Fault/Failure Model

Software Testing

(3104313)

Amirkabir University of Technology

Spring 1399-1400

Faults, Failure, and Error

Fault: A **static** defect in the software.

Error: An **incorrect internal state** that is the manifestation of some fault.

Failure: **External, incorrect behavior** with respect to the requirements or another description of the expected behavior.

A Concrete Example

```
public static int numZero (int [ ] arr)
{  // Effects: If arr is null throw NullPointerException
   // else return the number of occurrences of 0 in arr
   int count = 0;
   for (int i = 1; i < arr.length; i++)
   {
      if (arr [ i ] == 0)
      {
         count++;
      }
   }
   return count;
}
```

A Concrete Example

Fault: Should start searching at 0, not 1

```
public static int numZero (int [ ] arr)
{  // Effects: If arr is null throw NullPointerException
   // else return the number of occurrences of 0 in arr
   int count = 0;
   for (int i = 1; i < arr.length; i++)
   {
      if (arr [ i ] == 0)
      {
         count++;
      }
   }
   return count;
}
```

A Concrete Example

Fault: Should start searching at 0, not 1

```
public static int numZero (int [ ] arr)
{  // Effects: If arr is null throw NullPointerException
   // else return the number of occurrences of 0 in arr
   int count = 0;
   for (int i = 1; i < arr.length; i++)
   {
      if (arr [ i ] == 0)
      {
         count++;
      }
   }
   return count;
}
```

Test 1

[2, 7, 0]

Expected: 1

Actual: 1

A Concrete Example

```
public static int numZero (int [ ] arr)
{  // Effects: If arr is null throw NullPointerException
   // else return the number of occurrences of 0 in arr
   int count = 0;
   for (int i = 1; i < arr.length; i++)
   {
      if (arr [ i ] == 0)
      {
         count++;
      }
   }
   return count;
}
```

Fault: Should start searching at 0, not 1

Test 1
[2, 7, 0]
Expected: 1
Actual: 1

Error: i is 1, not 0, on the first iteration
Failure: none

A Concrete Example

```
public static int numZero (int [ ] arr)
{  // Effects: If arr is null throw NullPointerException
   // else return the number of occurrences of 0 in arr
   int count = 0;
   for (int i = 1; i < arr.length; i++)
   {
      if (arr [ i ] == 0)
      {
         count++;
      }
   }
   return count;
}
```

Fault: Should start searching at 0, not 1

Test 1
[2, 7, 0]
Expected: 1
Actual: 1

Error: i is 1, not 0, on the first iteration
Failure: none

A Concrete Example

Fault: Should start searching at 0, not 1

```
public static int numZero (int [ ] arr)
{  // Effects: If arr is null throw NullPointerException
   // else return the number of occurrences of 0 in arr
   int count = 0;
   for (int i = 1; i < arr.length; i++)
   {
      if (arr [ i ] == 0)
      {
         count++;
      }
   }
   return count;
}
```

Test 1
[2, 7, 0]
Expected: 1
Actual: 1

Error: i is 1, not 0, on the first iteration
Failure: none

Test 2
[0, 2, 7]
Expected: 1
Actual: 0

A Concrete Example

Fault: Should start searching at 0, not 1

```
public static int numZero (int [ ] arr)
{  // Effects: If arr is null throw NullPointerException
   // else return the number of occurrences of 0 in arr
   int count = 0;
   for (int i = 1; i < arr.length; i++)
   {
      if (arr [ i ] == 0)
      {
         count++;
      }
   }
   return count;
}
```

Test 1
[2, 7, 0]
Expected: 1
Actual: 1

Error: i is 1, not 0, on the first iteration
Failure: none

Test 2
[0, 2, 7]
Expected: 1
Actual: 0

Error: i is 1, not 0
Error propagates to the variable count
Failure: count is 0 at the return statement

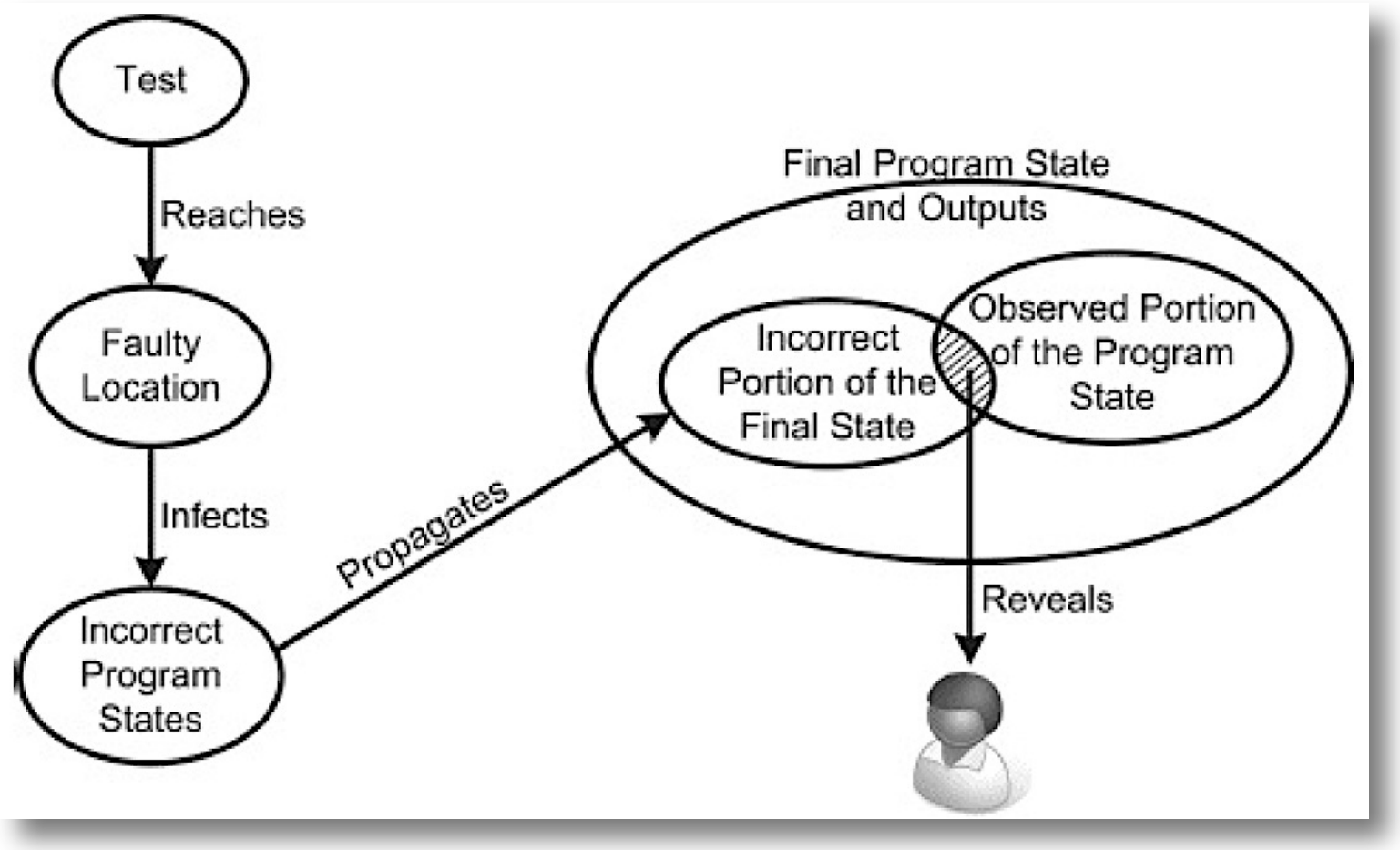
In-Class Exercise #4

Answer the following questions about the given program.

1. Identify the fault.
2. If possible, identify a test case that does not execute the fault.
3. If possible, identify a test case that executes the fault, but does not result in an error state.
4. If possible identify a test case that results in an error, but not a failure.
5. For the given test case, identify the first error state. Be sure to describe the complete state.

```
public int findLast (int[] x, int y)
{
//Effects: If x==null throw NullPointerException
//   else return the index of the last element
//   in x that equals y.
//   If no such element exists, return -1
   for (int i=x.length-1; i > 0; i--)
   {
      if (x[i] == y)
      {
         return i;
      }
   }
   return -1;
}
   // test:  x=[2, 3, 5]; y = 2
   //        Expected = 0
```

- You have 10 minutes
- Do the exercise in groups; put all names.
- Upload your answer in Moodle.



RIPR Model

Reachability, Infection, Propagation, and Revealability

```
public static boolean isLeap(int year) {
	if (year % 4 != 0) return false;
	if (year % 400 == 0) return true;
	if (year % 100 == 0) return false;

	return true;
}



//tests(expected)
//2000(true)
//2001(false)
//2004(true)
//2100(false)
```

```
public static boolean isLeap(int year) {
	if (year % 4 != 0) return false;
	if (year % 400 == 0) return true;
	if (year % 100 < 0) return false;

	return true;
}




//tests(expected)
//2000(true)
//2001(false)
//2004(true)
//2100(false)
```

```
public static boolean isLeap(int year) {
      if (year % 4 != 0) return false;
      if (year % 400 == 0) return true;
      if (year % 100 < 0) return false;

      return true;
}
```

```
//tests(expected)
//2000(true)
//2001(false)
//2004(true)
//2100(false)
```

Reachability	Infection	Propagation	Revealability