CISC/CMPE 327 Software Quality Assurance

Queen's University, 2019-fall

Lecture #8

Introduction to Systematic Testing, part 1

Introduction to Systematic Testing

- Outline
  - Today we begin a thorough look at software testing
  - Definitions: What is software testing?
  - Role of specifications
  - Levels of testing:
    unit, integration, system, acceptance

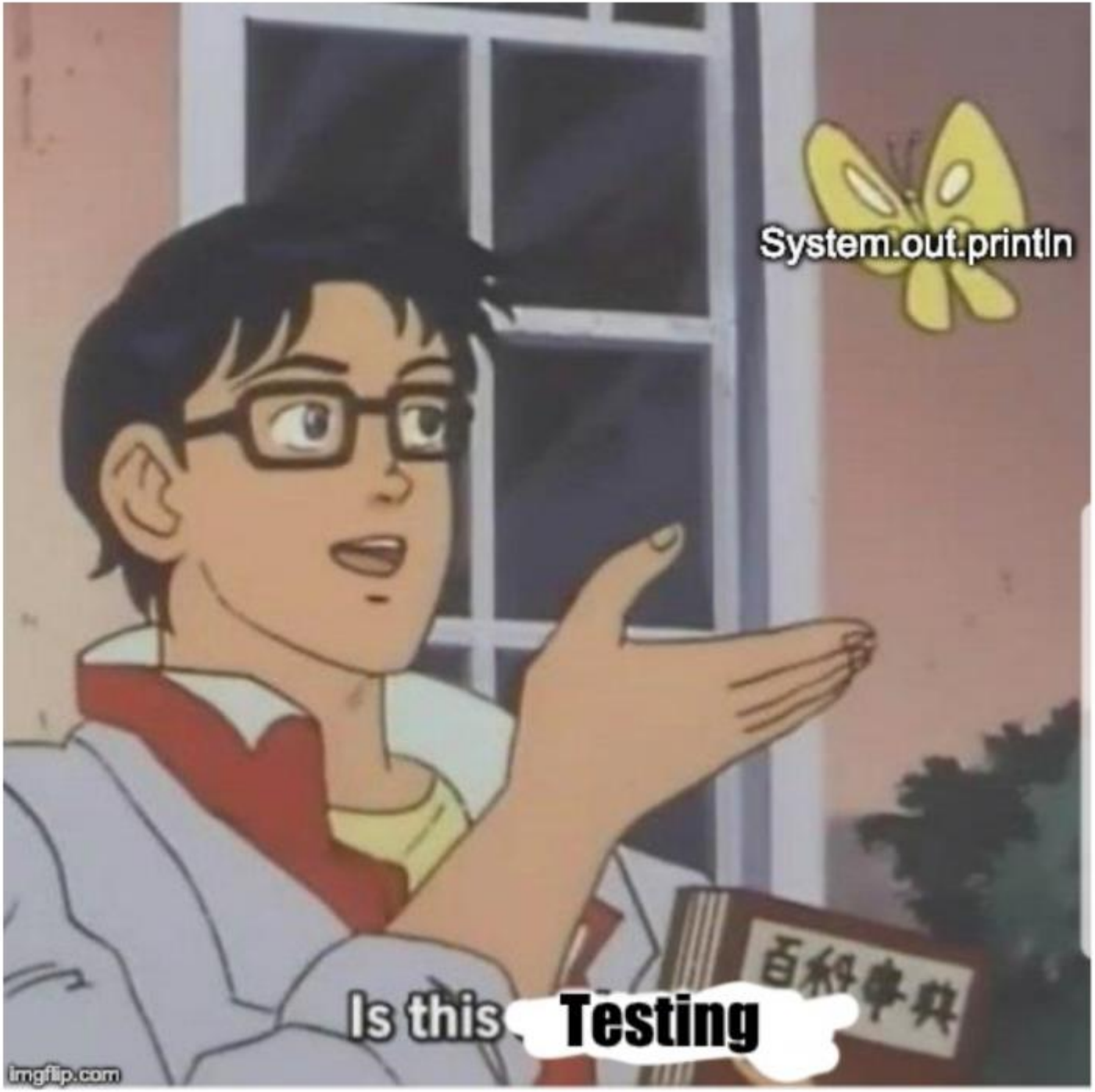
Me: I don't need to test this functionality...

Functionality: *breaks immediately*

Me:

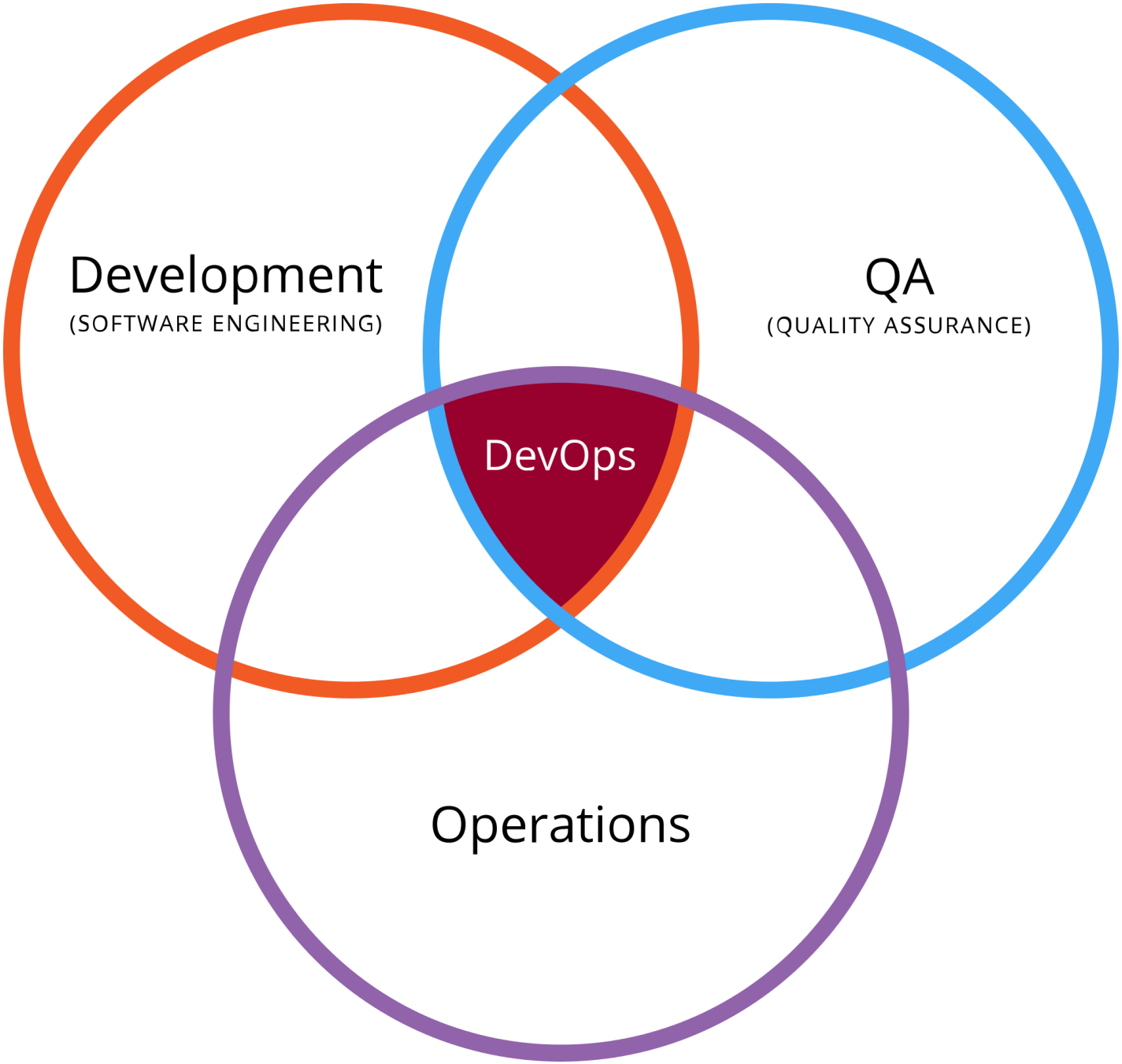
What is Testing?

- Testing is the process of executing software in a controlled manner to answer a question:
  - "Does the software behave as specified?"
- Specification
- Properties
- Testing is often associated with the words validation and verification


System.out.println
Is this
Testing
imgflip.com

What is Systematic Testing?

- An explicit discipline or procedure (a **system**) for
  - choosing and creating test cases
  - executing the tests and documenting the results
  - evaluating the results, ~~possibly~~ **automatically**
  - deciding when we are done (enough testing)


Development
(SOFTWARE ENGINEERING)
QA
(QUALITY ASSURANCE)
DevOps
Operations

What is Systematic Testing?

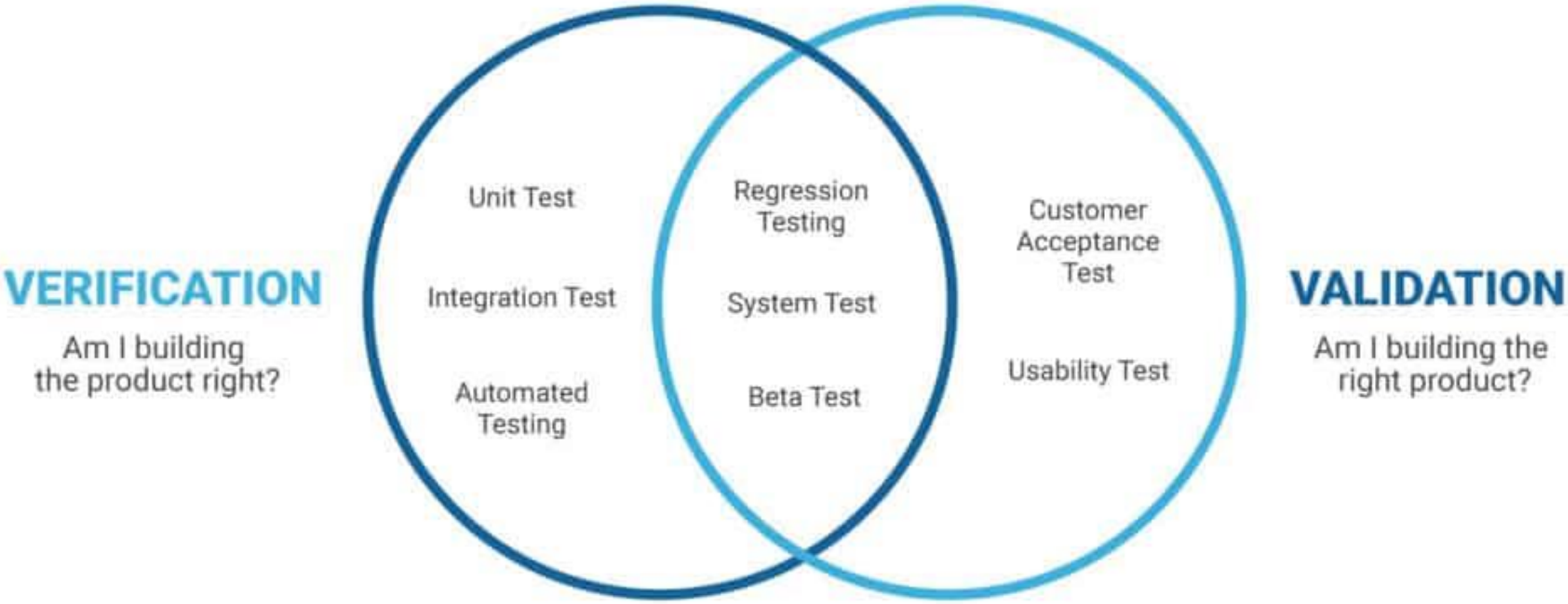
- Testing is at best complete
  - impossible to ever test completely
- Chooses a particular point of view and tests only from that point of view (the test criterion)
  - e.g., test only that every decision (if statement) can be executed either way

Verification vs. Validation

- Verification
  - Given a specification
    - Answers the question "are we doing the job right?"
  - Testing is most useful in verification
    - Inspection, measurement, analysis and formal methods are also important

Verification vs. **Validation**

- Validation
  - Answers the question "are we doing the right job?"
  - Meetings, reviews, and discussions to check that what has been specified is what was intended
    - Testing is less useful


VERIFICATION
Am I building the product right?
Unit Test
Integration Test
Automated Testing
Regression Testing
System Test
Beta Test
Customer Acceptance Test
Usability Test
VALIDATION
Am I building the right product?

Testing vs. Debugging

- Debugging is not Testing
  - Debugging -> analyzing and locating bugs [when something wrong]
  - Testing -> methodically searching for and exposing bugs

Debugging -> supports testing but cannot replace it

Exhaustive Testing vs. …

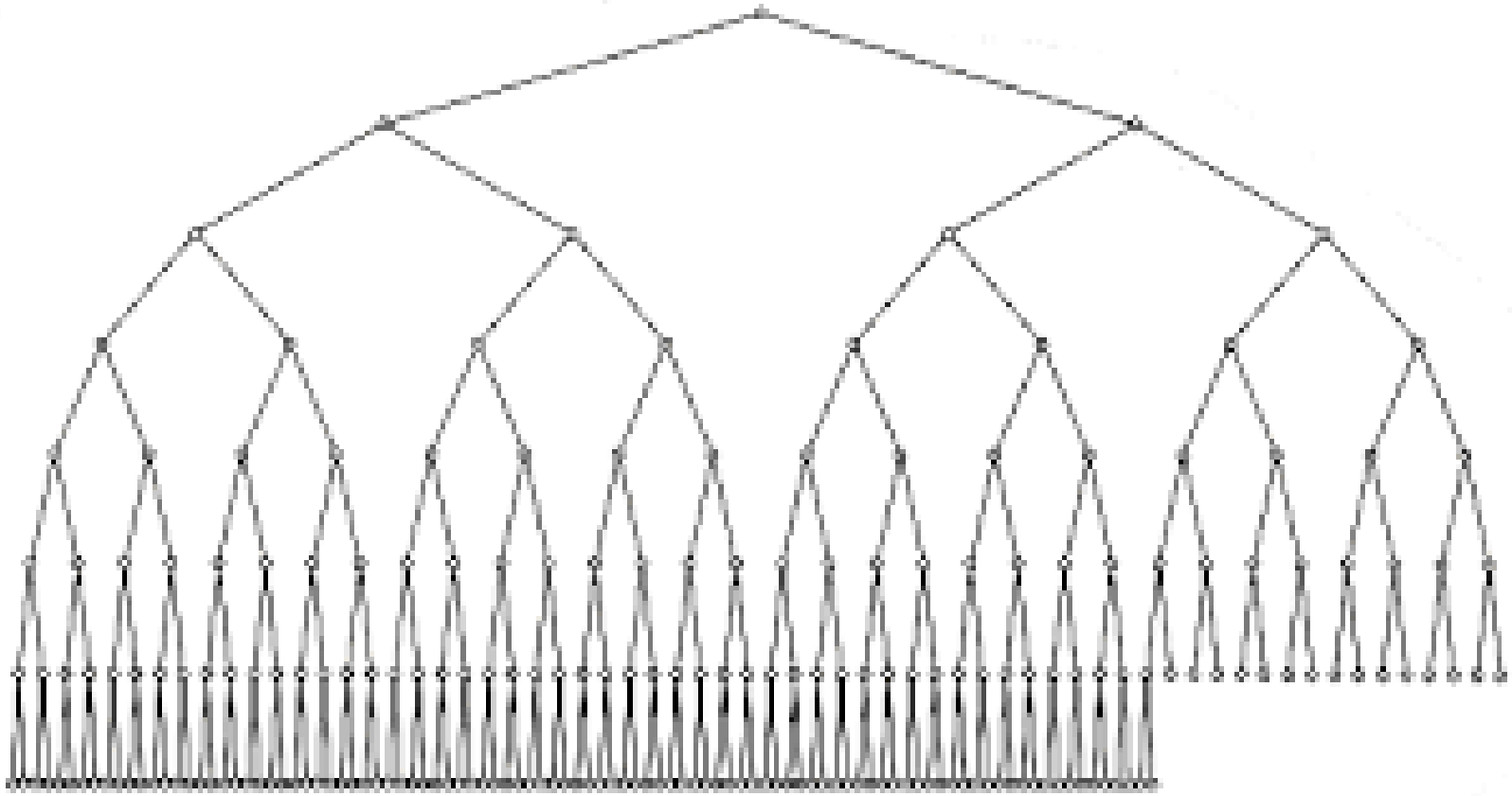
- NAND ("not AND") operation
  - Truth table:

A	B	A NAND B
true	true	false
true	false	true
false	true	true
false	false	true

  - Only 4 possible combinations of A and B, so 4 test cases

Exhaustive Testing vs. …

Path Explosion



The Role of Specifications

- The Need for Specification
  - Validation
  - Verification
- a single page
- a complex hierarchy of documents

Levels of Specification

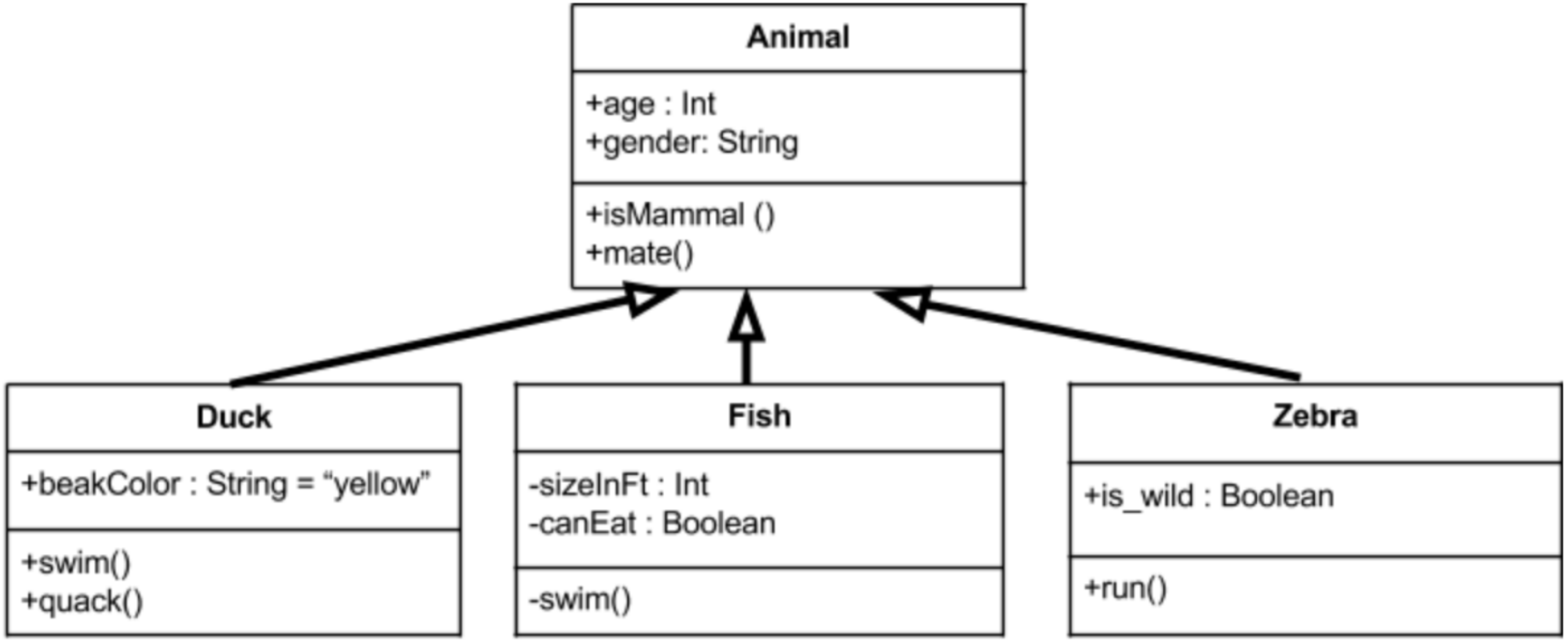
- Three Levels
  1. Functional specifications (or requirements)
  2. Design specifications
  3. Detailed design specifications

Levels of Specification

- 1) Functional specifications (requirements)
  - Precise description of the required behaviour (functionality) of the system
  - What the software should do, not how it should do it
  - Click "Exit" -> "Save" dialog if "has not been saved" -> otherwise "exit"

Levels of Specification

- 2) Design specifications
  - Describe the **architecture** of the design to implement the functional specification
  - Describe the components of the software and how they are to relate to one another


Animal
+age : Int
+gender: String
+isMammal ()
+mate()
Duck
+beakColor : String = “yellow”
+swim()
+quack()
Fish
-sizeInFt : Int
-canEat : Boolean
-swim()
Zebra
+is_wild : Boolean
+run()

Levels of Specification

- 3) Detailed design specifications
  - How to do (code)?
    - component of the architecture
    - individual code units,
  - Data Structure?
  - Data Storage?
  - Algorithms?
  - Input and expected outputs?
  - Invalid inputs?

Levels of Testing

- Corresponding Test Levels
  - 3) **Unit** testing addresses the verification that individual components of the architecture meet their detailed design specification

```java
import org.junit.*;
public class WriteAUnitTest {
    // JUnit calls this method one time before all tests
    @BeforeClass
    public static void setUp() {
        // Place code here for any set up required prior to tests
    }
    @AfterClass
    public static void finished() {
        // Place code here for any clean up that should be done after tests are finished
    }
    @Test
    public void testFirstName() {
        Person p=new Person();
        p.setFirstName("Stephen");
        Assert.assertEquals("Stephen", p.getFirstName());
    }
    @Test
    public void testLastName() {
        Person p=new Person();
        Assert.assertNotNull(p.getLastName());
    }
}
```

Levels of Testing

- Corresponding Test Levels
  - 2) **Integration** testing (a.k.a. component testing) verifies that the groups of units corresponding to architectural elements of the design specification can be integrated to work as a whole


Unit test vs. Integration test
freegifmaker.me

Levels of Testing

- Corresponding Test Levels
  - 1) **System** testing
    - verifies that the complete product meets the functional specification
  - 0) **Acceptance** testing
    - **validate** that
      - the software meets their real intentions
      - meet whatever functionally specified
      - accept the result

build passing coverage 100%

When she says those four special words

An Integral Task: Tests as Goals

- Each level of specification is written -> write the tests for that level
  - XP makes TESTS themselves the specification
- Tests be designed without knowledge of implementation
  - In XP, before implementation
- Otherwise we are tempted to simply test the software for what it actually does, not what it should do

Using Tests

- Evaluating Tests
  - Apply test
  - Evaluate test results
  - FAILED!
    a) the **tests are wrong**:
       UPDATE tests
    b) the **software is wrong**:
       FIX bugs
  - Back to Step 1 until 

Test Evolution

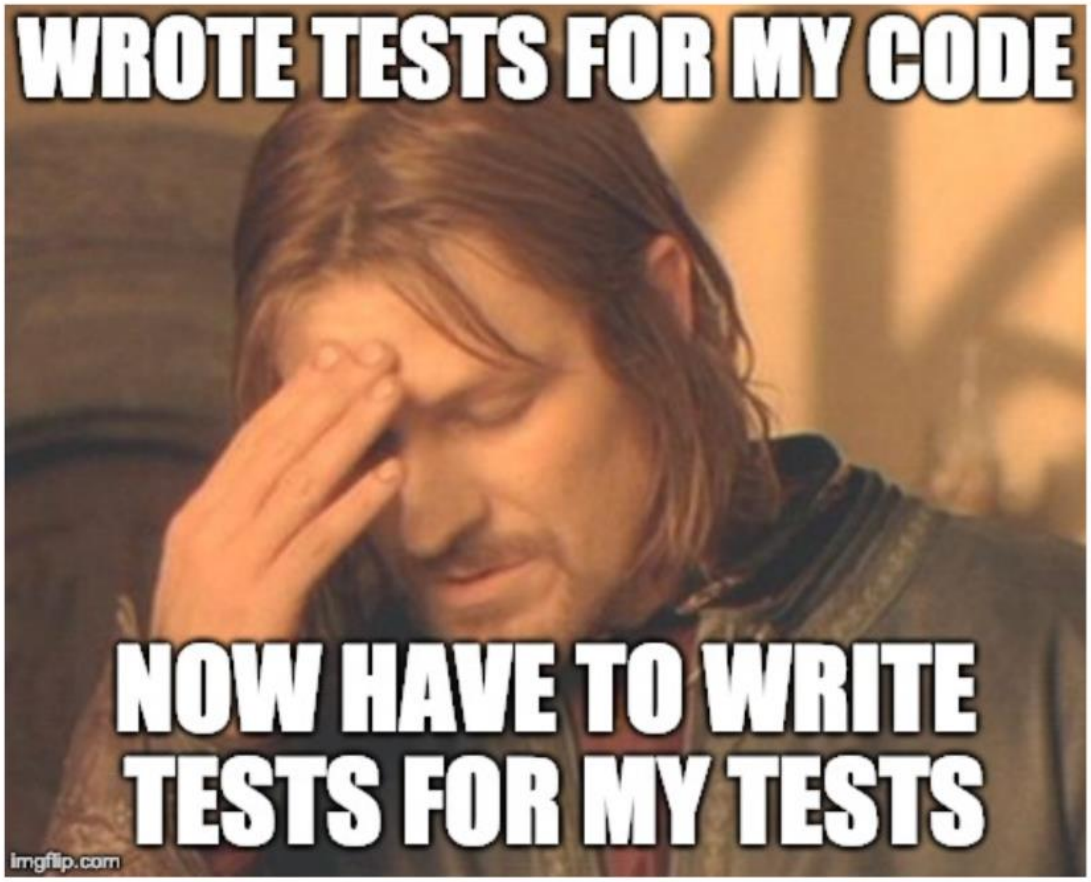
- Tests Don't Die!
  - Testing does not end when the software is accepted by the customer
  - Repeated, modified and extended
    - Continuously monitoring the failed parts when adding new features.

Test Evolution

- Tests Don't Die!
  - Maintenance of the tests!
    - That's how you control quality!
  - Practical continual testing
    - Automation

BUT!

- Test Adequacy



Summary

- Introduction to Testing
  - Testing addresses primarily the verification that software meets its specifications
  - Without some kind of specification, we cannot test
  - Testing is done at several levels, corresponding to the levels of functional, design, and detailed specifications in reverse order
  - Testing is not finished at acceptance, it remains for the life of the software system

Summary

- References
  - Sommerville, ch. 8, "Software Testing"
  - The Software Test Page (on the web)