Høyskolen Kristiania

ITP2200
Introduction to Software Testing

Bogdan Marculescu

Høyskolen Kristiania

Lecture 3
Graph Coverage
Logic Coverage

So, more programming this week?

Høyskolen Kristiania

What do we remember from last time?

Høyskolen Kristiania

We mentioned “Coverage”

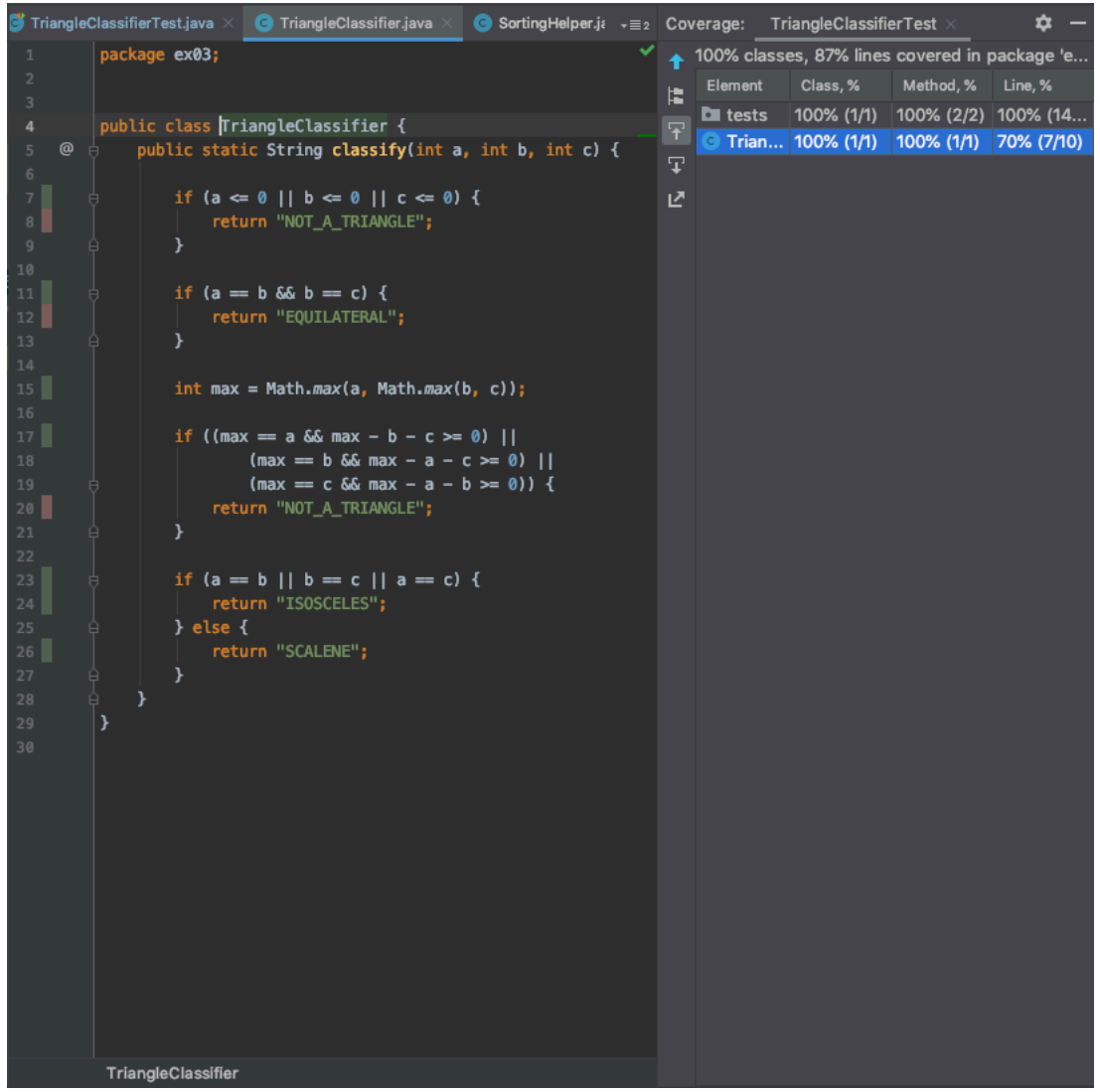
```java
public class TriangleClassifier {
    public static String classify(int a, int b, int c) {

        if (a <= 0 || b <= 0 || c <= 0) {
            return "NOT_A_TRIANGLE";
        }

        if (a == b && b == c) {
            return "EQUILATERAL";
        }

        int max = Math.max(a, Math.max(b, c));

        if ((max == a && max - b - c >= 0) ||
                (max == b && max - a - c >= 0) ||
                (max == c && max - a - b >= 0)) {
            return "NOT_A_TRIANGLE";
        }

        if (a == b || b == c || a == c) {
            return "ISOSCELES";
        } else {
            return "SCALENE";
        }
    }
}
```

```java
@Test
public void testScalene() throws Exception {
    int a = 5;
    int b = 7;
    int c = 9;

    String result = TriangleClassifier.classify(a, b, c);
    assertTrue(result.equalsIgnoreCase( anotherString: "SCALENE"));

}

@Test
public void testIsosceles() throws Exception{
    int a = 5;
    int b = 5;
    int c = 7;

    String result = TriangleClassifier.classify(a, b, c);
    assertTrue(result.equalsIgnoreCase( anotherString: "ISOSCELES"));
    assertFalse(result.equalsIgnoreCase( anotherString: "SCALENE"));
}
```

Code is at
https://github.com/bogdanmarculescu/itp2200

Høyskolen Kristiania

We mentioned “Coverage”



Interpreting Coverage:

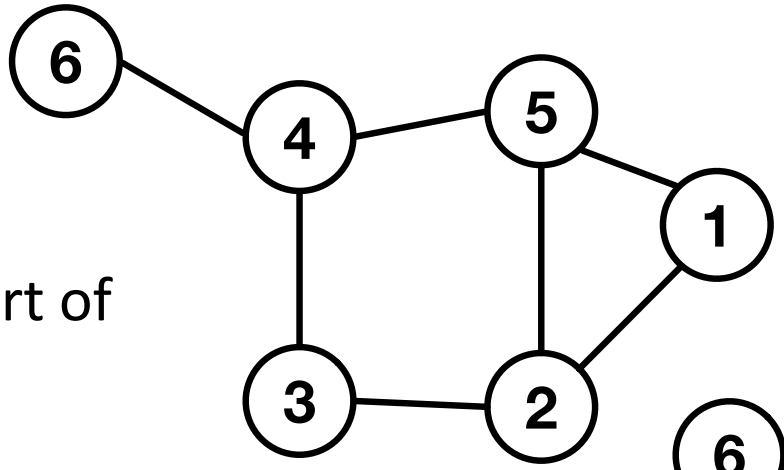
What happens to code that is not executed?

How should we change our test to cover those lines as well?

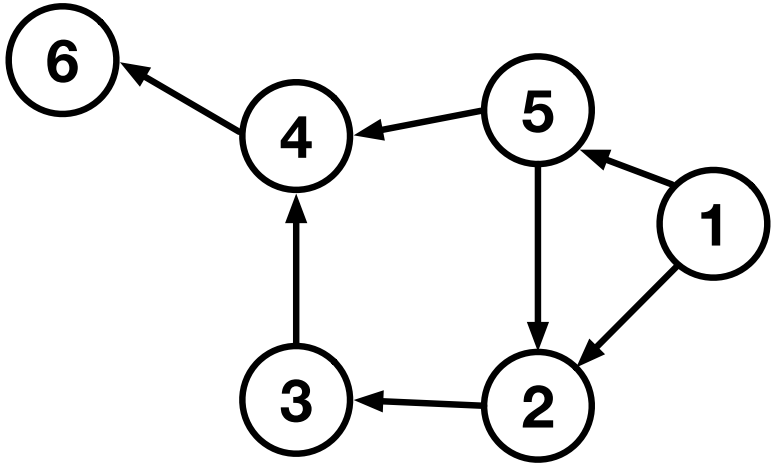
Høyskolen Kristiania

A (very brief) overview of graphs

A graph:
Structure
Nodes (Vertices, points)
- Objects that have some sort of connection



Edges
- Connection between them (can be directional or not)



Path
- A way of “moving” through the graph

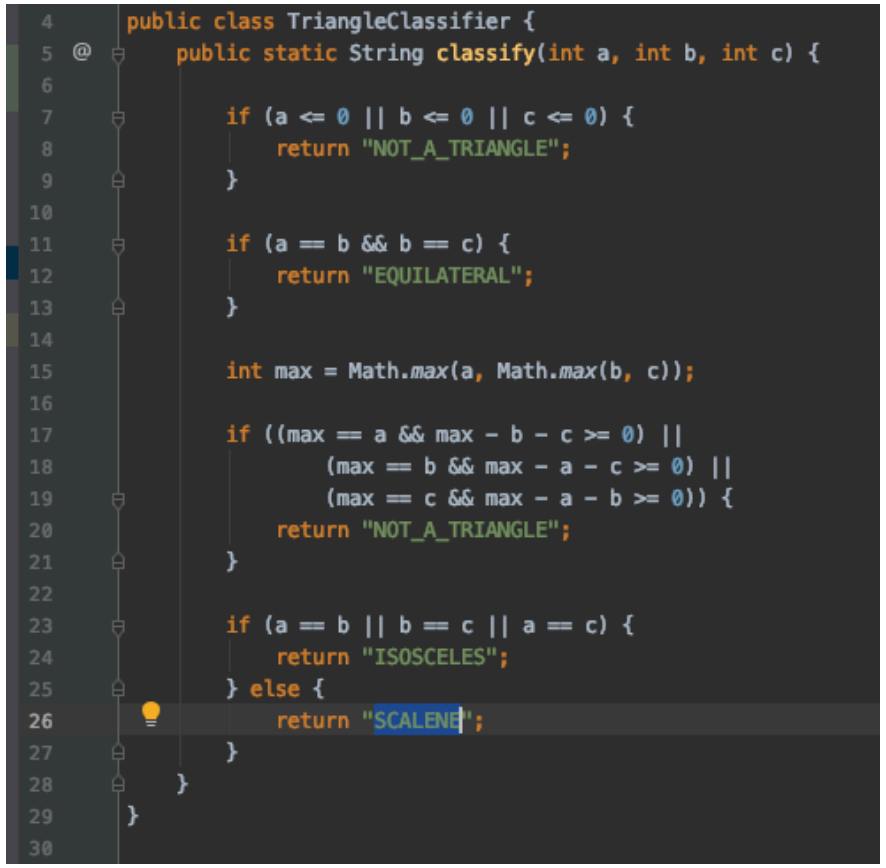
Høyskolen Kristiania

Code as a Graph

Abstraction of code:

Nodes – lines of code

Edges - transitions between lines (i.e. what gets executed)



```
4   public class TriangleClassifier {
5   @   public static String classify(int a, int b, int c) {
6
7           if (a <= 0 || b <= 0 || c <= 0) {
8               return "NOT_A_TRIANGLE";
9           }
10
11          if (a == b && b == c) {
12              return "EQUILATERAL";
13          }
14
15          int max = Math.max(a, Math.max(b, c));
16
17          if ((max == a && max - b - c >= 0) ||
18                  (max == b && max - a - c >= 0) ||
19                  (max == c && max - a - b >= 0)) {
20              return "NOT_A_TRIANGLE";
21          }
22
23          if (a == b || b == c || a == c) {
24              return "ISOSCELES";
25          } else {
26              return "SCALENE";
27          }
28      }
29  }
30
```

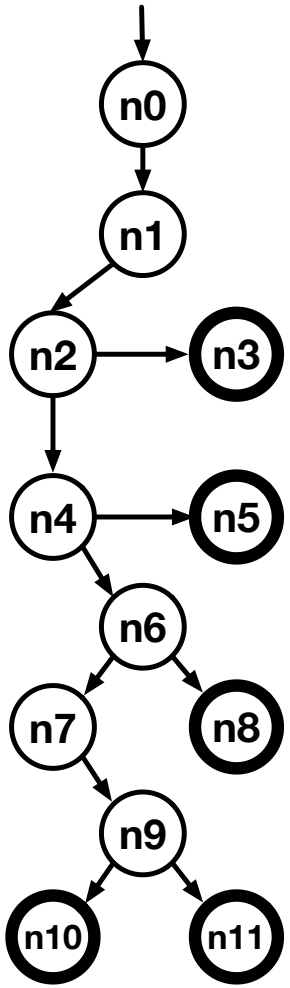
Høyskolen Kristiania

Code as a Graph

Abstraction of code:

Nodes – lines of code

Edges - transitions between lines (i.e. what gets executed)




```
4   public class TriangleClassifier {
5   @   public static String classify(int a, int b, int c) {
6
7           if (a <= 0 || b <= 0 || c <= 0) {
8               return "NOT_A_TRIANGLE";
9           }
10
11          if (a == b && b == c) {
12              return "EQUILATERAL";
13          }
14
15          int max = Math.max(a, Math.max(b, c));
16
17          if ((max == a && max - b - c >= 0) ||
18                  (max == b && max - a - c >= 0) ||
19                  (max == c && max - a - b >= 0)) {
20              return "NOT_A_TRIANGLE";
21          }
22
23          if (a == b || b == c || a == c) {
24              return "ISOSCELES";
25          } else {
26              return "SCALENE";
27          }
28      }
29  }
30
```

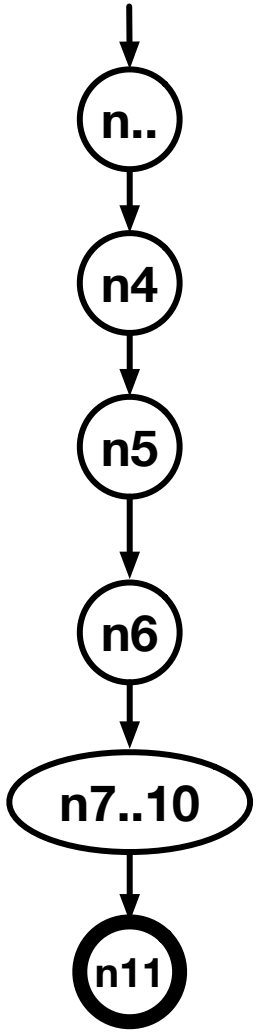
Høyskolen Kristiania

Code as a Graph

Abstraction of code:

Nodes – lines of code

Edges - transitions between lines (i.e. what gets executed)




```java
public class SortingHelper {

    public static int[] sortArray(int[] array){
        int n = array.length;
        int[] result = array.clone();
        int temp = 0;
        for(int i=0; i < n; i++) // Looping through the array length
        {
            for(int j=1; j < (n-i); j++)
            {
                if(result[j-1] > result[j])
                {
                    //swap elements
                    temp = result[j-1];
                    result[j-1] = result[j];
                    result[j] = temp;
                }

            }
        }

        return result;
    }
```

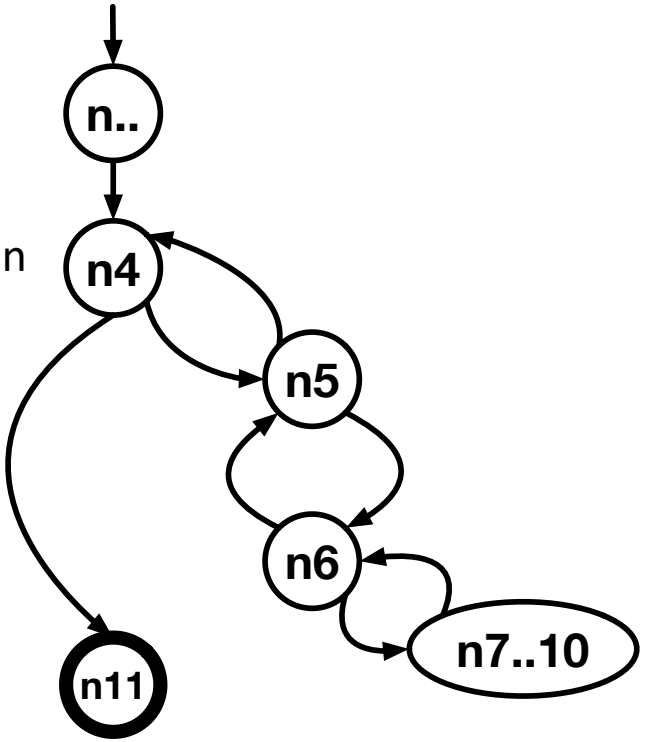
Høyskolen Kristiania

Code as a Graph

Abstraction of code:

Nodes – lines of code

Edges - transitions between lines (i.e. what gets executed)




```java
public class SortingHelper {

    public static int[] sortArray(int[] array){
        int n = array.length;
        int[] result = array.clone();
        int temp = 0;
        for(int i=0; i < n; i++) // Looping through the array length
        {
            for(int j=1; j < (n-i); j++)
            {
                if(result[j-1] > result[j])
                {
                    //swap elements
                    temp = result[j-1];
                    result[j-1] = result[j];
                    result[j] = temp;
                }

            }
        }

        return result;
    }
```

Høyskolen Kristiania

Back to our code

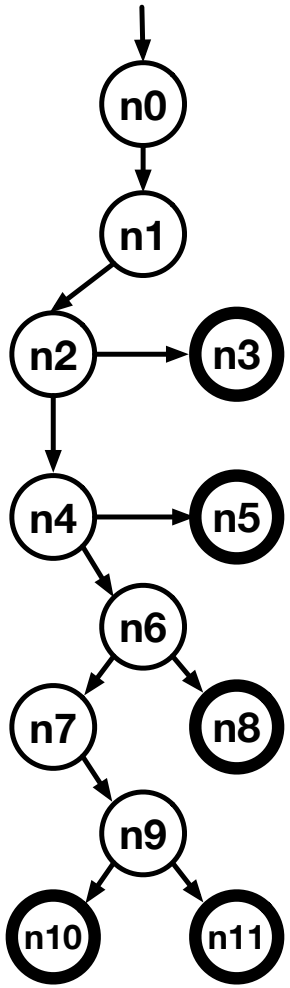
```java
public class TriangleClassifier {
    public static String classify(int a, int b, int c) {

        if (a <= 0 || b <= 0 || c <= 0) {
            return "NOT_A_TRIANGLE";
        }

        if (a == b && b == c) {
            return "EQUILATERAL";
        }

        int max = Math.max(a, Math.max(b, c));

        if ((max == a && max - b - c >= 0) ||
                (max == b && max - a - c >= 0) ||
                (max == c && max - a - b >= 0)) {
            return "NOT_A_TRIANGLE";
        }

        if (a == b || b == c || a == c) {
            return "ISOSCELES";
        } else {
            return "SCALENE";
        }
    }
}
```

```java
@Test
public void testScalene() throws Exception {
    int a = 5;
    int b = 7;
    int c = 9;

    String result = TriangleClassifier.classify(a, b, c);
    assertTrue(result.equalsIgnoreCase( anotherString: "SCALENE"));

}

@Test
public void testIsosceles() throws Exception{
    int a = 5;
    int b = 5;
    int c = 7;

    String result = TriangleClassifier.classify(a, b, c);
    assertTrue(result.equalsIgnoreCase( anotherString: "ISOSCELES"));
    assertFalse(result.equalsIgnoreCase( anotherString: "SCALENE"));
}
```

Høyskolen Kristiania

Code as a Graph

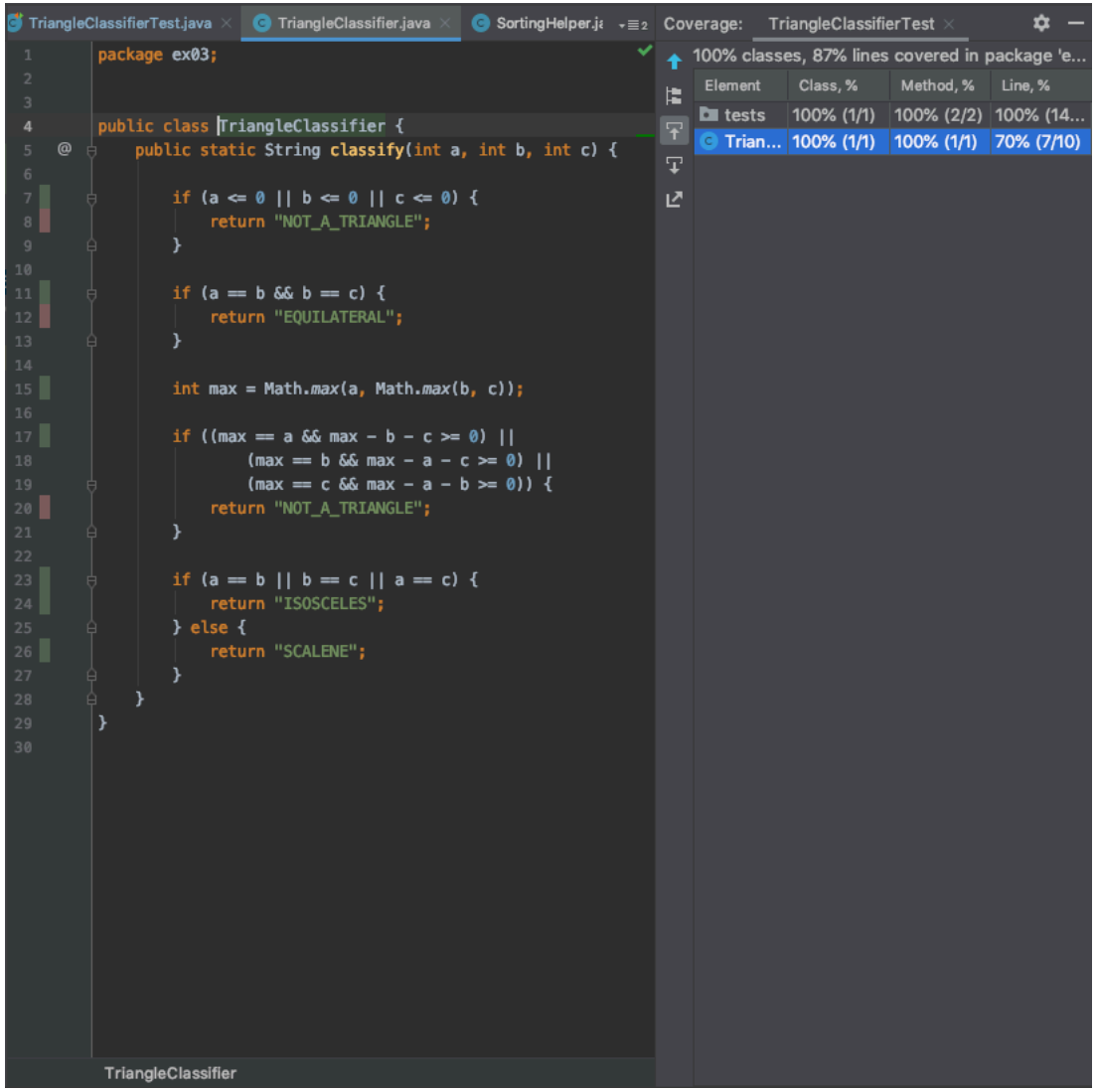
```java
public class TriangleClassifier {
    public static String classify(int a, int b, int c) {

        if (a <= 0 || b <= 0 || c <= 0) {
            return "NOT_A_TRIANGLE";
        }

        if (a == b && b == c) {
            return "EQUILATERAL";
        }

        int max = Math.max(a, Math.max(b, c));

        if ((max == a && max - b - c >= 0) ||
                (max == b && max - a - c >= 0) ||
                (max == c && max - a - b >= 0)) {
            return "NOT_A_TRIANGLE";
        }

        if (a == b || b == c || a == c) {
            return "ISOSCELES";
        } else {
            return "SCALENE";
        }
    }
}
```

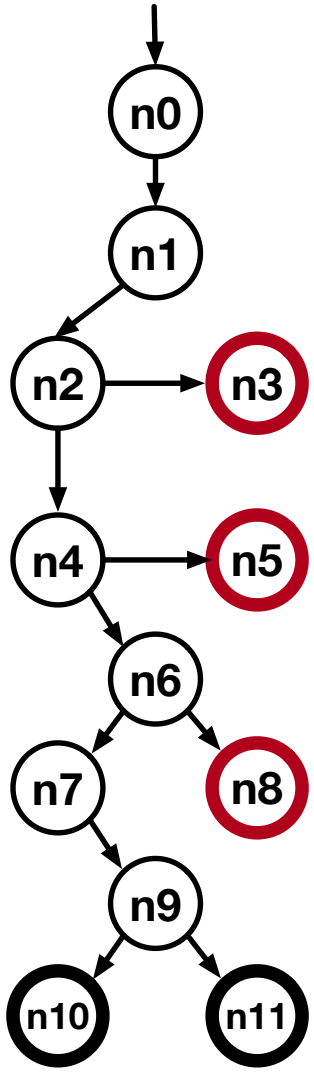



```java
@Test
public void testScalene() throws Exception {
    int a = 5;
    int b = 7;
    int c = 9;

    String result = TriangleClassifier.classify(a, b, c);
    assertTrue(result.equalsIgnoreCase( anotherString: "SCALENE"));

}

@Test
public void testIsosceles() throws Exception{
    int a = 5;
    int b = 5;
    int c = 7;

    String result = TriangleClassifier.classify(a, b, c);
    assertTrue(result.equalsIgnoreCase( anotherString: "ISOSCELES"));
    assertFalse(result.equalsIgnoreCase( anotherString: "SCALENE"));
}
```

Høyskolen Kristiania

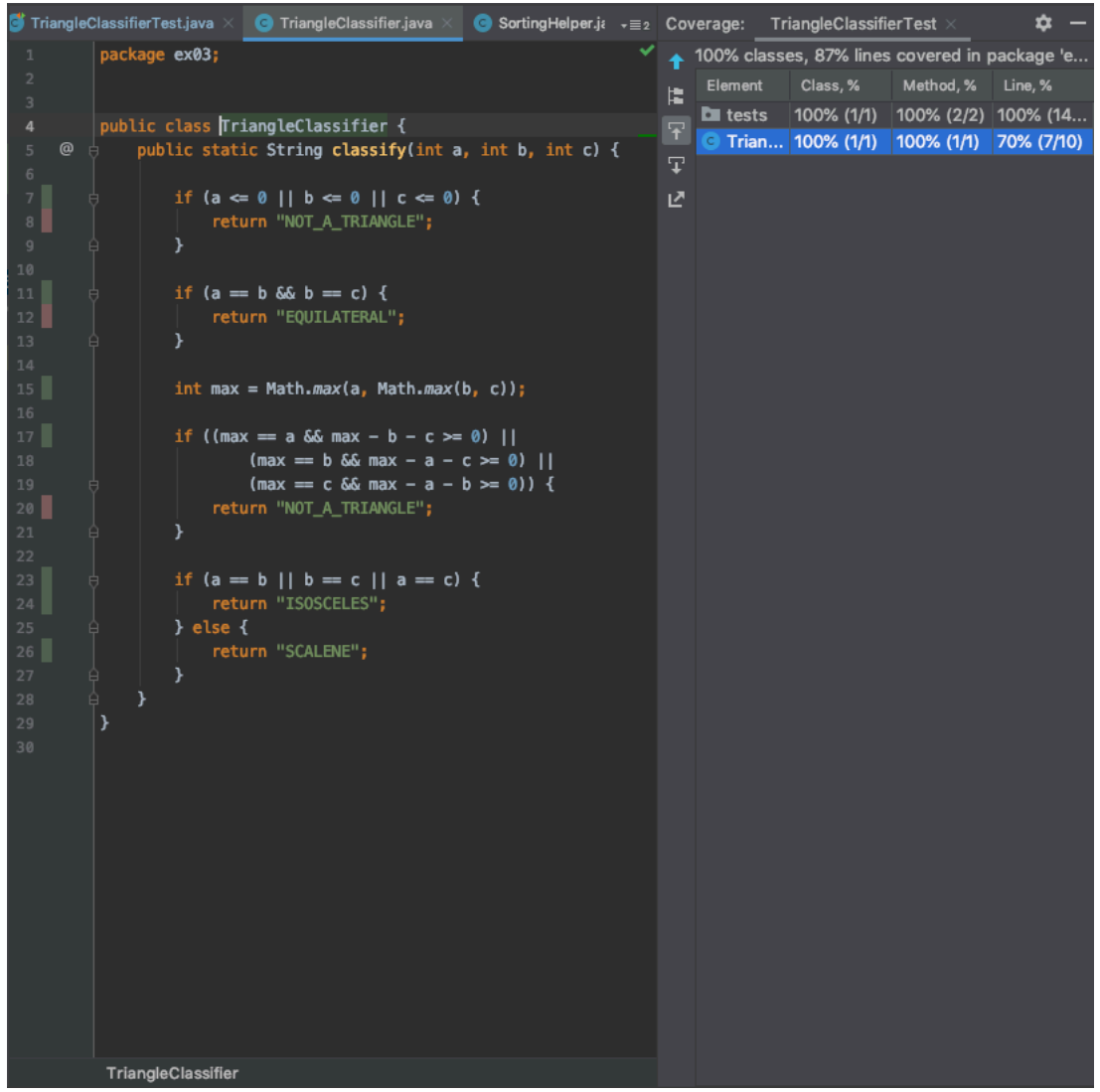
We mentioned “Coverage”



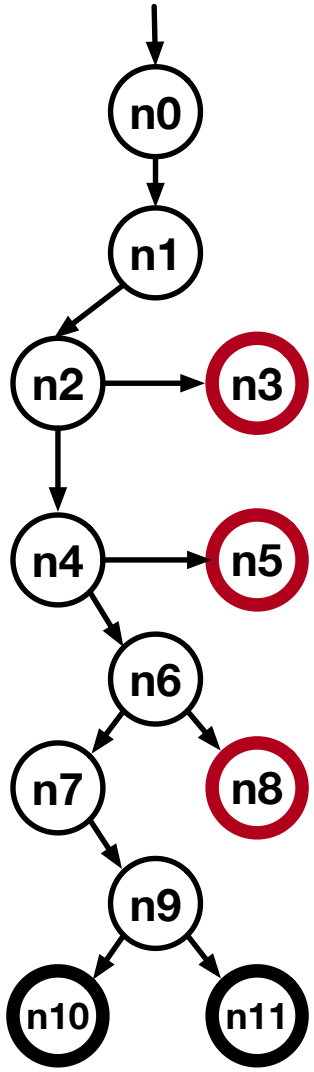


Høyskolen Kristiania

We mentioned “Coverage”




```java
package ex03;

public class TriangleClassifier {
    public static String classify(int a, int b, int c) {

        if (a <= 0 || b <= 0 || c <= 0) {
            return "NOT_A_TRIANGLE";
        }

        if (a == b && b == c) {
            return "EQUILATERAL";
        }

        int max = Math.max(a, Math.max(b, c));

        if ((max == a && max - b - c >= 0) ||
                (max == b && max - a - c >= 0) ||
                (max == c && max - a - b >= 0)) {
            return "NOT_A_TRIANGLE";
        }

        if (a == b || b == c || a == c) {
            return "ISOSCELES";
        } else {
            return "SCALENE";
        }
    }
}
```



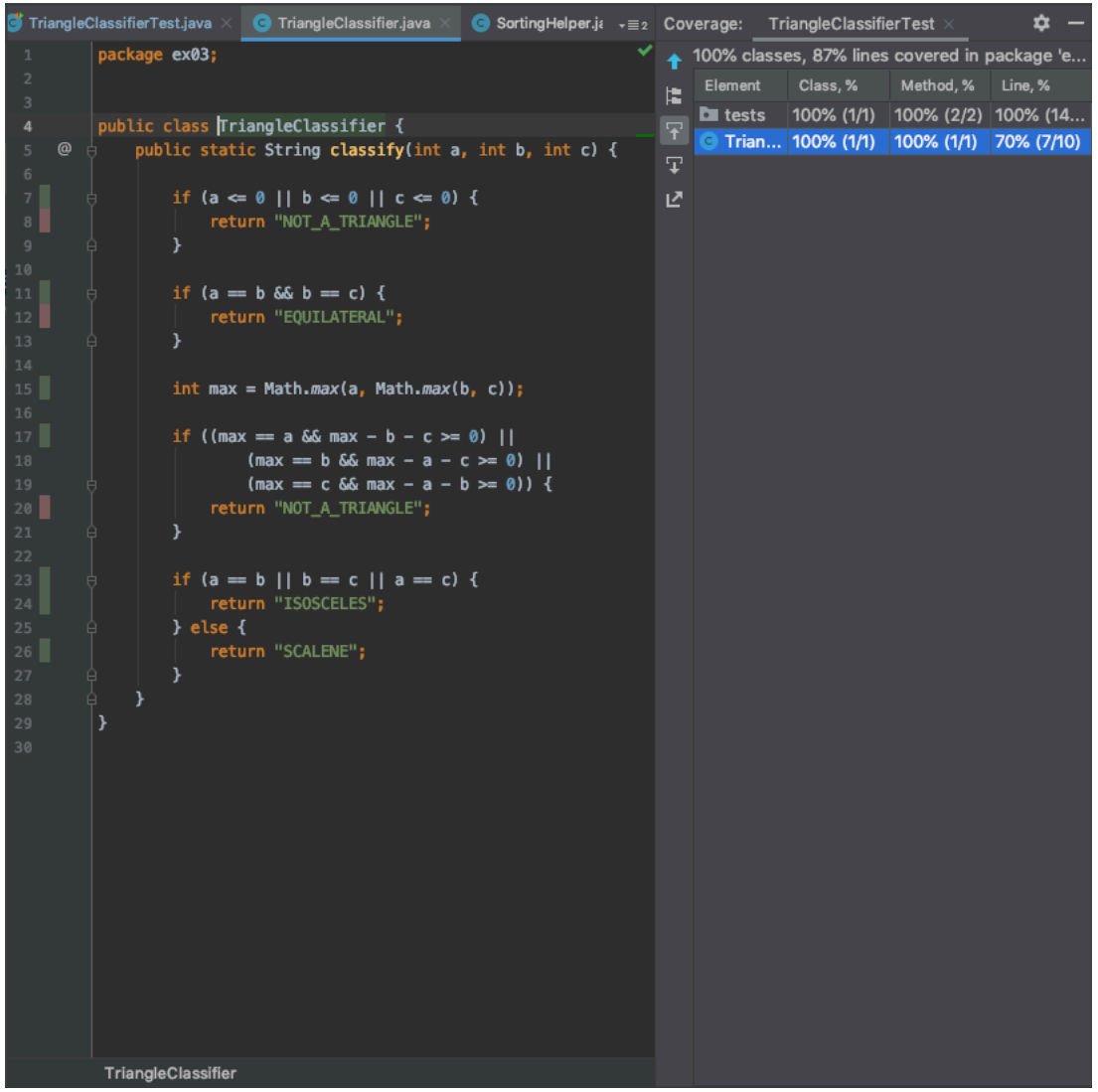
How can we cover the remaining nodes?

Are all nodes reachable?

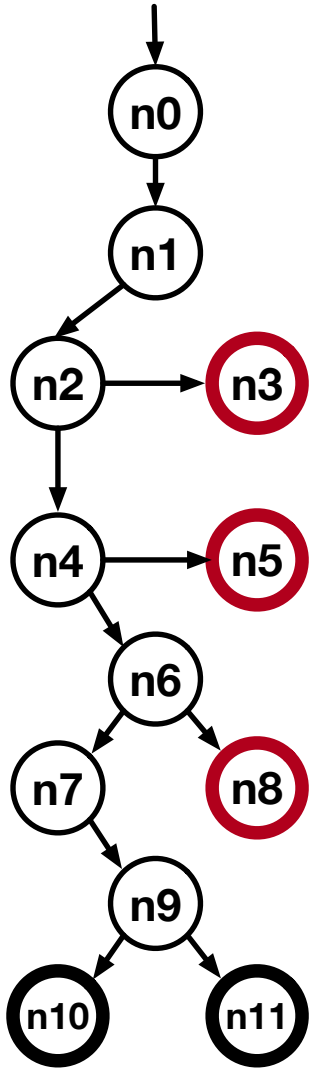
(Should we?
Is it worth doing?)

Høyskolen Kristiania

Code!!!




```java
package ex03;

public class TriangleClassifier {
    public static String classify(int a, int b, int c) {

        if (a <= 0 || b <= 0 || c <= 0) {
            return "NOT_A_TRIANGLE";
        }

        if (a == b && b == c) {
            return "EQUILATERAL";
        }

        int max = Math.max(a, Math.max(b, c));

        if ((max == a && max - b - c >= 0) ||
                (max == b && max - a - c >= 0) ||
                (max == c && max - a - b >= 0)) {
            return "NOT_A_TRIANGLE";
        }

        if (a == b || b == c || a == c) {
            return "ISOSCELES";
        } else {
            return "SCALENE";
        }
    }
}
```



Let’s add tests to cover the remaining nodes!

Høyskolen Kristiania

Høyskolen Kristiania

Questions so far?

Exercise for the Seminar

Rooms:

A3-01

A4-03

A5-10

A5-18

Høyskolen Kristiania

Exercise for the Seminar

Self-organized.
In repo:
Stats.java - Contains code for some basic stats on a group of numbers.

tests/StatsTest.java - Contains scaffolding for tests.

SO:
Draw up the graph for Stats.java
Write up what paths you want to cover (ideally all nodes will be covered)
Write up the tests for those paths.
Run (with coverage)!

Høyskolen Kristiania

Exercise for the Seminar

When done
PatternMatcher.java - Contains code for some string pattern matching.

tests/PatTest.java - Contains scaffolding for tests.

SO:
Draw up the graph for PatternMatcher.java
Write up what paths you want to cover (ideally all nodes will be covered)
Write up the tests for those paths.
Run (with coverage)!

Høyskolen Kristiania