cphbusiness

COPENHAGEN BUSINESS ACADEMY

Automated test

Car diagnose computer

cphbusiness

Car diagnose computer

- Car has to be build to be tested
- Each test will tell of a potential error
- Only errors that matters will be reported
- If all tests succeed, it is safe to drive the car

Class diagnosis

- Class~~Car~~ has to be build to be tested
- Each test will tell of a potential error
- Only errors that matters will be reported
- If all tests succeed, commit ~~drive~~ the Class ~~car~~

- Which aspects of the ~~Car~~Class should we test?
- How to make it easy to test a ~~Car~~Class

- How to automate the test
  - To make it easy to test after you have made changes

Test concepts – expected behaviour

- To test a method, we need to specify what are the expected behaviour
  - Behaviour:
    - Return value
    - Thrown exceptions
    - Calls to other objects
    - Changes to the object of the method
    - Changes to external components (for example database)

Three *levels* of tests

- **Unit tests**
  - At the level of individual methods and classes
  - Tool: JUnit
- **Integration tests**
  - Across layers
  - Tool: Mock objects
- **System tests**
  - The whole system
  - Tool: Mock, Performance

Specialized tests

- User interface tests
  - Does the user inteface work correctly?
  - Tools: Specialized automated input control
- Acceptance tests
  - A contract between the developers and the product owner on when a user story is implemented
  - Tools: Sometimes automates, sometimes not
- User testing
  - Letting future users of the system try to use a preliminary version of the system to see if they can use the program to solve their tasks.

Testing in the polygon project

- **Unit tests**
  - **Must have some**
- Integration tests
  - Should have one
- ~~System tests~~
- ~~User interface tests~~
- Acceptance tests
  - Cool if you do
- ~~User testing~~

White-box & black-box testing

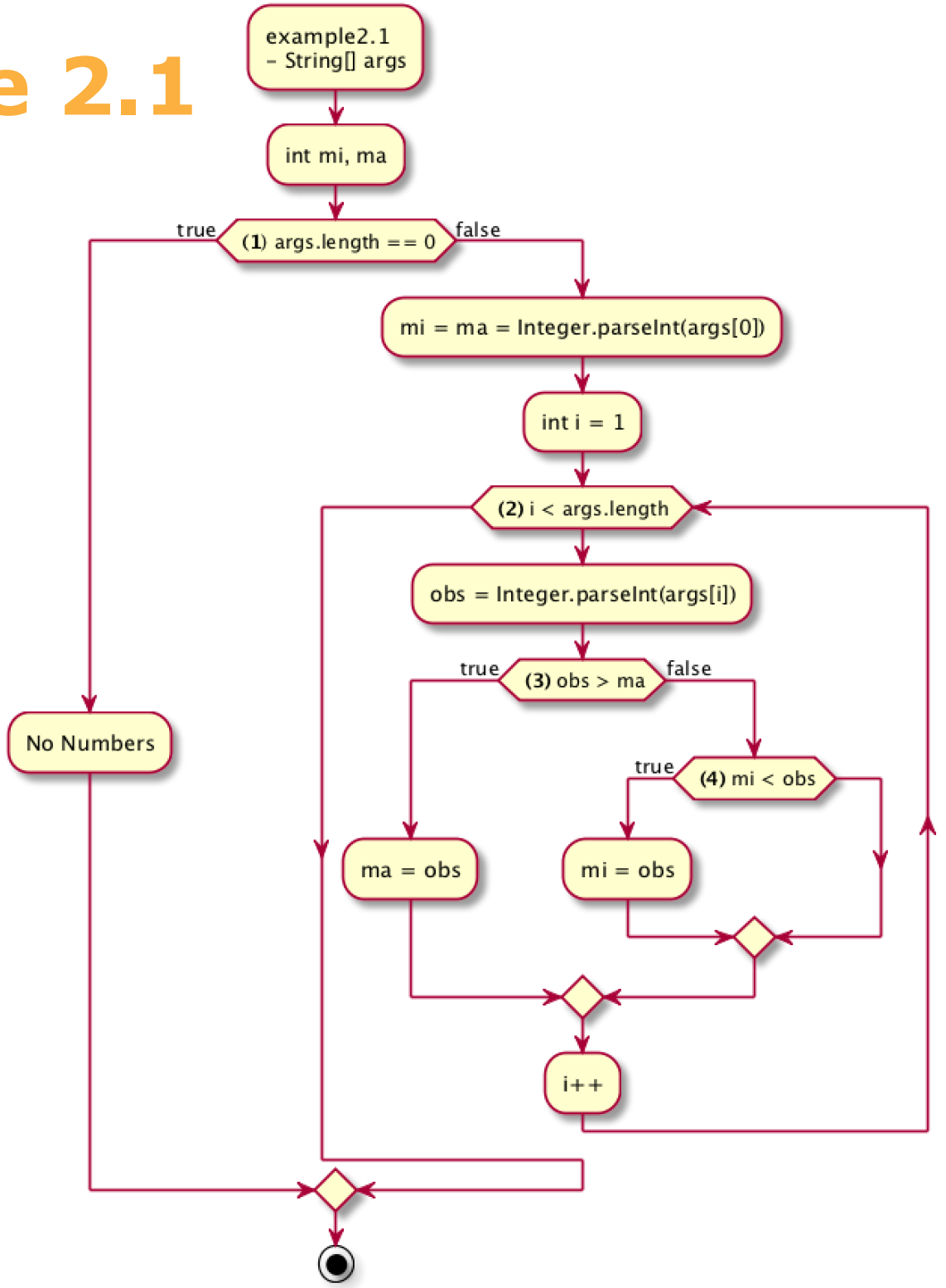
- White-box testing focuses on the text of the program.
  - The tester constructs a test suite that demonstrates that all branches of the program can be executed.
  - The test suite is said to cover the statements of the program.

- Black-box testing focuses on the problem that the program is supposed to solve
  - More precisely, the problem statement or specification for the program.

Example 2.1

```
public static void main ( String[] args )
{
  int mi, ma;
  if (args.length == 0)                                    /* 1 */
    System.out.println("No numbers");
  else
    {
      mi = ma = Integer.parseInt(args[0]);
      for (int i = 1; i < args.length; i++)                /* 2 */
        {
          int obs = Integer.parseInt(args[i]);
          if (obs > ma) ma = obs;                          /* 3 */
          else if (mi < obs) mi = obs;                     /* 4 */
        }
      System.out.println("Minimum = " + mi + "; maximum = " + ma);
    }
}
```

Example 2.1



Constructing test input

The resulting test suite includes enough input data sets to make sure that:

- all methods have been called,
- that both the true and false branches have been executed in if statements,
- that every loop has been executed zero, one, and more times – *why is "more times" important?*
- that all branches of every switch statement have been executed.

For every input data set, the expected output must be specified also.

Test coverage

- In practice white box testing is done on
  - Algorithms
  - To analyze for security holes

- Normally one use **automated test coverage** tools:
  - Measures to which extend our code has been tested
  - Integrated into Netbeans – colors your code red, green, yellow
  - It is pragmatic and "good enough".