CS 4501/6501: In-class 1
Fault, Error, Failure Model

24-Aug-2017

Names:

Purpose: Understand the difference between fault, error, and failure. Practice for the homework.
Instruction: Work with your neighbors in groups. Consider some faulty programs. I will put the code up for viewing on the screen. Focus on answering the questions:

```
line  1  public static int findLast(int[] x, int y)
line  2  {
line  3     if (x == null)
line  4        throw new NullPointerException();
line  5     for (int i=x.length-1; i>0; i--)
line  6     {
line  7        if (x[i] == y)
line  8           return i;
line  9     }
line 10     return -1;
line 11   }
// test: x = [2, 3, 5]; y = 2;
// expected = 0
```

1. **What is the fault?**

   The for-loop should include the 0 index

   ```
   for (int i=x.length-1; i>=0; i--)
   ```

2. **If possible, find a test input that does not reach (i.e., not execute) the fault.**

   A null value for x will result in a NullPointerException before the loop is reached. Thus, no execution of the fault.

   ```
   Input:           x = null; y = 3
   Expected output: NullPointerException
   Actual output:   NullPointerException
   ```

3. **If possible, find a test input that reaches the fault, but does not result in an error.**

   For any input where y appears in the second or later position, there is no error. Also, if x is empty, there is no error.

   ```
   Input:           x = [2, 3, 5]; y = 3
   Expected output: 1
   Actual output:   1
   ```

   Let's look at the state

   <u>A faulty program</u>

   ```
   < x={2, 3, 5}, y=3, PC=[if (x==null), (line 3)] >
   < x={2, 3, 5}, y=3, PC=[i=x.length-1, (line 5)] >
   < x={2, 3, 5}, y=3, i=2, PC=[i>0, (line 5)] >
   < x={2, 3, 5}, y=3, i=2, PC=[if (x[i]==y), (line 7)] >
   ```

```
< x={2, 3, 5}, y=3, i=2, PC=[i--, (line 5)] >
< x={2, 3, 5}, y=3, i=1, PC=[i>0, (line 5)] >
< x={2, 3, 5}, y=3, i=1, PC=[if (x[i]==y), (line 7)] >
< x={2, 3, 5}, y=3, i=1, PC=[return i, (line 8)] >
```

A correct program

```
< x={2, 3, 5}, y=3, PC=[if (x==null), (line 3)] >
< x={2, 3, 5}, y=3, PC=[i=x.length-1, (line 5)] >
< x={2, 3, 5}, y=3, i=2, PC=[i>=0, (line 5)] >
< x={2, 3, 5}, y=3, i=2, PC=[if (x[i]==y), (line 7)] >
< x={2, 3, 5}, y=3, i=2, PC=[i--, (line 5)] >
< x={2, 3, 5}, y=3, i=1, PC=[i>=0, (line 5)] >
< x={2, 3, 5}, y=3, i=1, PC=[if (x[i]==y), (line 7)] >
< x={2, 3, 5}, y=3, i=1, PC=[return i, (line 8)] >
```

The fault location is reached (i.e., the fault is executed). The program stops without state infection.

4. **If possible, find a test input that results in an error, but not a failure**

For an input where y is not in x, the missing path (i.e., an incorrect PC on the final loop that is not taken) is an error, but there is no failure.

```
Input:           x = [2, 3, 5]; y = 9
Expected output: -1
Actual output:   -1
```

Let's look at the state
A faulty program

```
< x={2, 3, 5}, y=9, PC=[if (x==null), (line 3)] >
< x={2, 3, 5}, y=9, PC=[i=x.length-1, (line 5)] >
< x={2, 3, 5}, y=9, i=2, PC=[i>0, (line 5)] >
< x={2, 3, 5}, y=9, i=2, PC=[if (x[i]==y), (line 7)] >
< x={2, 3, 5}, y=9, i=2, PC=[i--, (line 5)] >
< x={2, 3, 5}, y=9, i=1, PC=[i>0, (line 5)] >
< x={2, 3, 5}, y=9, i=1, PC=[if (x[i]==y), (line 7)] >
< x={2, 3, 5}, y=9, i=1, PC=[i--, (line 5)] >
< x={2, 3, 5}, y=9, i=0, PC=[i>0, (line 5)] >
< x={2, 3, 5}, y=9, PC=[return -1, (line 10)] >
```

A correct program

```
< x={2, 3, 5}, y=9, PC=[if (x==null), (line 3)] >
< x={2, 3, 5}, y=9, PC=[i=x.length-1, (line 5)] >
< x={2, 3, 5}, y=9, i=2, PC=[i>=0, (line 5)] >
< x={2, 3, 5}, y=9, i=2, PC=[if (x[i]==y), (line 7)] >
< x={2, 3, 5}, y=9, i=2, PC=[i--, (line 5)] >
< x={2, 3, 5}, y=9, i=1, PC=[i>=0, (line 5)] >
< x={2, 3, 5}, y=9, i=1, PC=[if (x[i]==y), (line 7)] >
< x={2, 3, 5}, y=9, i=1, PC=[i--, (line 5)] >
< x={2, 3, 5}, y=9, i=0, PC=[i>=0, (line 5)] >
< x={2, 3, 5}, y=9, i=0, PC=[if (x[i]==y), (line 7)] >
< x={2, 3, 5}, y=9, i=0, PC=[i--, (line 5)] >
< x={2, 3, 5}, y=9, i=-1, PC=[i>=0, (line 5)] >
< x={2, 3, 5}, y=9, i=-1, PC=[return -1, (line 10)] >
```

5. **Find a test input that results in a failure**

For an input where y is in the first position, there is an error (because the PC is outside the loop) and there is a failure.

```
Input:           x = [2, 3, 5]; y = 2
Expected output: 0
Actual output:   -1
```

Let's look at the state
A faulty program

```
< x={2, 3, 5}, y=2, PC=[if (x==null), (line 3)] >
< x={2, 3, 5}, y=2, PC=[i=x.length-1, (line 5)] >
< x={2, 3, 5}, y=2, i=2, PC=[i>0, (line 5)] >
< x={2, 3, 5}, y=2, i=2, PC=[if (x[i]==y), (line 7)] >
< x={2, 3, 5}, y=2, i=2, PC=[i--, (line 5)] >
< x={2, 3, 5}, y=2, i=1, PC=[i>0, (line 5)] >
< x={2, 3, 5}, y=2, i=1, PC=[if (x[i]==y), (line 7)] >
< x={2, 3, 5}, y=2, i=1, PC=[i--, (line 5)] >
< x={2, 3, 5}, y=2, i=0, PC=[i>0, (line 5)] >
< x={2, 3, 5}, y=2, PC=[return -1, (line 10)] >
```

A correct program

```
< x={2, 3, 5}, y=2, PC=[if (x==null), (line 3)] >
< x={2, 3, 5}, y=2, PC=[i=x.length-1, (line 5)] >
< x={2, 3, 5}, y=2, i=2, PC=[i>=0, (line 5)] >
< x={2, 3, 5}, y=2, i=2, PC=[if (x[i]==y), (line 7)] >
< x={2, 3, 5}, y=2, i=2, PC=[i--, (line 5)] >
< x={2, 3, 5}, y=2, i=1, PC=[i>=0, (line 5)] >
< x={2, 3, 5}, y=2, i=1, PC=[if (x[i]==y), (line 7)] >
< x={2, 3, 5}, y=2, i=1, PC=[i--, (line 5)] >
< x={2, 3, 5}, y=2, i=0, PC=[i>=0, (line 5)] >
< x={2, 3, 5}, y=2, i=0, PC=[if (x[i]==y), (line 7)] >
< x={2, 3, 5}, y=2, i=0, PC=[return i, (line 8)] >
```

6. **Identify the *first* error state for the given test**

The key aspect of the error state is that the PC is outside the loop (following the `false` evaluation of the 0 > 0 test). In a correct program, the PC should be at the if-test (line 7) with index `i=0`

First error state: PC = just before `return -1`