CS 213 SOFTWARE METHODOLOGY

Lily Chang

CS Department @ Rutgers New Brunswick

FALL 2023

Overview of Software Testing

Lecture Note #7

Cost of Software Failure

- F-16 : crossing equator using autopilot
    - Result: plane flipped over
    - Reason? Reuse of autopilot software
- The Therac-25 accidents (1985-1987), (at least five died due to overdoses of radiation)
    - Reason: Bad event handling in the GUI program
- NASA Mars Climate Orbiter destroyed due to incorrect orbit insertion (September 23, 1999)
    - Reason: Unit conversion problem
- Boeing MAX 737 – lost hundreds of human lives
    - Reason: software bugs
- Volvo recalled 59,000 cars over software fault that can temporarily shut down the engine
    - Reason: software bugs

Terminology

Reliability – the probability that a software system WILL NOT cause system failure for a specified time under specified conditions (IEEE Std. 982-1989)

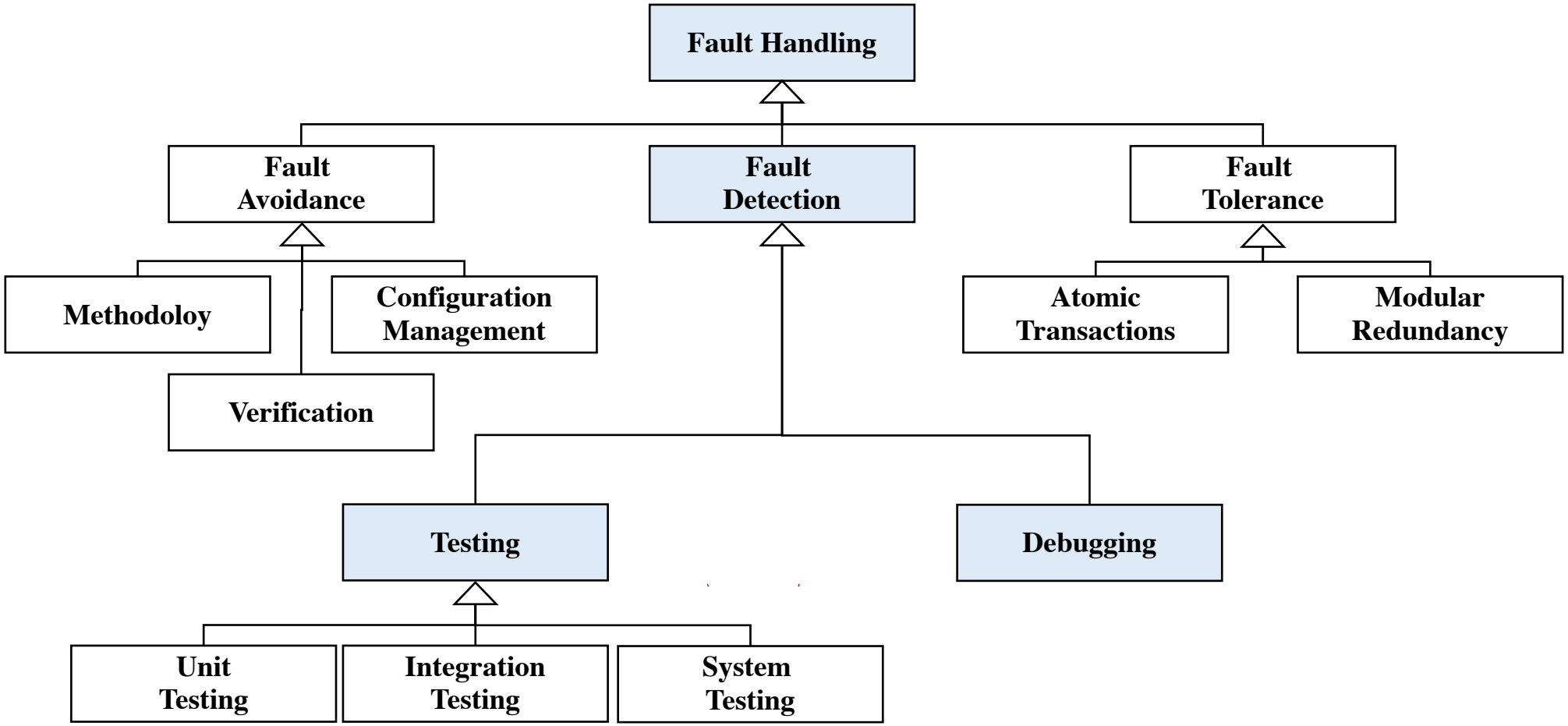
Failure – Any deviation of the observed behavior from the specified behavior

Erroneous state (error) – the system is in a state such that further processing by the system can lead to a failure

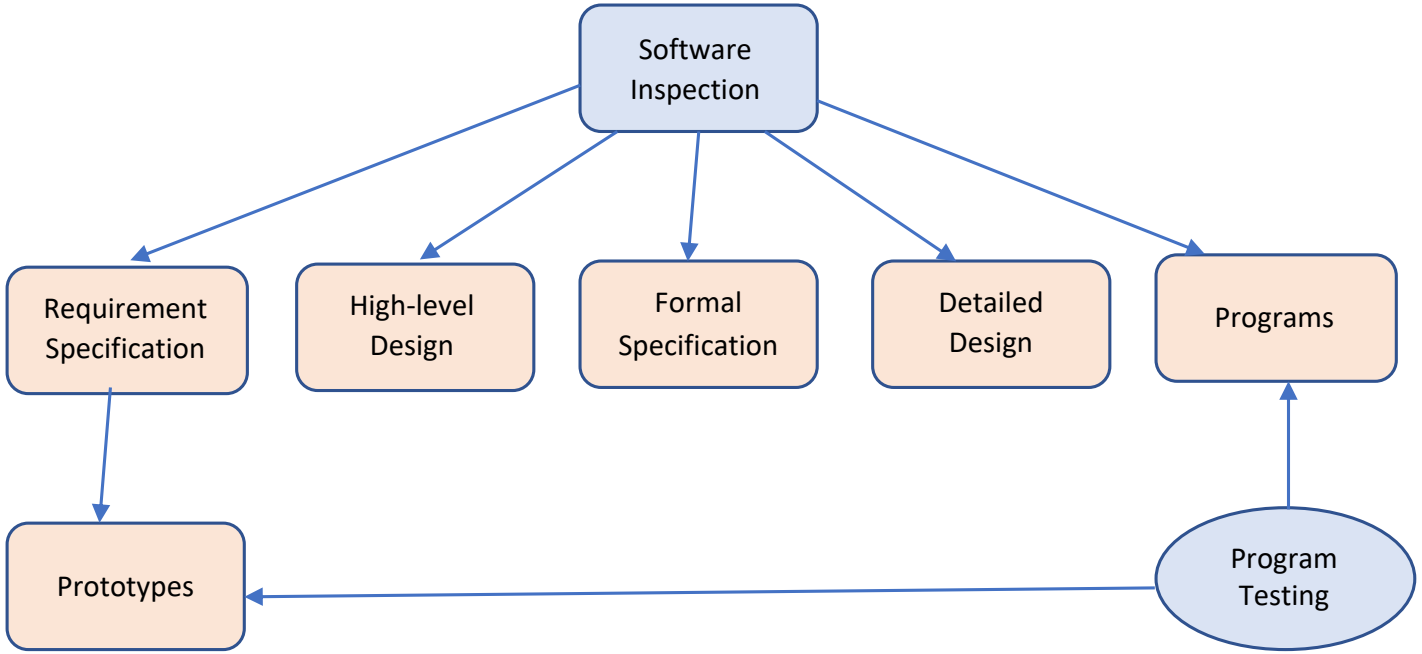
Fault (or defect, or bug) – the mechanical or algorithmic cause of an error

Testing – systematic attempt to find faults in a planned way in the implemented software

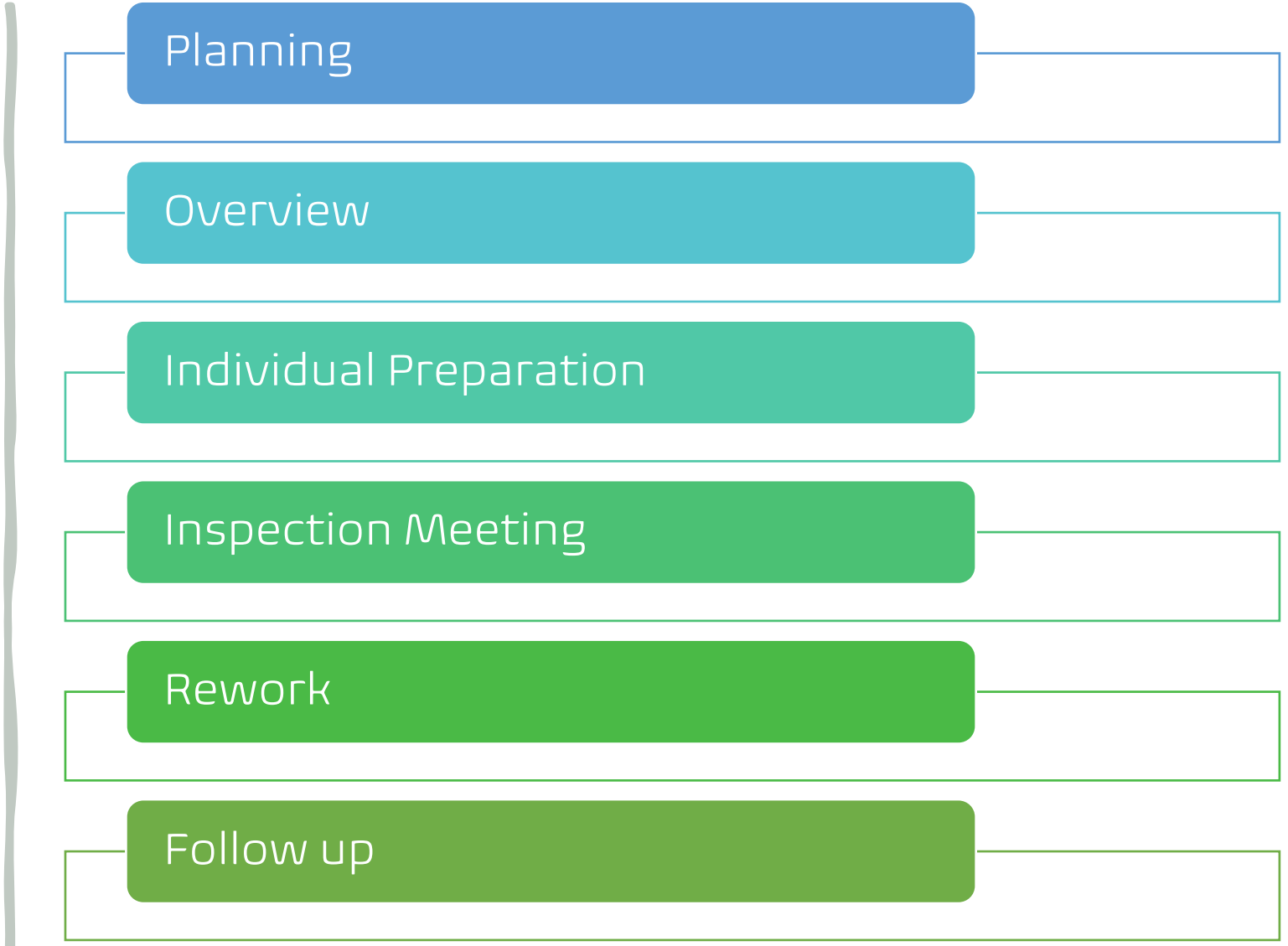
Fault Handling Techniques



Static and Dynamic Approach for Software Testing



Software Inspection Process – Static Approach



Software Testing – Dynamic Approach

- The software system is executed.
- The process of finding differences between the expected behavior specified by system models and the observed behavior of the implemented system.
- The attempt to show that the implementation of the system is inconsistent with the system models.
- The goal is to design tests that exercise defects in the system and to reveal problems.
- Software Testing is aimed at breaking the system!

Software Testing Plan

It is impossible to completely test any nontrivial module or system

Practical limitations – Complete testing is prohibitive in time and cost

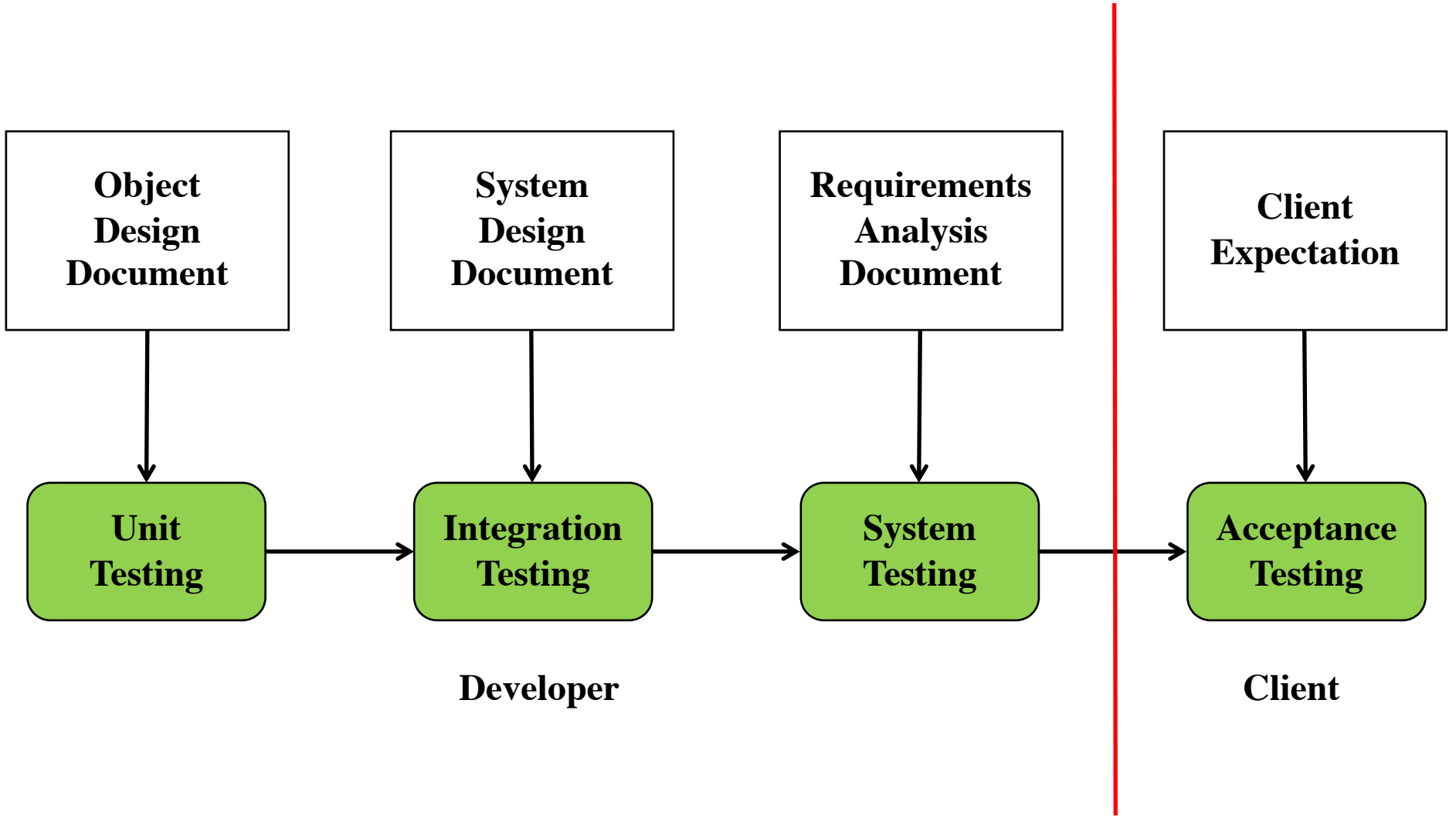
Theoretical limitations: e.g., Halting problem

“Testing can only show the presence of bugs, not their absence” (Dijkstra)

Testing is not for free

=> Define your goals and priorities!!

Software Testing Activities



Unit Testing

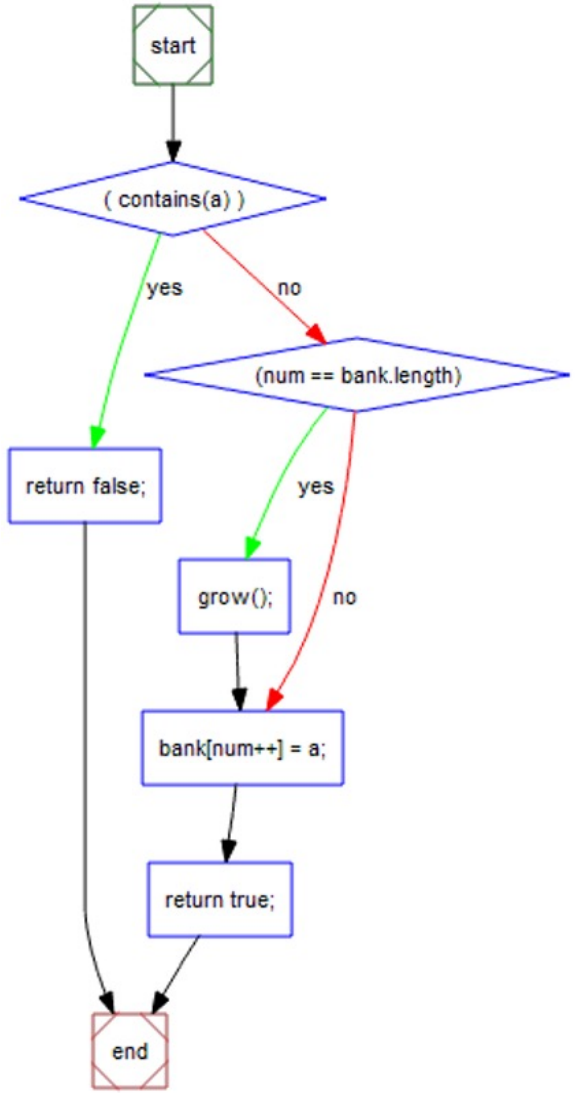
Individual component (class or subsystem)

Carried out by developers

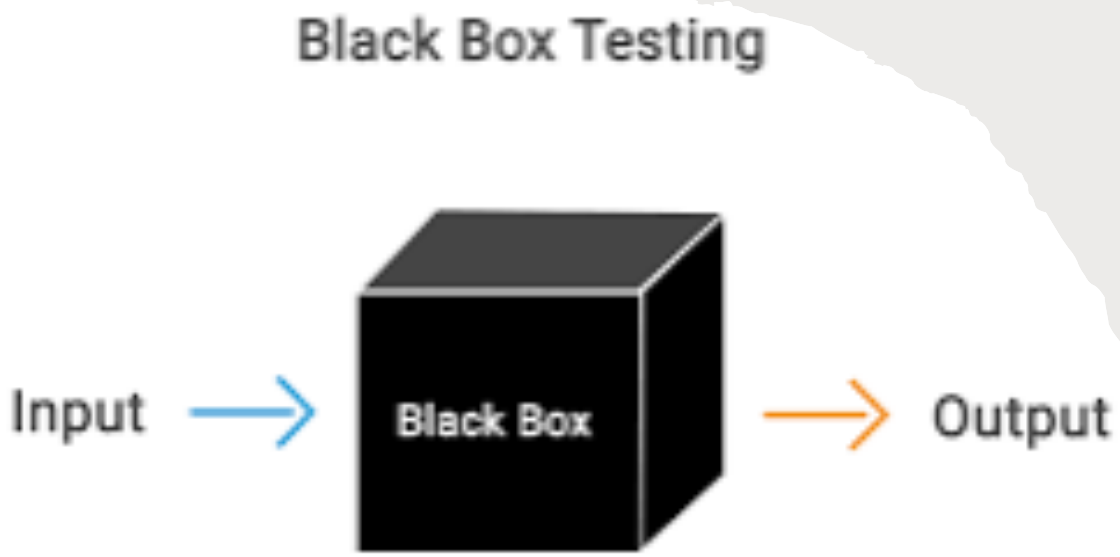
Goal: Confirm that the component or subsystem is correctly coded and carries out the intended functionality

Unit Testing Techniques

- Black-box testing
  - Functional testing
  - Does not focus on the implementation details
- White-box testing
  - Structural testing
  - Focus on the control structure and coverage of the code being exercised
  - Code coverage, Branch coverage, Condition coverage, Path coverage



Black-Box Testing



- Required Information: only requirement specification
- Independent of the implementation; test design can be in parallel with implementation
- Focus on the I/O behavior
- If for any given input, we can predict the output, then the component passes the test
    - Requires test oracle (expected test results)
- Goal – Reduce number of test cases by equivalence class partitioning
    - Divide input conditions into equivalence classes
    - Choose test cases for each equivalence class.

Black-Box Testing – Test case selection

Input is valid across range of values

- Developer selects test cases from 3 equivalence classes:
  - Below the range
  - Within the range
  - Above the range

Input is only valid if it is a member of a discrete set

- Developer selects test cases from 2 equivalence classes:
  - Valid discrete values
  - Invalid discrete values

Boundary value analysis

- test cases at the boundary
- min - 1 and max + 1

Example – Black box testing

```
public int getNumDaysInMonth(int month, int year) { }
```

Representation for month:

1: January, 2: February, ..., 12: December

Representation for year:

1904, ... 1999, 2000,..., 2006, ...

How many test cases do we need for the black box testing of getNumDaysInMonth() method?

Example– Equivalence classes

- For the month parameter,
  - Valid – 3 equivalence classes
    - Months with 31 days, JAN, MAR, MAY, JUL, AUG, OCT, DEC
    - Months with 30 days, APR, JUN, SEPT, NOV, and
    - February can have 28 or 29 days
- For the year parameter,
  - Valid – 2 equivalence classes: Leap years and non-leap years (year >= 1900)

Example – Test cases selection

Equivalence class for valid input	Input		Expected Output
	month	year	
Months with 31 days, non-leap years	7 (July)	1901	31
Months with 31 days, leap years	7 (July)	1904	31
Months with 30 days, non-leap years	6 (June)	1901	30
Months with 30 days, leap years	6 (June)	1904	30
Months with 28 or 29 days, non-leap years	2 (February)	1901	28
Months with 28 or 29 days, leap years	2 (February)	1904	29

Boundary Testing

- Special case of equivalence testing focuses on the conditions at the boundary of the equivalence classes
- Select elements from the “edges” of the equivalence class

EQUIVALENCE CLASS	VALUE FOR MONTH INPUT	VALUE FOR YEAR INPUT	EXPECTED OUTPUT
Leap years divisible by 400	2 (February)	2000	29
Non-leap years divisible by 100	2 (February)	1900	28
lower bound of month	1	2000	31
upper bound of month	12	2020	31

Another Example

Given a Java method with the signature:

```
public double ticketPrice (int age, boolean isMember);
```

This method returns the ticket price for a single admission to a museum. The functional requirements are as follows.

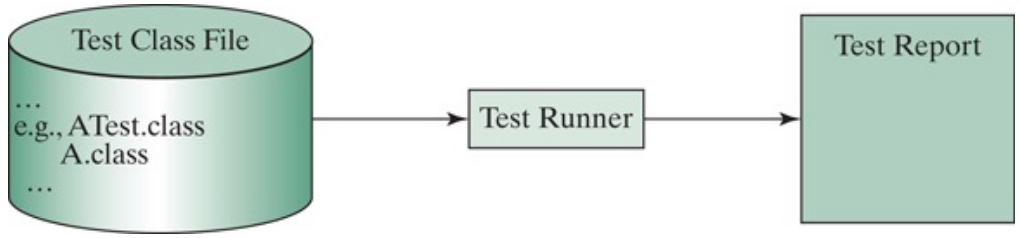
- The system shall be able to determine the ticket price for the admission based on the age and the membership status of a person.
- For the children 12 years old and under, the system shall apply the ticket price $12.
- People beyond 12 years old are considered as adults. The system shall apply the ticket price $20 for adults.
- The system shall use the ticket price $16 for senior citizens who are 65 or older.
- The system shall apply a 10% discount to the adult ticket price if the person is holding the museum membership.
- No discount for children and senior citizens' admissions.

Design the test cases for Black-Box testing with equivalence class partitioning.

EQUIVALENCE CLASS #	INPUT (PARAMETERS)		EXPECTED OUTPUT
	age	isMember	
1	<=12	true	12
2	<=12	false	12
3	>12 and <65	true	18
4	>12 and <65	false	20
5	>=65	true	16
6	>=65	false	16

JUnit Test Framework

- Software testing is expensive and tedious, thus use CASE (Computer Aided Software Engineering) tools as much as possible
  - Automate the tests by implementing test cases, so they are repeatable
  - Regression testing, refactoring, software change
- JUnit is the de facto framework for testing Java programs.
- JUnit is a third-party open-source library packed in a jar file, which contains a tool called test runner to run test programs



JUnit Test Framework

- IntelliJ includes the JUnit package
- Resources and documentation: https://junit.org/junit5

Useful Methods in JUnit Assertion class

assertTrue(boolean condition)

assertFalse(boolean condition)

assertNull(Object testobject)

assertEquals(Object expected, Object actual) //according to equals() method

assertEquals(int expected, int actual); //according to ==

assertEquals(double expected, double expected); //less than or equal to the tolerance value

assertSame(Object expected, Object actual); //if refer to the same object in memory

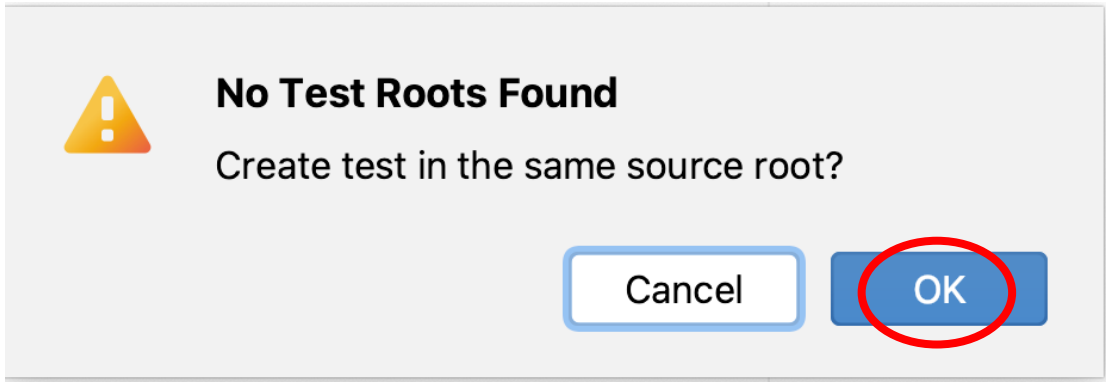
- Many more overloading methods:

https://junit.org/junit5/docs/current/api/org.junit.jupiter.api/org/junit/jupiter/api/Assertions.html

Five Steps of Unit Testing OO Software

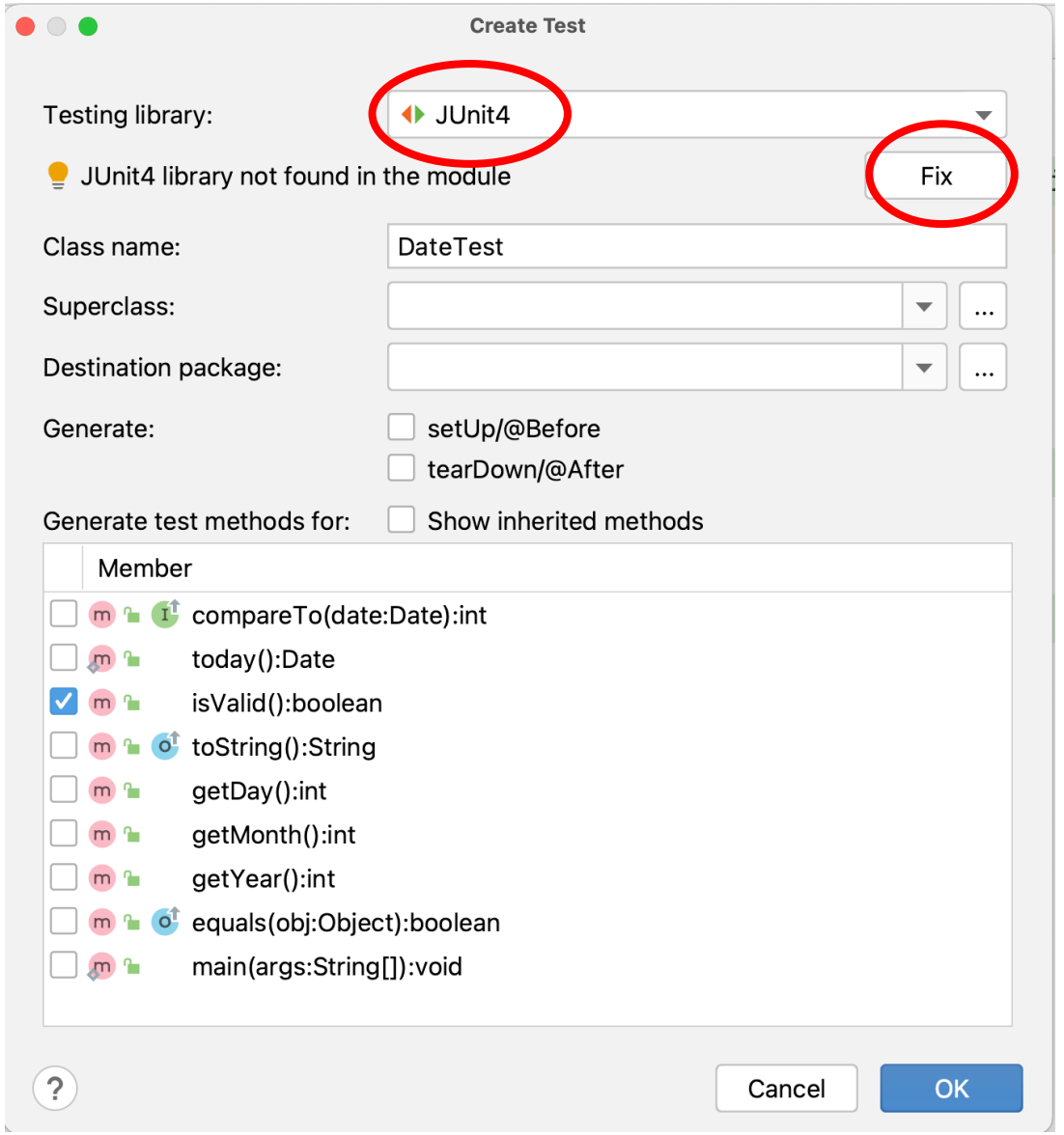
Create	Select	Compute	Execute	Verify
Create an object and select a method to execute	Select values (test cases) for the input parameters of the method	Compute the expected values to be returned by the method	Execute the selected method on the created object using the selected input values	Verify the result of executing the method Compare the expected output and the actual output

Create Test Classes in IntelliJ

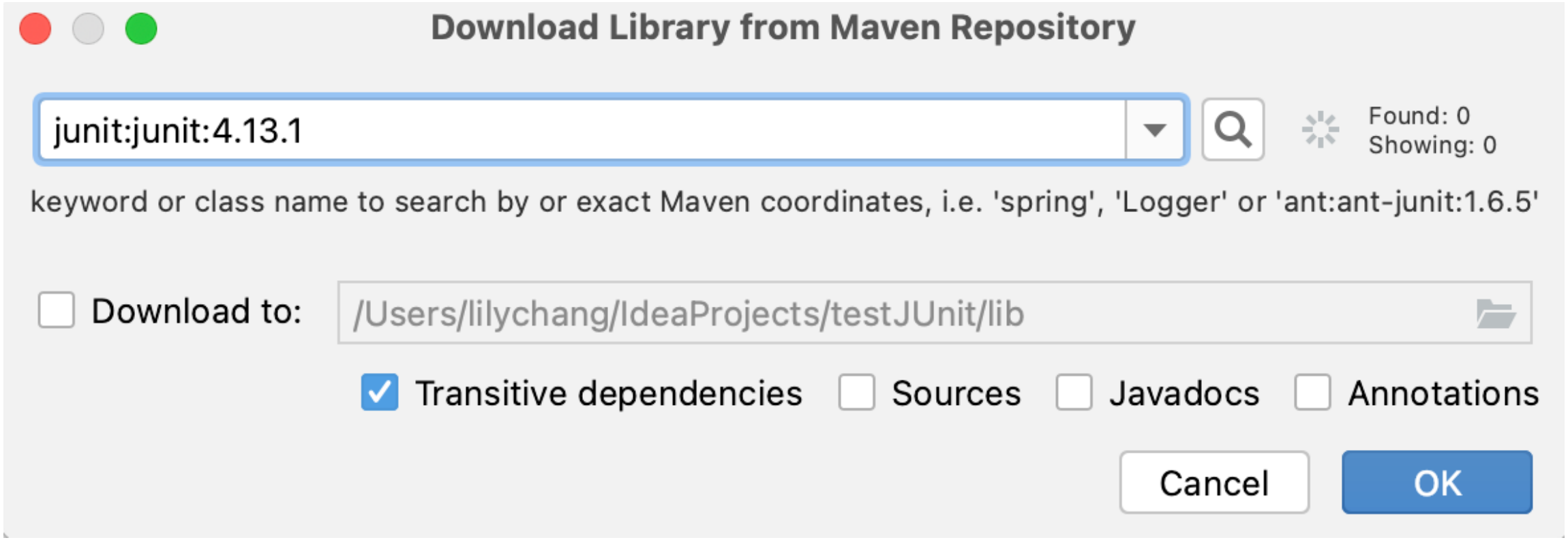


- Open the Java class you wanted to test
- Click on the class name, and select create test
- Click OK if you see the warning message below

SELECT THE JUNIT VERSION AND THE METHODS TO TEST



Download the Library



YOU MAY NEED TO ADD JUNIT TO THE CLASSPATH