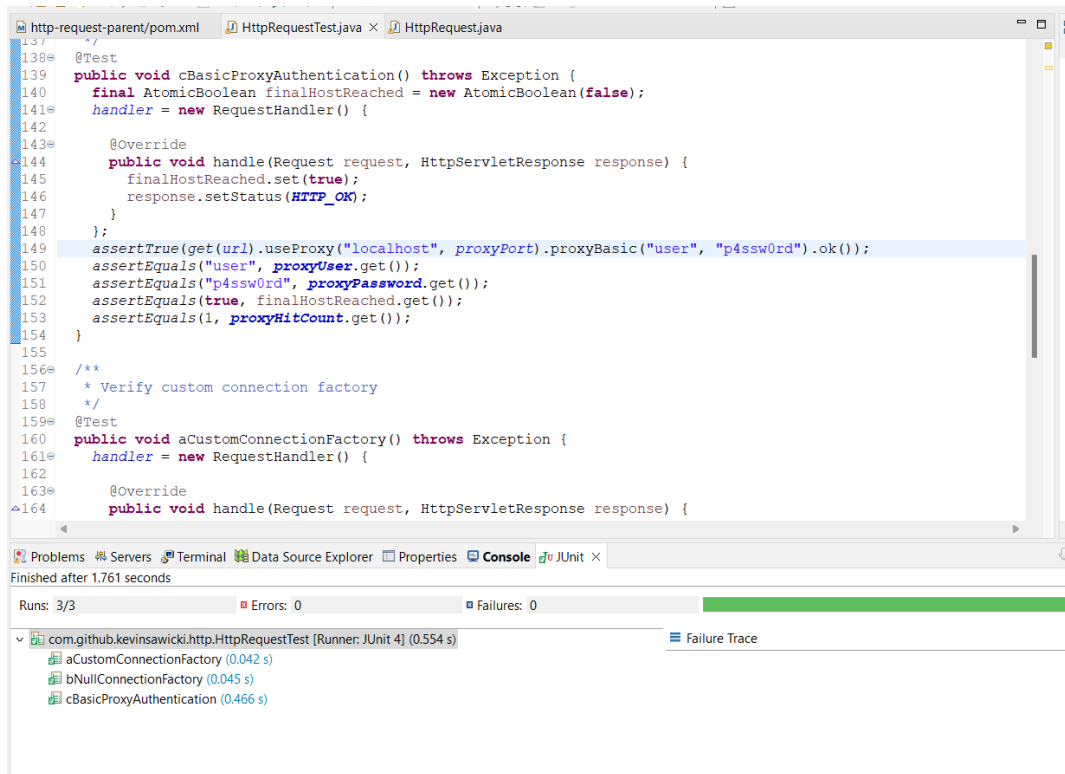
Failure Trace

```
java.lang.AssertionError
    at TakeHomeFinal.basicHash(TakeHomeFinal.java:46)
```


3. Flaky Tests

A.

1. The test `cBasicProxyAuthentication` is flaky. Because it is connecting through the proxy. If proxy fails the test also fails.

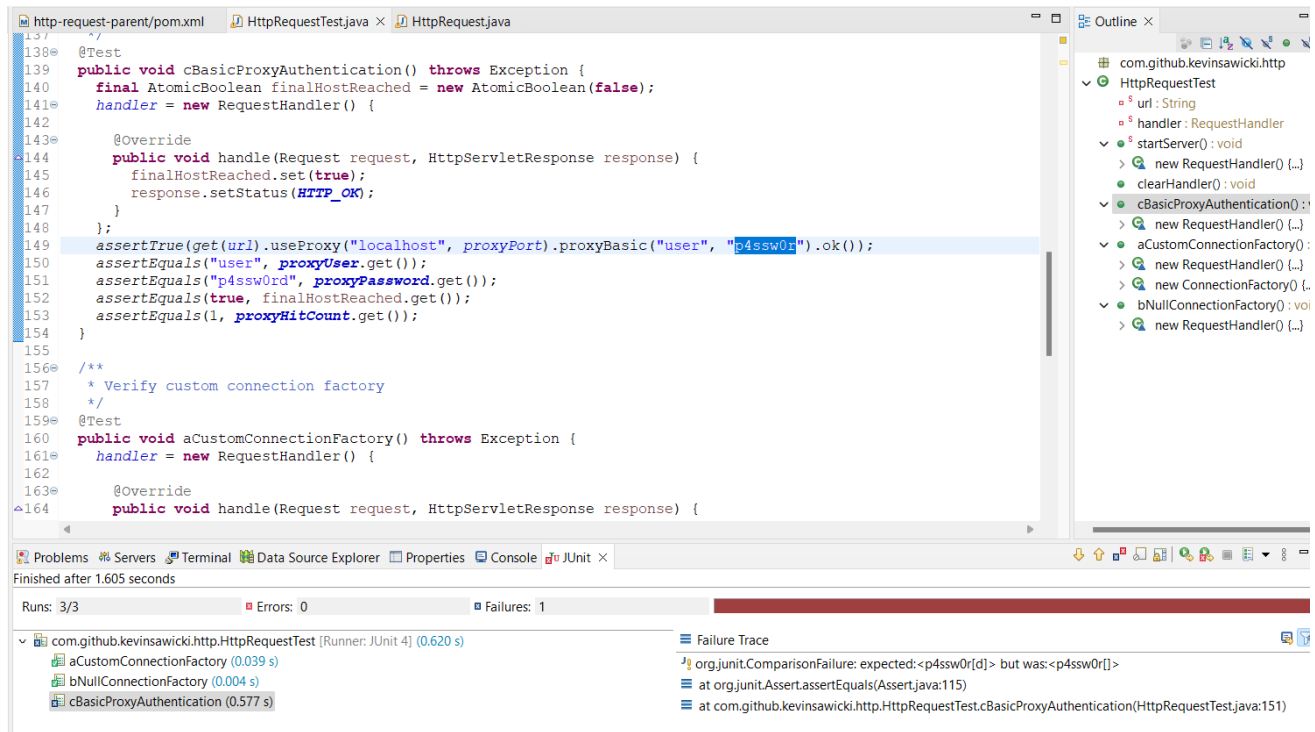


The screenshot shows an IDE with the `HttpRequestTest.java` file open. The test `cBasicProxyAuthentication` is highlighted. The test code is as follows:

```
138 @Test
139 public void cBasicProxyAuthentication() throws Exception {
140     final AtomicBoolean finalHostReached = new AtomicBoolean(false);
141     handler = new RequestHandler() {
142
143         @Override
144         public void handle(Request request, HttpServletResponse response) {
145             finalHostReached.set(true);
146             response.setStatus(HTTP_OK);
147         }
148     };
149     assertTrue(get(url).useProxy("localhost", proxyPort).proxyBasic("user", "p4ssw0rd").ok());
150     assertEquals("user", proxyUser.get());
151     assertEquals("p4ssw0rd", proxyPassword.get());
152     assertEquals(true, finalHostReached.get());
153     assertEquals(1, proxyHitCount.get());
154 }
155
156 /**
157  * Verify custom connection factory
158  */
159 @Test
160 public void aCustomConnectionFactory() throws Exception {
161     handler = new RequestHandler() {
162
163         @Override
164         public void handle(Request request, HttpServletResponse response) {
```

The IDE's bottom panel shows the test results: "Runs: 3/3", "Errors: 0", "Failures: 0". The test `cBasicProxyAuthentication` is listed with a duration of 0.466 s.

2. 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.



The screenshot shows the same IDE with the `HttpRequestTest.java` file open. The test `cBasicProxyAuthentication` is highlighted, but the password in the code is now `"p4ssw0r"`. The IDE's bottom panel shows the test results: "Runs: 3/3", "Errors: 0", "Failures: 1". The test `cBasicProxyAuthentication` is listed with a duration of 0.577 s.

The failure trace is shown in the bottom right panel:

```
org.junit.ComparisonFailure: expected:<p4ssw0r[d]> but was:<p4ssw0r[]>
    at org.junit.Assert.assertEquals(Assert.java:115)
    at com.github.kevinsawicki.http.HttpRequestTest.cBasicProxyAuthentication(HttpRequestTest.java:151)
```

Exception:

Failure Trace



```
org.junit.ComparisonFailure: expected:<p4ssw0r[d]> but was:<p4ssw0r[]>  
    at org.junit.Assert.assertEquals(Assert.java:115)  
    at com.github.kevinsawicki.http.HttpRequestTest.cBasicProxyAuthentication(HttpRequestTest.java:151)
```

3. 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.

B.

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

