

# ITP2200

# Introduction to Software Testing

**Bogdan Marculescu**

# Lecture 2

## Coverage and Coverage criteria

# So, how did the homework go?



# Quick assignment until next week

## Homework:

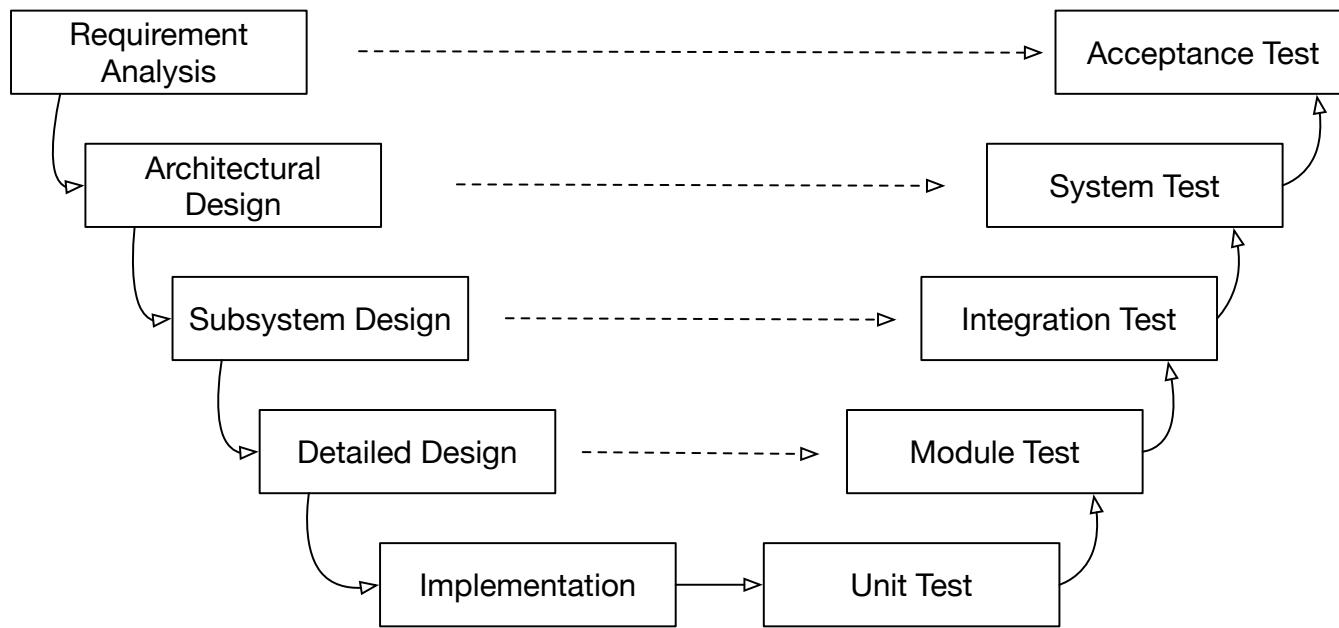
- Have a quick look at your projects from last semester.
- Try to map all the (very many, I know) definitions to those projects
- Write a quick description and let's discuss them
- Prepare questions about how they can be improved in terms of quality



# What do we remember from last time?



# What do we remember from last time?



- Or the one-sentence guide to the entire field of software engineering

# Some simple code

What values we should use?

- Are there any values that would reveal a bug?

A:

```
if ( a >= 0 ) return "a is positive";  
else return " A BUUUUG! ";
```



# Some simple code

What values we should use?

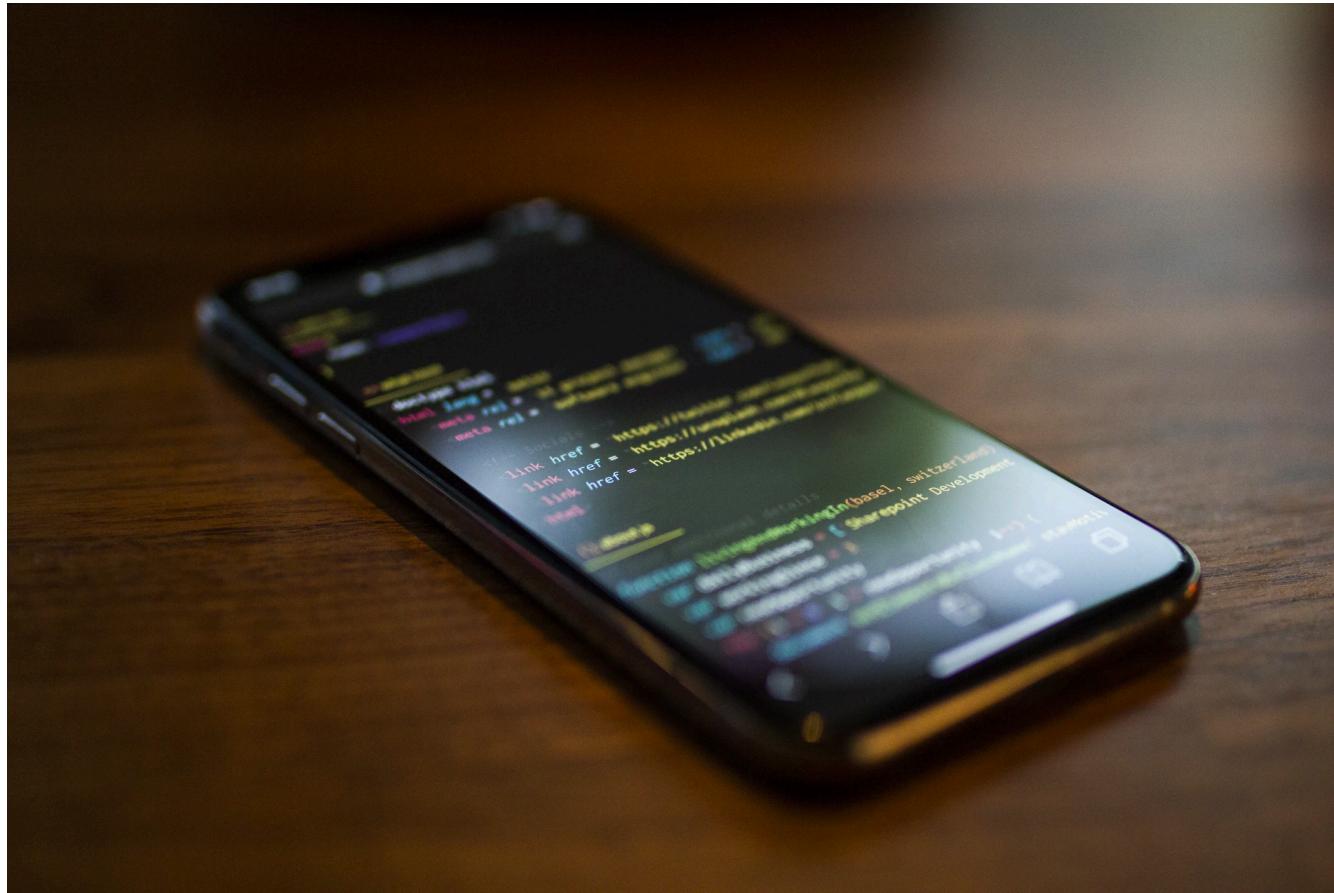
- Are there any values that would reveal a bug?

B:

```
for(int i=0; i < array.length; i++)  
{  
    System.out.print(array[i] + " ");  
    if ( array[i] < 0) negatives++;  
    else positives++;  
}
```



# Quality as a Goal

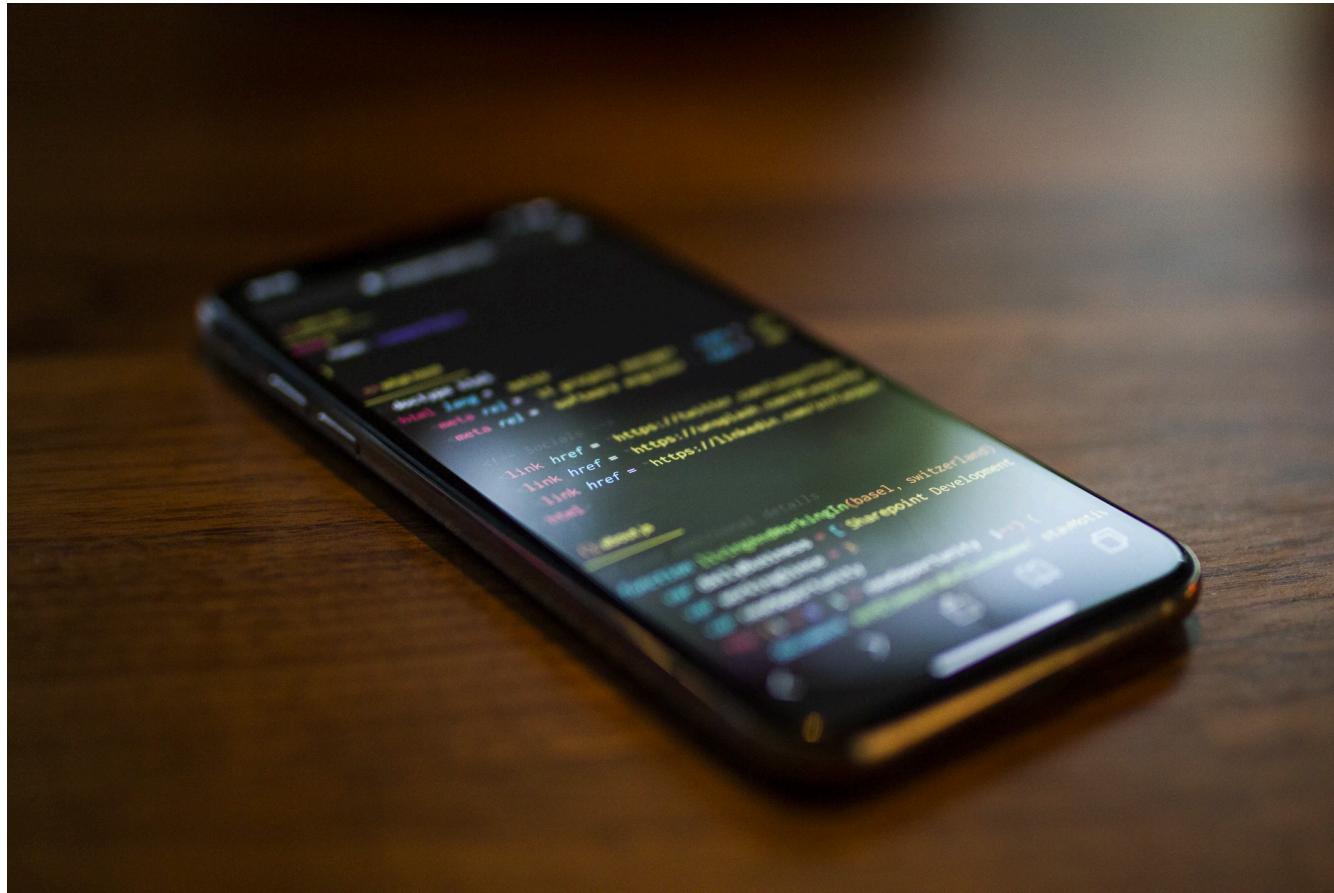


**Audience participation  
question:**

Examples of good software vs  
bad software?

How did you evaluate them?

# Quality as a Goal



**Verification:** Does the product fulfil the requirements?  
- Build the product right.

**Validation:** Does the product fit with intended usage?  
- Build the right product.

# Wait, "bugs"?

Well, technically:

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

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

**Software Failure:** External, incorrect behaviour with respect to the requirements or other description of the expected behaviour.



# “bugs” for short



# So, how do we know something is wrong?

**Testing:** Evaluating software by running it.

**Test Failure:** Execution that yields the “wrong” result.

**Debugging:** The process of finding a fault that caused a failure.

**Test Case Values:** The input values necessary to complete some execution of the software under test.



# How do we know a “wrong” result?



**Expected Results:** program's intended behaviour.

**Software Observability:** How easy it is to observe the behaviour of a program.

**Software Controllability:** How easy it is to provide a program with the needed inputs.

# Coverage as a means of evaluation

What values we should use?

- Are there any values that would reveal a bug?

A:

```
if ( a >= 0 ) return "a is positive";  
else return " A BUUUUG! ";
```



# Coverage as a means of evaluation

What values we should use?

- Are there any values that would reveal a bug?

B:

```
for(int i=0; i < array.length; i++)  
{  
    System.out.print(array[i] + " ");  
    if ( array[i] < 0) negatives++;  
    else positives++;  
}
```



# Enough theory, let's get to code!

```
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



Questions so far?

---

# Exercise for the Seminar



**Rooms:**

**A3-01**

**A4-01**

**A4-03**

**A5-18**

# Exercise for the Seminar



In groups (self-organized, so pick a room you like, chat to the people there, figure it out):

- Discuss previous projects, especially **JavaScript** and the current **OOP exercises**
- What did you test in the past? Do you remember what tests you ran?
- What parts of the code were/weren't covered?
- What inputs did you use?

# Exercise for the Seminar



Select a module you find interesting from one of the projects:

- Discuss, for existing tests (or tests that you remember), how execution proceeds through the code
- Write test cases that cover different execution paths. (Try to cover as many as possible)
- Do you think the module is ready for delivery now, and why?

# More definitions

Definition 1.17 Test Case: A test case is composed of the test case values, expected results, prefix values, and postfix values necessary for a complete execution and evaluation of the software under test.

Definition 1.18 Test Set: A test set is simply a set of test cases.

Definition 1.19 Executable Test Script: A test case that is prepared in a form to be executed automatically on the test software and produce a report.