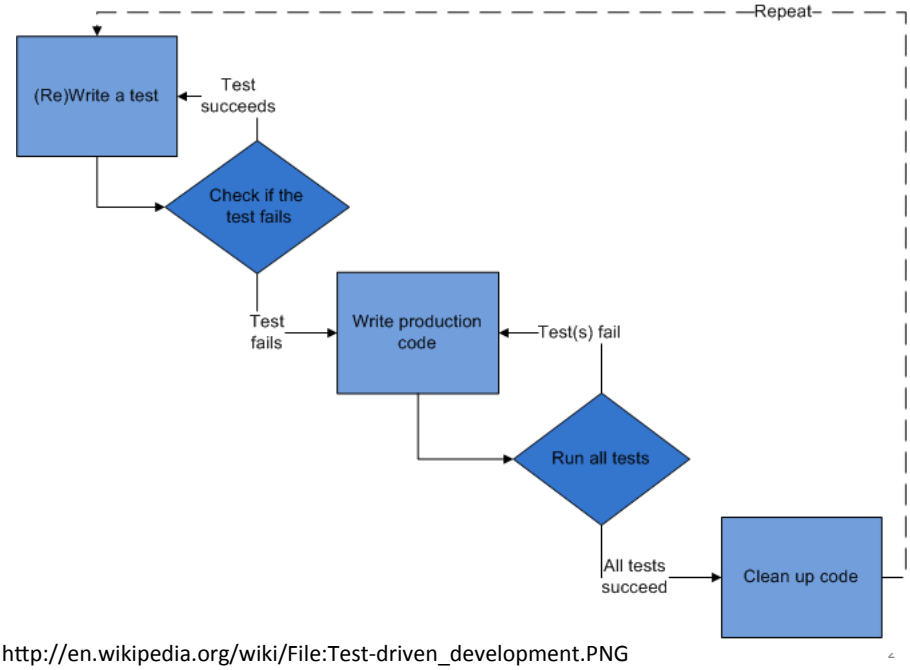
Test-Driven Programming

1



http://en.wikipedia.org/wiki/File:Test-driven_development.PNG

Unit Testing

- Unit test: a small piece of program written by the developer to test the correctness of a unit in a particular case.
- Unit: In the context of OOP, a method or a class.

3

Unit Testing in Java

- We will use JUnit and Eclipse.
- Case Study:
  - Rational class
  - Clock class

4

Rational

- Defines a class called **Rational** that represents rational numbers, which are simply the quotient of two integers (numerator and denominator)
- Rational numbers support the standard

Addition: $\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}$

Multiplication: $\frac{a}{b} \times \frac{c}{d} = \frac{ac}{bd}$

Subtraction: $\frac{a}{b} - \frac{c}{d} = \frac{ad - bc}{bd}$

Division: $\frac{a}{b} \div \frac{c}{d} = \frac{ad}{bc}$

5

From Eric Roberts, *The Art and Science of Java*, Addison Wesley.

Rational

- The constructors for the class are overloaded.
  - Calling the constructor with no argument creates a Rational initialized to 0,
  - calling it with one argument creates a Rational equal to that integer, and
  - calling it with two arguments creates a fraction.
- A rational number should be normalized: numerator and denominator are reduced to lowest terms.

6

Clock

- Define a class named Clock that represents the daily time.
- A clock keeps the hour, minute and second.
- A clock is printable via **toString()** in the uniform format HH:MM:SS.

7

Exercise

- A **tick()** method increments the time of a clock by one second.
- Write test cases for the **tick()** method.

8

F.I.R.S.T.

- **Fast**
  - Tests should be fast.
  - They should run quickly. When tests run slow, you won't want to run them frequently.
  - If you don't run them frequently, you won't find problems early enough to fix them easily.
  - You won't feel as free to clean up the code. Eventually the code will begin to rot.

9

F.**I**.R.S.T.

- **Independent**
  - Tests should not depend on each other.
  - One test should not set up the conditions for the next test.
  - You should be able to run each test independently and run the tests in any order you like.

10

F.I.**R**.S.T.

- **Repeatable**
  - Tests should be repeatable in any environment.
  - You should be able to run the tests
    - in the production environment,
    - in the QA environment, and
    - on your laptop while riding home on the train without a network.
  - If your tests aren't repeatable in any environment, you'll always have an excuse for why they fail.

11

F.I.R.**S**.T.

- **Self-Validating**
  - The tests should have a boolean output.
  - Either they pass or fail.
  - You should not have to read through a log file to tell whether the tests pass.
  - You should not have to manually compare two different text files to see whether the tests pass.

12

F.I.R.S.**T.**

- **Timely**
  - The tests need to be written in a timely fashion.
  - Unit tests should be written *just before* the production code that makes them pass.
  - If you write tests after the production code, you may find the production code to be hard to test.
  - You may decide that some production code is too hard to test.
  - You may not design the production code to be testable.

13