

# Test Driven Development and Unit Test

# Contents

- TDD (Test Driven Development) ( 1 / 2 )
  - What is TDD ?
  - TDD Cycle
  - Benefit of TDD
  - Discussion

# Contents

- Unit Test ( 2 / 2 )
  - What is Unit Test ?
  - Goal
  - Mock Object
  - Benefit

# Have you ever been in these situation ?

เคยไหมที่ไม่รู้จะเริ่มเขียนโปรแกรมยังไงดี

*No idea where to start ?*

เคยไหมที่เขียนไปแล้วโปรแกรมเพิ่มขึ้นเรื่อยๆ แต่ไปไม่ถึง Goal ซักที

*Plenty of code, and seem to be further but still not  
achieve the goal*

# Have you ever been in these situation ?

เคยไหมที่เขียนโปรแกรมไปแล้วพบกับปัญหา ทำให้ต้องกลับมาแก้ไขและเรียบเรียงใหม่

*Found a defect that have to fix or re-design at the very beginning ?*

เคยไหมที่ *function* ที่เขียนขึ้นมาไม่ได้ใช้ เพราะอยู่นอกเหนือจาก *requirement*

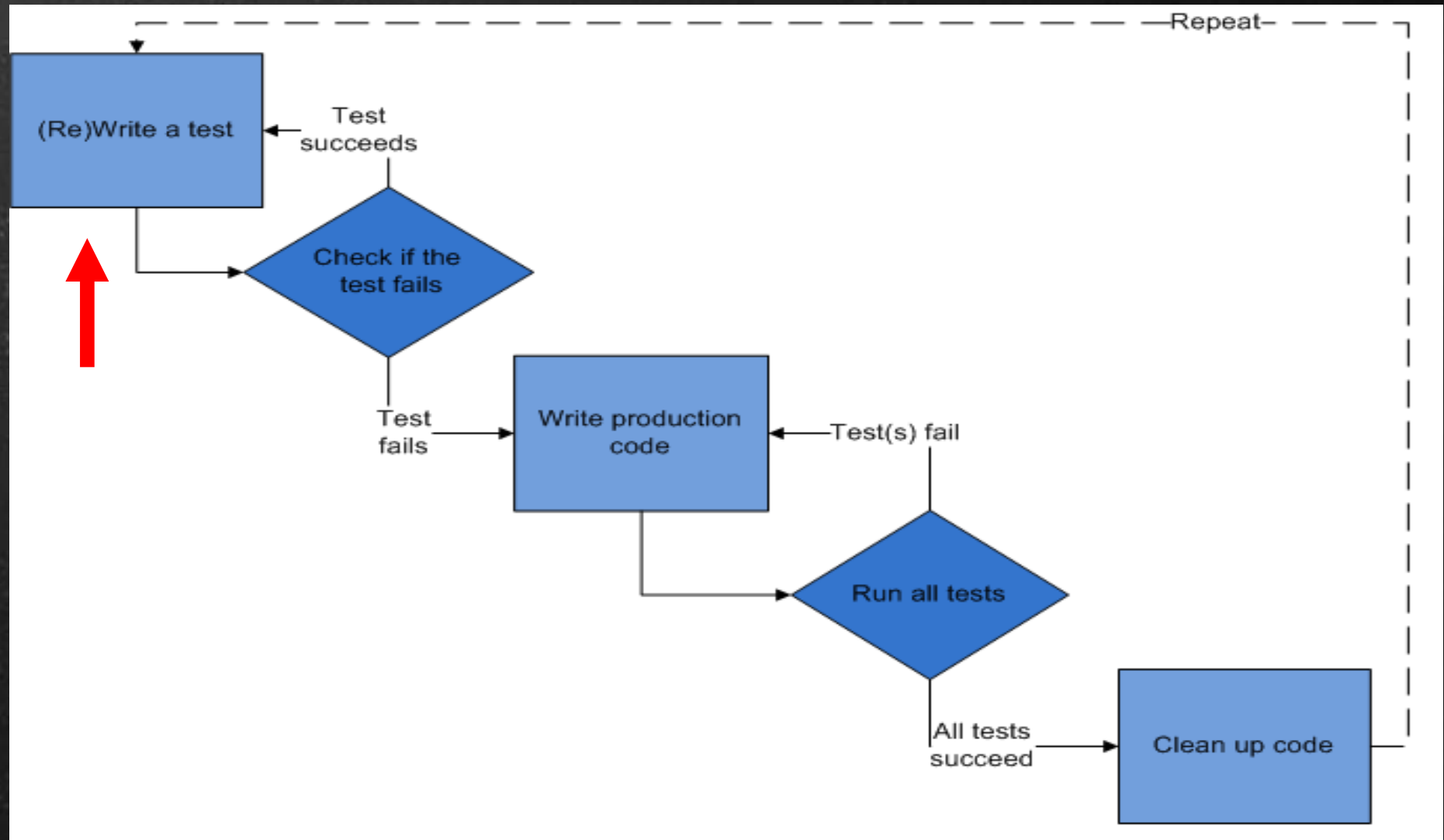
*Wasting time to implement a useless function that out of requirement*



# What is TDD?

Test Driven Development or TDD is a software development process that relies on the repetition of a very short development cycle.

# TDD Cycle



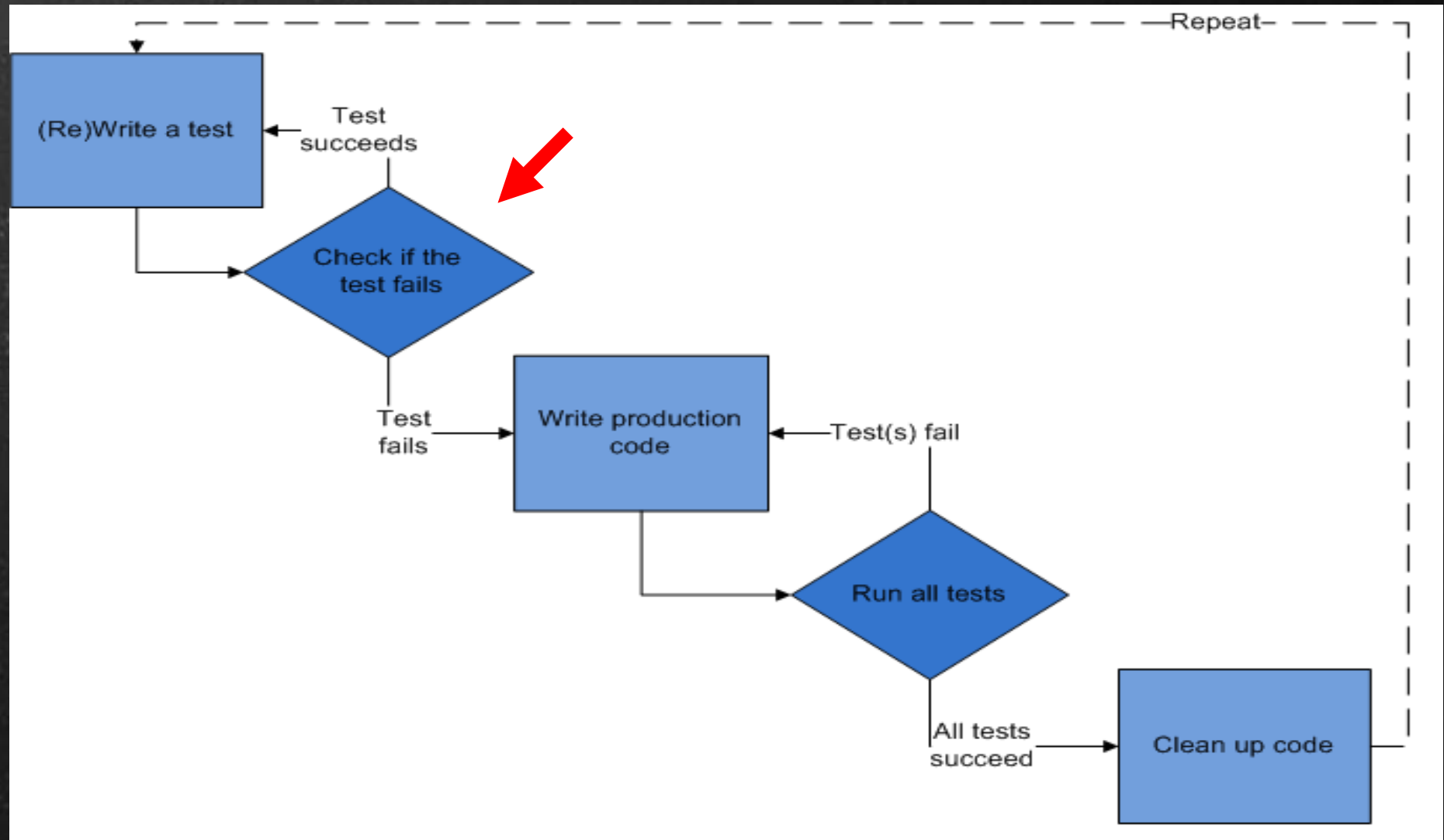
# How ?

1. Before writing any code, you must first write an automated test for your code. While writing the automated tests, you must take into account all possible inputs, errors, and outputs. This way, your mind is not clouded by any code that's already been written.

- To write a test, the developer must clearly understand the feature's specification and requirements.
- The developer can accomplish this through use cases and user stories that cover the requirements and exception conditions.



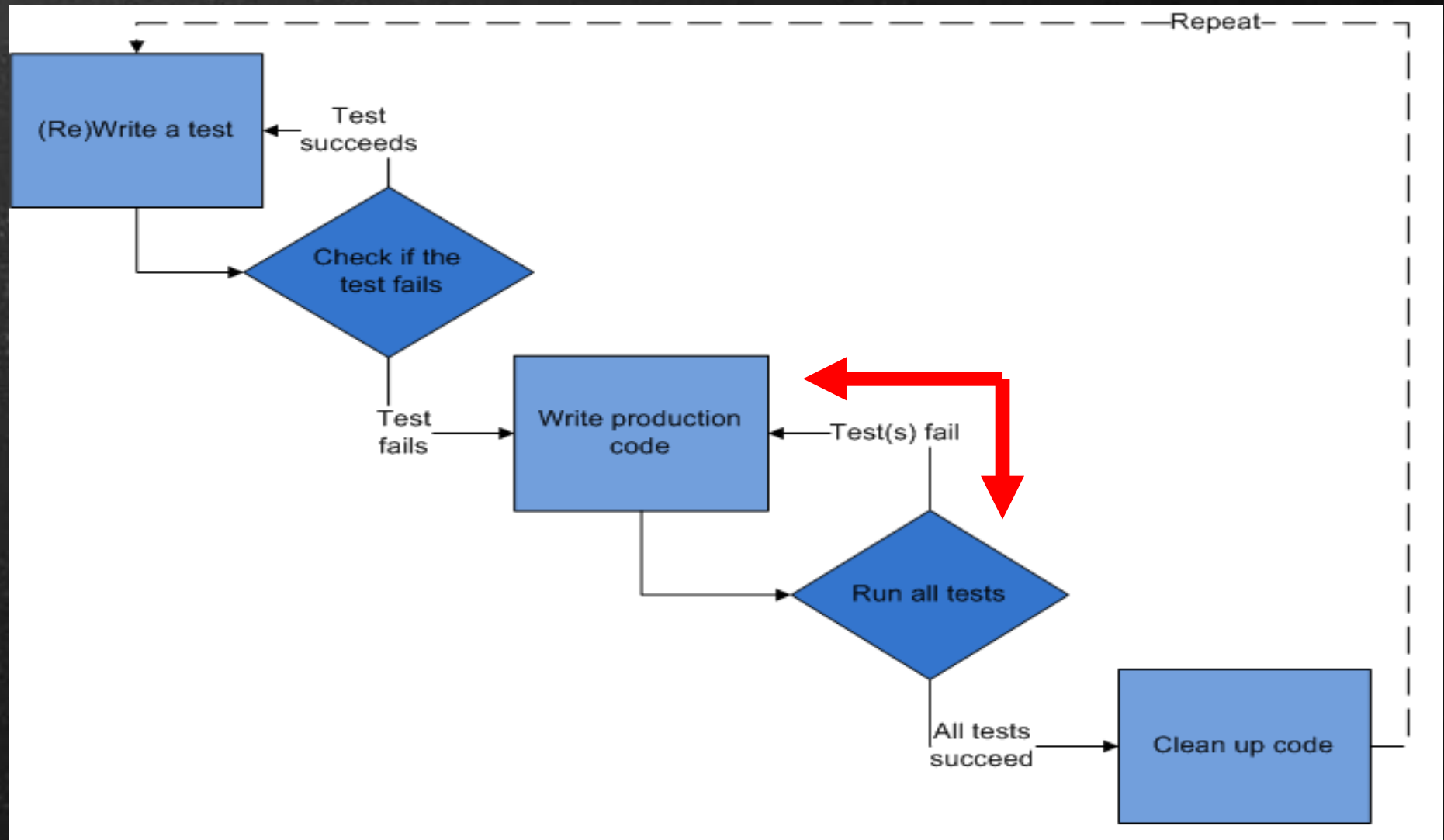
# TDD Cycle



2. The first time you run your automated test, the test should fail ( indicating that the code is not yet ready )

- If it does not fail, then either the proposed “new” feature already exists or the test is defective.

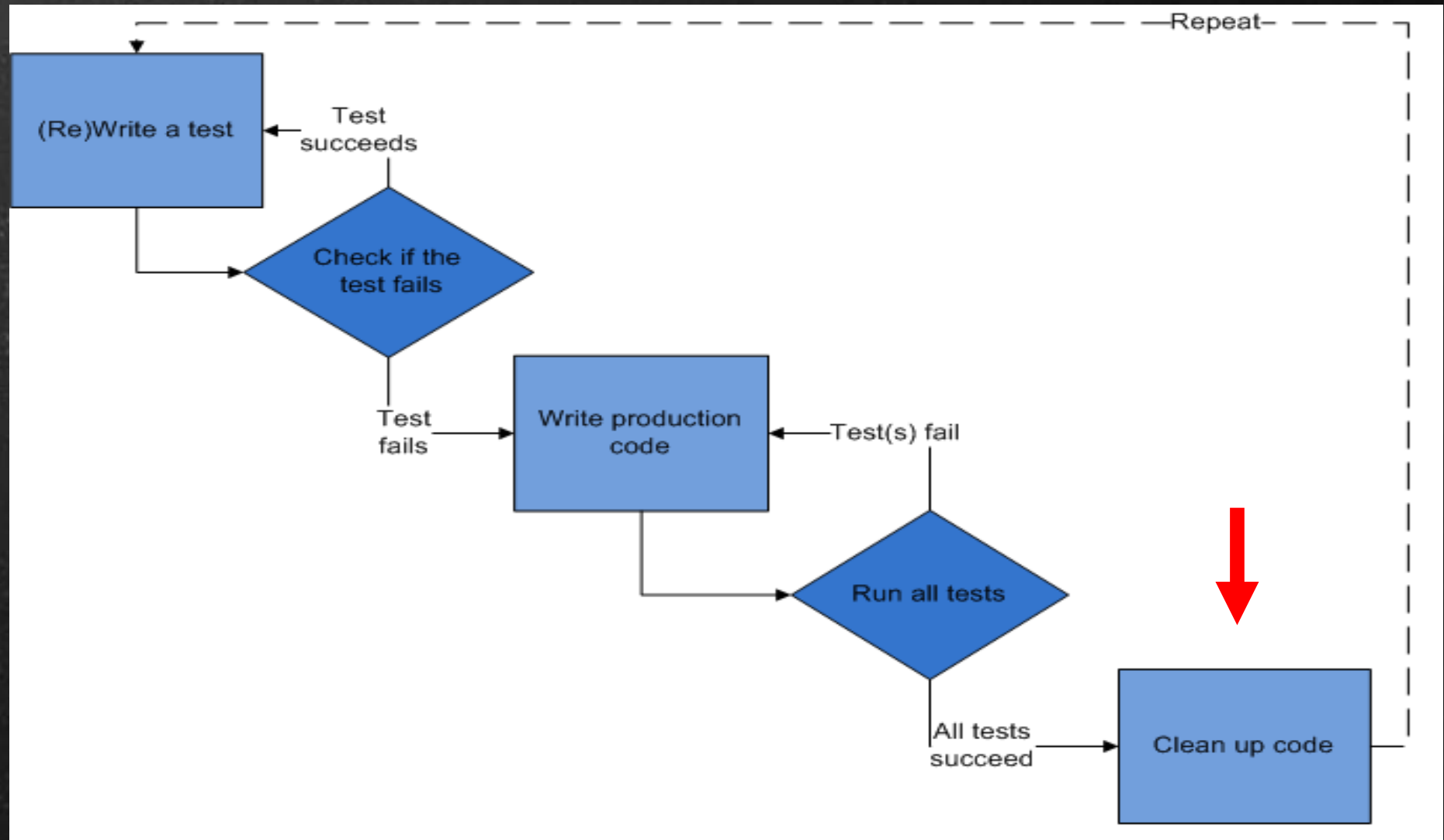
# TDD Cycle



3. Afterward, you can begin programming. Since there's already an automated test, as long as the code fails it, it means that it's still not ready. The code can be fixed until it passes all assertions.

- Writing some code that aim to pass the test only. Don't cover other function that is not a part of the test.

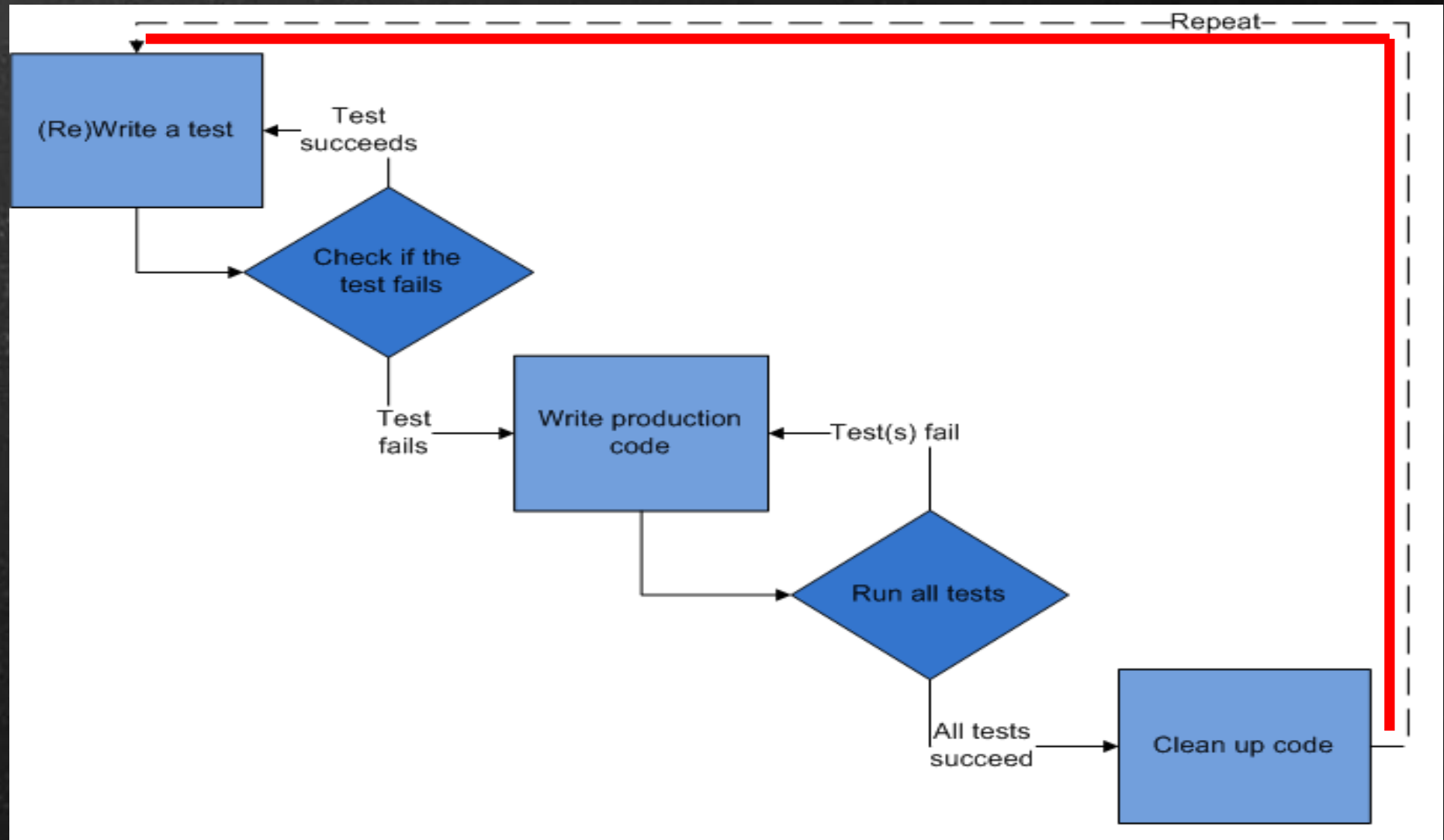
# TDD Cycle





4. Since the code passes the test, you can then begin cleaning it up, via refactoring. As long as the code still passes the test, it means that it still works. You no longer have to worry about changes that introduce new bugs.

# TDD Cycle



5. Start the whole thing over again with some other method or program.

# In Brief

1. The developer writes a failing automated test case that defines a desired improvement or new function.
2. Produces code to pass that test.
3. Re-factors the new code to acceptable standards.

# Benefit of TDD

1. TDD offers more than just simple validation of correctness, but can also drive the design of a program.

- They have to imagine how the functionality will be used by clients before implement it.
- This benefit is complementary to Design by Contract as it approaches code through test cases rather than through mathematical assertions or preconceptions.



2. TDD offers the ability to take small steps when required.

- It allows a programmer to focus on the task at hand as the first goal is to make the test pass. So, raising up the confidence in the code.

3. Total code implementation time is typically shorter.

- Large numbers of tests help to limit the number of defects in the code.

4. TDD can lead to more modularized, flexible, and extensible code.

- Because the methodology requires that the developers think of the software in terms of small units that can be written and tested independently and integrated together later. This leads to smaller, more focused classes, looser coupling, and cleaner interfaces.

5. Since TDD has Refactor Phase, the code is cleaned up and easier to be implemented by other developers.

# Discussion

*" TDD shorten the overall time but it increase the complexity for implementing the software project "*

*What do you think ? ..*



# What is Unit Testing?

In computer programming, unit testing is a method by which individual units of source code are tested to determine if they are fit for use. Unit tests are created by programmers or occasionally by white box testers during the development process.

# Goal of Unit Testing

The primary goal of unit testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and determine whether it behaves exactly as you expect.

# More about Unit Testing

Each unit is tested separately before integrating them into modules to test the interfaces between modules. Unit testing has proven its value in that a large percentage of defects are identified during its use.

# Case study without unit testing

For example, if you have two units and decide it would be more cost effective to glue them together and initially test them as an integrated unit, an error could occur in a variety of places:

- Is the error due to a defect in unit 1?
- Is the error due to a defect in unit 2?
- Is the error due to defects in both units?
- Is the error due to a defect in the interface between the units?
- Is the error due to a defect in the test?



Finding the error (or errors) in the integrated module is much more complicated than first isolating the units, testing each, then integrating them and testing the whole.



# Benefit of unit testing

## 1. Simplifies Integration

- Unit testing may reduce uncertainty in the units themselves and can be used in a bottom-up testing style approach.
- By testing the parts of a program first and then testing the sum of its parts, integration testing becomes much easier.

## 2. Facilitate Change

- Unit testing allows the programmer to refactor code at a later date, and make sure the module still works correctly (e.g., in regression testing).
- The procedure is to write test cases for all functions and methods so that whenever a change causes a fault, it can be quickly identified and fixed.

### **3. Design**

- an automatic test to test the internal workings of a class. It should be a stand-alone test which is not related to other resources

# Be Aware

## Separation of interface from implementation

- Because some classes may have references to other classes, testing a class can frequently spill over into testing another class.



## Simulate Situation : class that depend on a database

In order to test the class, the tester often writes code that interacts with the database. This is a mistake, because a unit test should usually not go outside of its own class boundary, and especially should not cross such process/network boundaries.

- Because this can introduce unacceptable performance problems to the unit test-suite.



- Crossing such unit boundaries turns unit tests into integration tests, and when test cases fail, makes it less clear which component is causing the failure.

Instead, the software developer should create an abstract interface around the database queries, and then implement that interface with their own **mock object**.

# What is Mock Object ?

- Mock Object is simulated object that mimic the behavior of real object in controlled ways.
- A programmer typically creates a mock object to test the behavior of some other object, in much the same way that a car designer uses a crash test dummy to simulate the dynamic behavior of a human in vehicle impacts.

By abstracting mock object, the independent unit can be more thoroughly tested than may have been previously achieved. This results in a higher quality unit that is also more maintainable.

Figure (A)

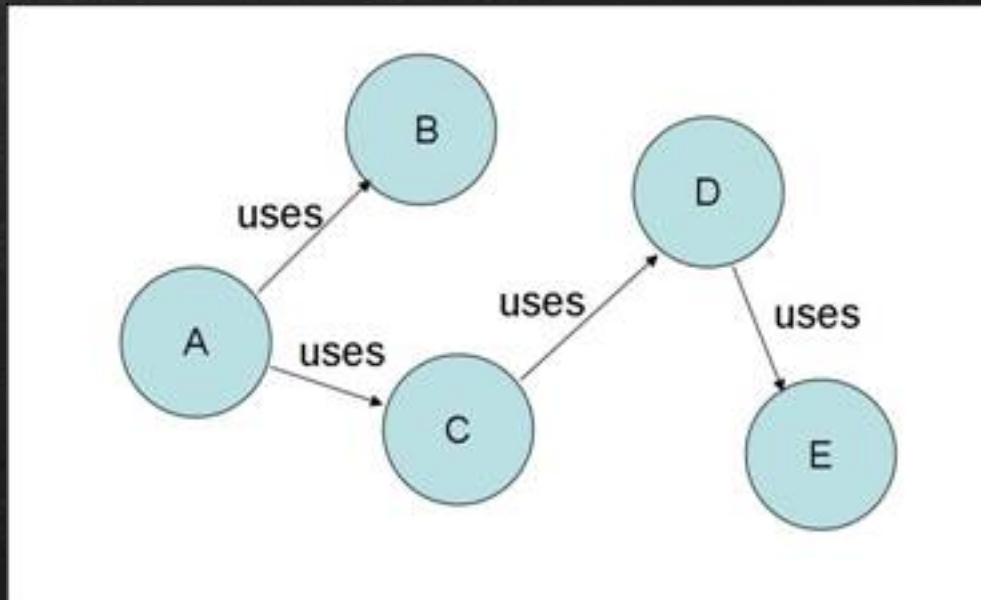
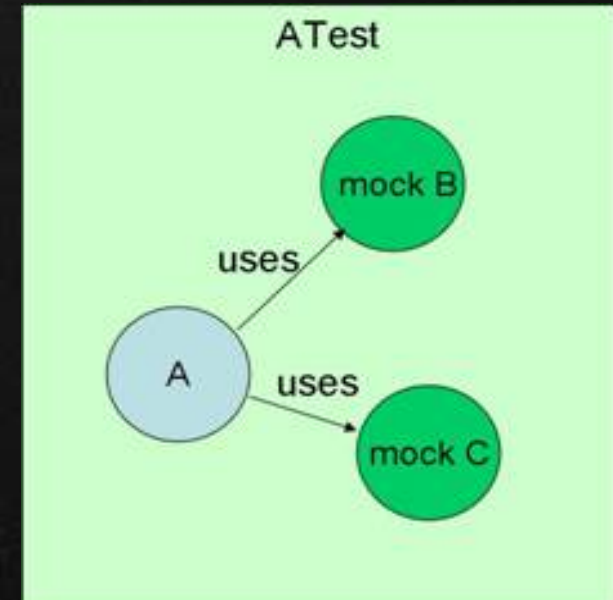


Figure (B)



# Example : Accepting input in java

Manual assign value to variable instead of assign value from Scanner (input reader class).



# Some available frameworks

LANGUAGE	FRAMEWORK
java	junit
c#	csUnit
python	pyunit
visual basic	vbUnit3



# Example

Create a calculator program with has only one function, adder, which calculate the addition between 2 numbers.

There are 2 cases for the result

- 1.show the correct summation answer if both number are valid input
  - 2.show statement "invalid input" whether 1st number or 2nd number is an invalid input
- \* valid number mean valid number format

# References

<http://net.tutsplus.com/tutorials/php/the-newbies-guide-to-test-driven-development>

[http://en.wikipedia.org/wiki/Test-driven\\_development](http://en.wikipedia.org/wiki/Test-driven_development)

[http://msdn.microsoft.com/en-us/library/aa292197\(v=vs.71\).aspx](http://msdn.microsoft.com/en-us/library/aa292197(v=vs.71).aspx)

[http://en.wikipedia.org/wiki/Mock\\_object](http://en.wikipedia.org/wiki/Mock_object)