Software Testing Project

CA267

Donnacha Maguire
Graham Bartley
Fionn Ó Maoileoin

The Problem

Premise:

- “A man who makes wooden ladders isn't too precise with his measurements and the steps on his ladders are not necessarily equidistant.”
- “The safety officer is concerned about these irregular steps...”

Requirements:

- The problem requires checking distances between rungs on a ladder.
- The objective is to determine the largest distance.

The Solution

- We exchanged code with another group
  - We thought it would be more beneficial to test using someone else's code, especially in relation to black box testing.
  - Good practice in using other people's code and in the case of white box testing, analyzing it.
- We kept a blog
  - Blog entries were written periodically, typically after our meetings which detailed our progress, meeting agendas and plans for the future of the project.
  - The blog entries can be found here https://loop.dcu.ie/blog/index.php?userid=120301

Types of Testing

- Black Box Testing
  - Boundary Value Testing (BVT)
    - Robust
    - Worst Case
    - Special Value
  - Equivalence Class Testing (ECT)
    - Weak
    - Strong
    - Traditional
- White Box Testing
  - Decision Tables (DT)
    - D-D Path

Boundary Value Testing

Single fault assumption

Here i return the minimum value but the program does not deal with minus numbers making 0 the minimum

```
C:\Users\Donnacha>java BlackBox
How many steps:
3
Enter the locations:
-2147483648 10 20
Enter a number greater than 0
```

I use 3 nominal values here in order to show that the program works normally

```
C:\Users\Donnacha>java BlackBox
How many steps:
3
Enter the locations:
10 20 30
The largest gap is 10
```

Boundary Value Testing

More Single fault assumption…

The program still works at the maximum value but as soon as you go one value over the maximum it causes the code to break

```
How many steps:
3
Enter the locations:
1 2 2147483647
The largest gap is 2147483645
```

```
How many steps:
3
Enter the locations:
1 2 2147483648
Exception in thread "main" java.util.InputMismatchException: For input string: "
2147483648"
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at BlackBox.readArray(BlackBox.java:32)
        at BlackBox.main(BlackBox.java:12)
```

Boundary Value Testing

Robust Testing

For this test case I exceeded the maximum by a huge amount guaranteeing it would fail, I was not able to exceed the minimum as the code won't accept any numbers below 0

```
How many steps:
3
Enter the locations:
1 2 2147500000
Exception in thread "main" java.util.InputMismatchException: For input string: "
2147500000"
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at BlackBox.readArray(BlackBox.java:32)
        at BlackBox.main(BlackBox.java:12)
```

Boundary Value Testing

Worst Case Testing

I test what happens when two of the values which give errors are entered, a value above the maximum and a value below the minimum, the code won't accept the minus figure and an error is shown for the maximum.

```
C:\Users\Donnacha>java BlackBox
How many steps:
2
Enter the locations:
-2147483647 2147483648
Enter a number greater than 0
Exception in thread "main" java.util.InputMismatchException: For input string: "
2147483648"
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at BlackBox.readArray(BlackBox.java:27)
        at BlackBox.main(BlackBox.java:12)
```

Boundary Value Testing

Special Value Testing

The special value I entered to try and break the code was 0, I entered it for the amount of steps and the location, printing that the largest gap was 0, showing the code could handle the special value

```
C:\Users\Donnacha>java BlackBox
How many steps:
1
Enter the locations:
0
The largest gap is 0

C:\Users\Donnacha>java BlackBox
How many steps:
0
Enter the locations:
The largest gap is 0

C:\Users\Donnacha>
```

Equivalence Class Testing

For the sake of carrying out ECT, I assume the number of steps to be 2 since the program can theoretically have infinite inputs. This gives 3 inputs (1 for number of steps and 2 for step distances) and 9 equivalence classes (3 for each input). I assume this based on the fact that any other input for step distance past the second one would be equivalent to testing the first two

Equivalence Class Partitioning

These input domains can be partitioned into equivalence classes as follows:

$\cancel{X_1 = X_1 1, X_1 2, X_1}$

[1] $X_1 = X_1 1 \cup X_1 2 \cup X_1 3$

where

$-\infty < X_1 1 < -2{,}147{,}483{,}648$

$-2{,}147{,}483{,}648 \leq X_1 2 \leq 2{,}147{,}483{,}647$

$2{,}147{,}483{,}647 < X_1 3 < \infty$

[2] $X_2 = X_2 1 \cup X_2 2 \cup X_2 3$

where

$-\infty < X_2 1 < 1$

$1 \leq X_2 2 \leq 2{,}147{,}483{,}647$

$2{,}147{,}483{,}647 < X_2 3 < \infty$

[3] $X_3 = X_3 1 \cup X_3 2 \cup X_3 3$

where

$-\infty < X_3 1 < x_2 + 1$

$x_2 + 1 \leq X_3 2 \leq 2{,}147{,}483{,}647$

$2{,}147{,}483{,}647 < X_3 3 < \infty$

[n] $X_n = X_n 1 \cup X_n 2 \cup X_n 3$

$-\infty < X_n 1 < x_{n-1} + 1$

$x_{n-1} + 1 \leq X_n 2 \leq 2{,}147{,}483{,}647$

$2{,}147{,}483{,}647 < X_n 3 < \infty$

Weak Equivalence Class Testing

- Take one representative value from each equivalence class in a test case
- Generated 3 test cases for this problem
- Program handled 100% of valid input in test case #2
- Program handled 0% of invalid inputs, crashed for 100%

```
C:\Users\bartleg3.CA\Desktop\ECT>java BlackBox
How many steps:
-3000000000 0 -1
Exception in thread "main" java.util.InputMismatchException: For input string: "-3000000000"
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at BlackBox.getAmountOfSteps(BlackBox.java:77)
        at BlackBox.main(BlackBox.java:9)

C:\Users\bartleg3.CA\Desktop\ECT>java BlackBox
How many steps:
2 3 5
Enter the locations:
The largest gap is 2

C:\Users\bartleg3.CA\Desktop\ECT>java BlackBox
How many steps:
3000000000 3000000000 3000000000
Exception in thread "main" java.util.InputMismatchException: For input string: "3000000000"
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at BlackBox.getAmountOfSteps(BlackBox.java:77)
        at BlackBox.main(BlackBox.java:9)

C:\Users\bartleg3.CA\Desktop\ECT>
```

Strong Equivalence Class Testing

- Generates test cases based on the cartesian product of the equivalence classes
- Generated 27 test cases for this problem (3 x 3 x 3)
- Representative values are again used for each equivalence class
- Program handled 100% of valid input
- Program handled 11.5% of invalid input by prompting for valid input, it crashed for the other 88.5%

Test Case	x_1	x_2	x_3
SE1	-3×10^9	0	-1
SE2	-3×10^9	0	5
SE3	-3×10^9	0	3×10^9
SE4	-3×10^9	3	-1
SE5	-3×10^9	3	5
SE6	-3×10^9	3	3×10^9
SE7	-3×10^9	3×10^9	-1
SE8	-3×10^9	3×10^9	5
SE9	-3×10^9	3×10^9	3×10^9
SE10	2	0	-1
SE11	2	0	5
SE12	2	0	3×10^9
SE13	2	3	-1
SE14	2	3	5
SE15	2	3	3×10^9
SE16	2	3×10^9	-1
SE17	2	3×10^9	5
SE18	2	3×10^9	3×10^9
SE19	3×10^9	0	-1
SE20	3×10^9	0	5
SE21	3×10^9	0	3×10^9
SE22	3×10^9	3	-1
SE23	3×10^9	3	5
SE24	3×10^9	3	3×10^9
SE25	3×10^9	3×10^9	-1
SE26	3×10^9	3×10^9	5
SE27	3×10^9	3×10^9	3×10^9

Strong Equivalence Class Testing

- This screenshot shows 6 out of the 27 strong test cases with an example of valid, handled invalid and unhandled invalid inputs

```
C:\Users\bartleg3.CA\Desktop\ECT>java BlackBox
How many steps:
2 3 -1
Enter the locations:
Enter a number greater than 0
AZ
Exception in thread "main" java.util.NoSuchElementException
        at java.util.Scanner.throwFor(Unknown Source)
        at java.util.Scanner.next(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at BlackBox.readInLadder(BlackBox.java:40)
        at BlackBox.main(BlackBox.java:12)

C:\Users\bartleg3.CA\Desktop\ECT>java BlackBox
How many steps:
2 3 5
Enter the locations:
The largest gap is 2

C:\Users\bartleg3.CA\Desktop\ECT>java BlackBox
How many steps:
2 3 3000000000
Enter the locations:
Exception in thread "main" java.util.InputMismatchException: For input string: "3000000000"
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at BlackBox.readInLadder(BlackBox.java:34)
        at BlackBox.main(BlackBox.java:12)

C:\Users\bartleg3.CA\Desktop\ECT>java BlackBox
How many steps:
2 3000000000 -1
Enter the locations:
Exception in thread "main" java.util.InputMismatchException: For input string: "3000000000"
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at BlackBox.readInLadder(BlackBox.java:24)
        at BlackBox.main(BlackBox.java:12)

C:\Users\bartleg3.CA\Desktop\ECT>java BlackBox
How many steps:
2 3000000000 5
Enter the locations:
Exception in thread "main" java.util.InputMismatchException: For input string: "3000000000"
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at BlackBox.readInLadder(BlackBox.java:24)
        at BlackBox.main(BlackBox.java:12)

C:\Users\bartleg3.CA\Desktop\ECT>java BlackBox
How many steps:
2 3000000000 3000000000
Enter the locations:
Exception in thread "main" java.util.InputMismatchException: For input string: "3000000000"
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at BlackBox.readInLadder(BlackBox.java:24)
        at BlackBox.main(BlackBox.java:12)
```

Traditional Equivalence Class Testing

- Based on valid and invalid inputs
- One test contains all valid inputs
- Invalid tests work on single fault assumption
- All inputs must be used at least once
- Generated 7 test cases for this problem
- Program handled 100% of valid input as expected
- Program handled 33.3% of invalid inputs by prompting for further input and crashed for the other 66.6%, as before

Test Case	x_1	x_2	x_3
TE1	2	3	5
TE2	-3×10^9	3	5
TE3	3×10^9	3	5
TE4	2	0	5
TE5	2	3×10^9	5
TE6	2	3	-1
TE7	2	3	3×10^9

Traditional Equivalence Class Testing

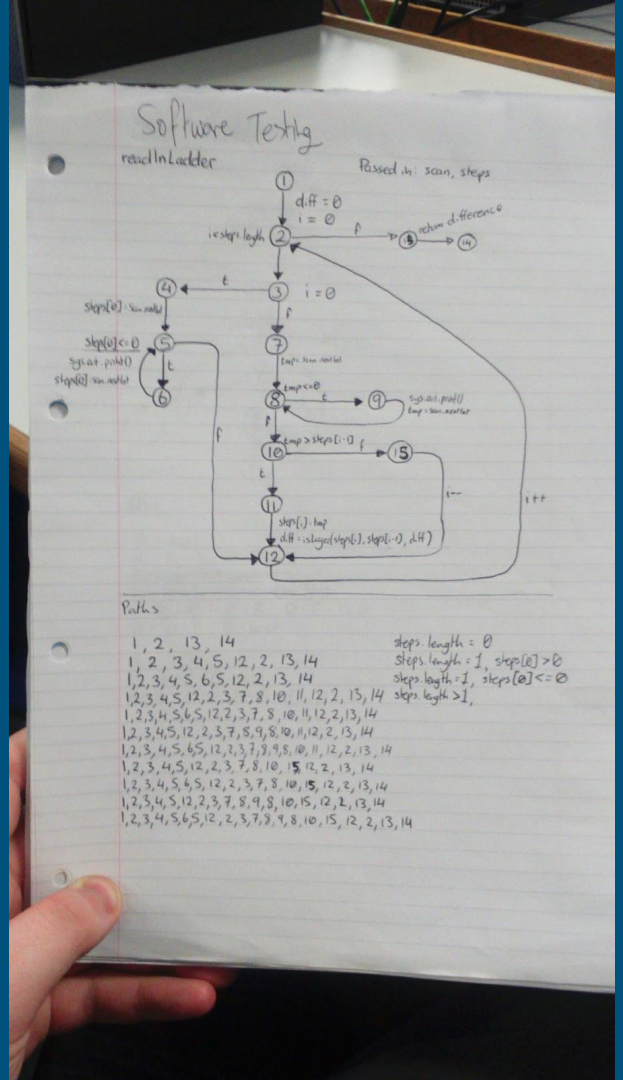
- This screenshot shows 6 out of the 7 traditional test cases (the last one crashed the program) with an example of valid, handled invalid and unhandled invalid inputs

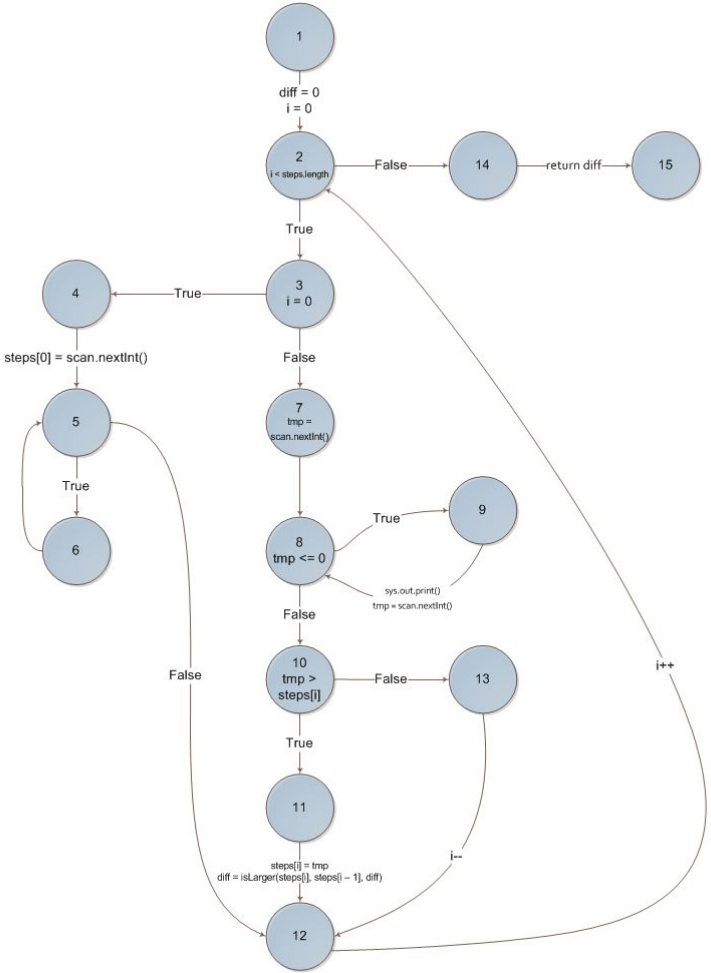
```
C:\Users\bartleg3.CA\Desktop\ECT>java BlackBox
How many steps:
2 3 5
Enter the locations:
The largest gap is 2

C:\Users\bartleg3.CA\Desktop\ECT>java BlackBox
How many steps:
-3000000000 3 5
Exception in thread "main" java.util.InputMismatchException: For input string: "-3000000000"
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at BlackBox.getAmountOfSteps(BlackBox.java:77)
        at BlackBox.main(BlackBox.java:9)

C:\Users\bartleg3.CA\Desktop\ECT>java BlackBox
How many steps:
3000000000 3 5
Exception in thread "main" java.util.InputMismatchException: For input string: "3000000000"
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at BlackBox.getAmountOfSteps(BlackBox.java:77)
        at BlackBox.main(BlackBox.java:9)

C:\Users\bartleg3.CA\Desktop\ECT>java BlackBox
How many steps:
2 0 5
Enter the locations:
Enter a number greater than 0

^Z
Exception in thread "main" java.util.NoSuchElementException
        at java.util.Scanner.throwFor(Unknown Source)
        at java.util.Scanner.next(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at BlackBox.readIntLadder(BlackBox.java:34)
        at BlackBox.main(BlackBox.java:12)

C:\Users\bartleg3.CA\Desktop\ECT>java BlackBox
How many steps:
2 3000000000 5
Enter the locations:
Exception in thread "main" java.util.InputMismatchException: For input string: "3000000000"
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at BlackBox.readIntLadder(BlackBox.java:24)
        at BlackBox.main(BlackBox.java:12)

C:\Users\bartleg3.CA\Desktop\ECT>java BlackBox
How many steps:
2 3 -1
Enter the locations:
Enter a number greater than 0
^Z
Exception in thread "main" java.util.NoSuchElementException
        at java.util.Scanner.throwFor(Unknown Source)
        at java.util.Scanner.next(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at BlackBox.readIntLadder(BlackBox.java:40)
        at BlackBox.main(BlackBox.java:12)
```

Decision Tables

- Broke the program into blocks
- Decided to focus on readInLadder method
  - Other decision trees trivial

```
static int readInLadder(Scanner scan, int [] steps)
{
      int difference = 0;
      for(int i = 0; i < steps.length; i++)      1
      {
            if(i == 0) 2
            {
                  steps[0] = scan.nextInt();      3
                  while (steps[0] <= 0)
                  {
                        System.out.println("Enter a number greater
than 0");                                         4
                        steps[0] = scan.nextInt();
                  }
            }
            else
            {
                  int tmp = scan.nextInt();       5
                  while (tmp <= 0)
                  {
```

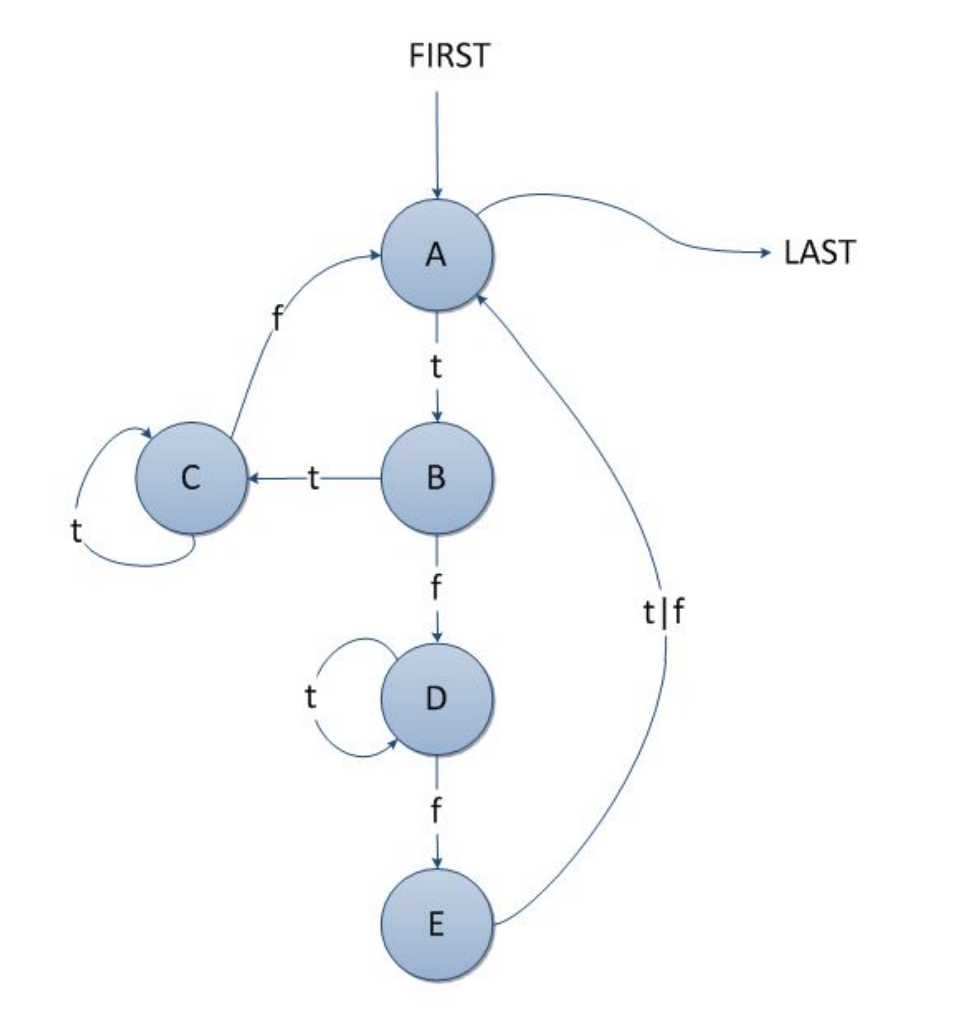

Software Testing
reactInLadder
Passed in: scan, steps
diff = 0
i = 0
return difference
i = 0
steps[0] = scan.nextInt
steps[0] <= 0
sys.out.print()
steps[0] = scan.nextInt
tmp = scan.nextInt
tmp <= 0
sys.out.print()
tmp = scan.nextInt
tmp > steps[i-1]
i--
i++
steps[i] = tmp
diff = isLarger(steps[i], steps[i-1], diff)
Paths
1, 2, 13, 14
steps.length = 0
1, 2, 3, 4, 5, 12, 2, 13, 14
steps.length = 1, steps[0] > 0
1,2,3,4,5,6,5,12,2,13,14
steps.length = 1, steps[0] <= 0
1,2,3,4,5,12,2,3,7,8,10,11,12,2,13,14
steps.length > 1,
1,2,3,4,5,6,5,12,2,3,7,8,10,11,12,2,13,14
1,2,3,4,5,12,2,3,7,8,9,8,10,11,12,2,13,14
1,2,3,4,5,6,5,12,2,3,7,8,9,8,10,11,12,2,13,14
1,2,3,4,5,12,2,3,7,8,10,15,12,2,13,14
1,2,3,4,5,6,5,12,2,3,7,8,10,15,12,2,13,14
1,2,3,4,5,12,2,3,7,8,9,8,10,15,12,2,13,14
1,2,3,4,5,6,5,12,2,3,7,8,9,8,10,15,12,2,13,14


1
diff = 0
i = 0
2
i < steps.length
False
14
return diff
15
True
3
i = 0
True
4
steps[0] = scan.nextInt()
5
True
6
False
7
tmp =
scan.nextInt()
8
tmp <= 0
True
9
sys.out.print()
tmp = scan.nextInt()
False
10
tmp >
steps[i]
False
13
True
11
steps[i] = tmp
diff = isLarger(steps[i], steps[i – 1], diff)
i--
12
False
i++

DD Paths

- Further simplified the flow control graph into a DD-Path diagram

A: i < steps.length()
B: i == 0
C: steps[0] <= 0
D: tmp <= 0
E: tmp > steps[i-1]



DD Paths

- Found all routes through DD Path Diagram
  - 11 test cases found
- Loops are underlined
- Not comprehensive
  - Efficient/effective

Af - last
At - Bt - Cf - Af - last
At - Bt - Ct - Af - last
At - Bt - Cf - At - Bf - Df - Ef - Af - last
At - Bt - Cf - At - Bf - Df - Et - Af - last
At - Bt - Cf - At - Bf - Dt - Ef - Af - last
At - Bt - Cf - At - Bf - Dt - Et - Af - last
At - Bt - Ct - At - Bf - Df - Ef - Af - last
At - Bt - Ct - At - Bf - Df - Et - Af - last
At - Bt - Ct - At - Bf - Dt - Ef - Af - last
At - Bt - Ct - At - Bf - Dt - Et - Af - last

DD Paths

All inputs handled well

- Only when given the right array size

Every part of the program works as it should within DD Path bounds

```
//Empty array
int [] ladderNums1 = new int [0];
System.out.println("Test Case 1:" + readInLadder(scan, ladderNums1));
```

Command Prompt

```
C:\Users\Fionn\Documents\Coding Shit>java ReadLadderTest
Test Case 1:0

C:\Users\Fionn\Documents\Coding Shit>_
```

```
//Array has one (valid) element
int [] ladderNums1 = new int [1];
System.out.println("Test Case 2:" + readInLadder(scan, ladderNums1));
```

Command Prompt

```
C:\Users\Fionn\Documents\Coding Shit>java ReadLadderTest
10
Test Case 2:0

C:\Users\Fionn\Documents\Coding Shit>_
```

```
//First element is <=0, there is at least one element that is <= its predecessor
int [] ladderNums1 = new int [6];
System.out.println("Test Case 11:" + readInLadder(scan, ladderNums1));
```

Command Prompt

```
C:\Users\Fionn\Documents\Coding Shit>java ReadLadderTest
0 5 10 20 0 40 50 60
Enter a number greater than 0
Enter a number greater than 0
Test Case 11:20
```

What We Learned

- Apply different methods of testing in a practical way
- Analyze other people’s code and target specific weak points when testing
- In Depth look at the types of testing we each did individually

Any Questions?

Donnacha Maguire
Graham Bartley
Fionn Ó Maoileoin

BlackBox.java

```java
import java.util.Scanner;

public class BlackBox
{
    public static void main(String [] args)
    {
        Scanner scan = new Scanner(System.in);
        System.out.println("How many steps: ");
        int amountOfSteps = getAmountOfSteps(scan);
        int [] steps = new int [amountOfSteps];
        System.out.println("Enter the locations: ");
        System.out.println("The largest gap is " + readInLadder(scan,steps));
    }

    static int readInLadder(Scanner scan, int [] steps)
    {
        int difference = 0;

        for(int i = 0; i < steps.length; i++)
        {
            //Ensure first element is not compared against anything before it
            if(i == 0)
            {
                steps[0] = scan.nextInt();
                while (steps[0] <= 0)
                {
                    System.out.println("Enter a number greater than 0");
                    steps[0] = scan.nextInt();
                }
            }
            //All other elements need to be compared against the previous element
            else
            {
                int tmp = scan.nextInt();

                //Is number less than 0?
                while (tmp <= 0)
                {
                    System.out.println("Enter a number greater than 0");
                    tmp = scan.nextInt();
                }
                if(tmp > steps[i-1])
                {
                    steps[i] = tmp;
                    difference = isLarger(steps[i],steps[i-1], difference);
                }

                else
                {
                    System.out.println("Step can't be less than previous number!");
                    //Set i back to allow a correct input
                    i--;
                }
            }
```

Java source file | length : 1847 lines : 87 | Ln : 1 Col : 1 Sel : 0 | 0 | Dos\Windows | ANSI | INS