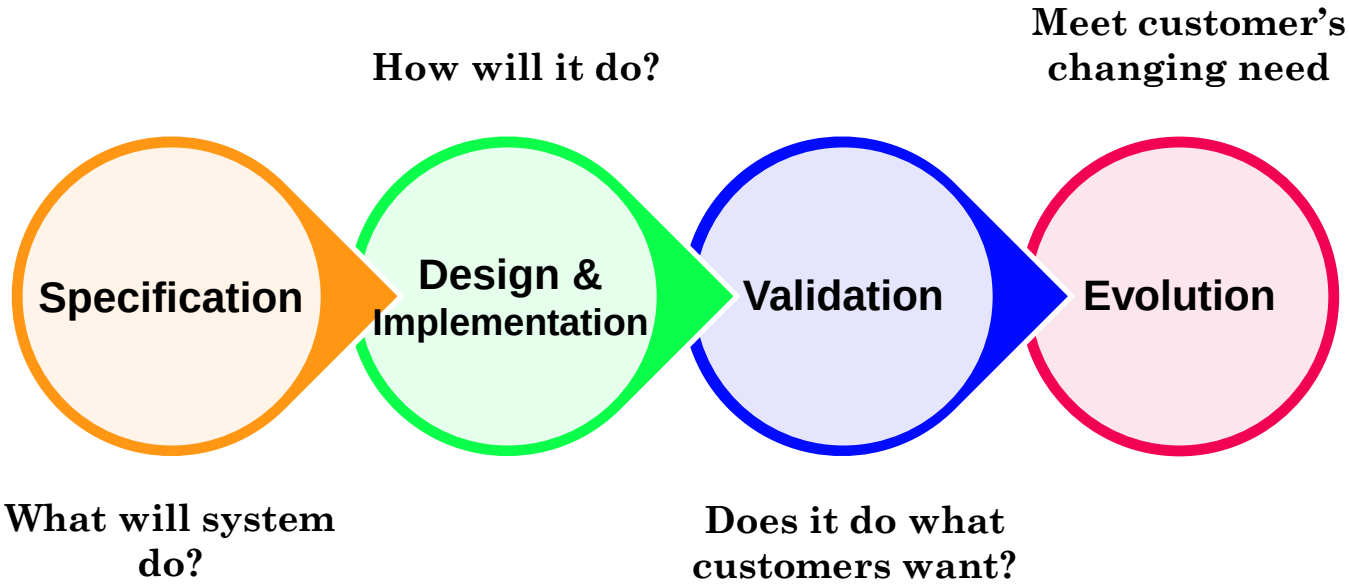
بسم الله الرحمن الرحيم

SE-200
Software Engineering

Dr. Qurat-ul-Ain

Recap



Software Process Activities

Last Week!

- Development Testing
  - Unit Testing
  - Component Testing
  - System Testing
- User Testing
  - Alpha Testing
  - Beta Testing
  - Acceptance Testing

This Week

- Release Testing
  - Requirement-based Testing
  - Scenario Testing
  - Performance Testing
    - Load Testing
    - Stress Testing
- Test-driven Development
  - Regression Testing

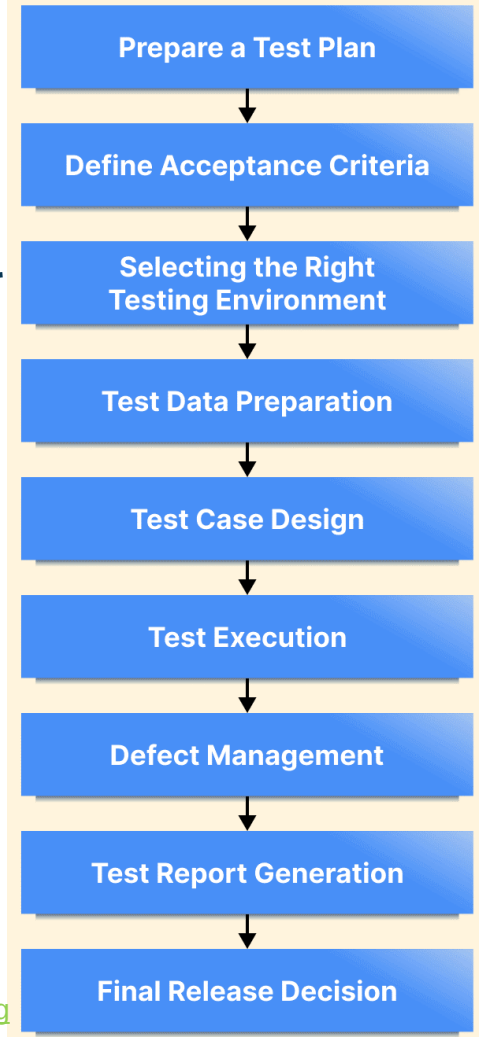
Release Testing

Release testing

- testing a particular release of a system that is intended for use outside of the development environment
- to convince the supplier of the system that it is good enough for use
  - system delivers its specified functionality, performance and dependability, and that it does not fail during normal use
- Usually a **black-box testing** process (i.e. functional)

where tests are only derived from the system specification

Database, automation tester
Manual tester, network protocol
Operating system



Test data generator
Valid test data, invalid test data
No data, boundary data

https://www.lambdatest.com/learning-hub/release-testing

Release testing VS System testing

- Release testing is a form of system testing
- Important differences:
  - System testing by the development team should focus on discovering **bugs** in the system (= *defect testing*)
  - The objective of release testing is to check that the system meets its **requirements** and is good enough for external use (= *validation testing*)
  - A **separate team** that has not been involved in the system development, should be responsible for release testing

Requirements based testing

- Requirements-based testing involves examining each requirement and developing a test or tests for it
  - Requirements should be testable.
  - Set of test cases can be defined for each requirement.
  - This is **validation testing**.

Requirements based testing

- MentCare system requirements:
  - If a patient is known to be allergic to any particular medication, then prescription of that medication shall result in a warning message being issued to the system user
  - If a prescriber chooses to ignore an allergy warning, they shall provide a reason why this has been ignored

Associated requirement tests (1/2)

- Set up a patient record with no known allergies. Prescribe medication for allergies that are known to exist. Check that a warning message is not issued by the system
- Set up a patient record with a known allergy. Prescribe the medication to that the patient is allergic to, and check that the warning is issued by the system.

Associated requirement tests (2/2)

- Set up a patient record in which allergies to two or more drugs are recorded. Prescribe both of these drugs separately and check that the correct warning for each drug is issued.
- Prescribe two drugs that the patient is allergic to. Check that two warnings are correctly issued.
- Prescribe a drug that issues a warning and overrule that warning. Check that the system requires the user to provide information explaining why the warning was overruled.

Scenario Testing

- A form of release testing whereby typical scenarios of use are used to develop test cases for the system
  - Set of test cases defined for each user story.
  - Testing sequence of requirements.
  - User story should be credible and fairly complex.
  - Evaluations of user story should be simple.
  - It should be validation testing.

Scenario Testing : Reading

George is a nurse who specializes in mental health care. One of his responsibilities is to visit patients at home to check that their treatment is effective and that they are not suffering from medication side effects.

On a day for home visits, George logs into the Mentcare system and uses it to print his schedule of home visits for that day, along with summary information about the patients to be visited. He requests that the records for these patients be downloaded to his laptop. He is prompted for his key phrase to encrypt the records on the laptop.

One of the patients whom he visits is Jim, who is being treated with medication for depression. Jim feels that the medication is helping him but believes that it has the side effect of keeping him awake at night. George looks up Jim's record and is prompted for his key phrase to decrypt the record. He checks the drug prescribed and queries its side effects. Sleeplessness is a known side effect, so he notes the problem in Jim's record and suggests that he visit the clinic to have his medication changed. Jim agrees, so George enters a prompt to call him when he gets back to the clinic to make an appointment with a physician. George ends the consultation, and the system re-encrypts Jim's record.

After finishing his consultations, George returns to the clinic and uploads the records of patients visited to the database. The system generates a call list for George of those patients whom he has to contact for follow-up information and make clinic appointments.

Scenario Testing : Example

George is a nurse who specializes in mental health care. One of his responsibilities is to visit patients at home to check that their treatment is effective and that they are not suffering from medication side effects.

On a day for home visits, George logs into the Mentcare system and uses it to print his schedule of home visits for that day, along with summary information about the patients to be visited. He requests that the records for these patients be downloaded to his laptop. He is prompted for his key phrase to encrypt the records on the laptop.

One of the patients whom he visits is Jim, who is being treated with medication for depression. Jim feels that the medication is helping him but believes that it has the side effect of keeping him awake at night. George looks up Jim's record and is prompted for his key phrase to decrypt the record. He checks the drug prescribed and queries its side effects. Sleeplessness is a known side effect, so he notes the problem in Jim's record and suggests that he visit the clinic to have his medication changed. Jim agrees, so George enters a prompt to call him when he gets back to the clinic to make an appointment with a physician. George ends the consultation, and the system re-encrypts Jim's record.

After finishing his consultations, George returns to the clinic and uploads the records of patients visited to the database. The system generates a call list for George of those patients whom he has to contact for follow-up information and make clinic appointments.

Features tested by scenario

- Authentication by logging on to the system.
- Downloading and uploading of specified patient records to a laptop.
- Home visit scheduling
- Encryption and decryption of patient records on a mobile device
- Record retrieval and modification
- Links with the drugs database that maintains side-effect information
- The system for call prompting

Performance testing

- Part of release testing may involve testing the emergent properties of a system, such as performance and reliability.
- Performance tests usually involve planning a series of tests where the load is steadily increased until the system performance becomes unacceptable.
- Combination of validation and defect testing.

Performance testing

- Tests should reflect the **usage ("operational") profile** of the system.

- Example: If 90% of the transactions in a system are of type A, 5% of type B, and the remainder of types C, D, and E, then you have to design the operational profile so that the vast majority of tests are of type A.

- These operational profiles are then testing with incremental load strategy.

Performance testing

- Stress testing is a form of performance testing where the system is deliberately overloaded to test its failure behavior.

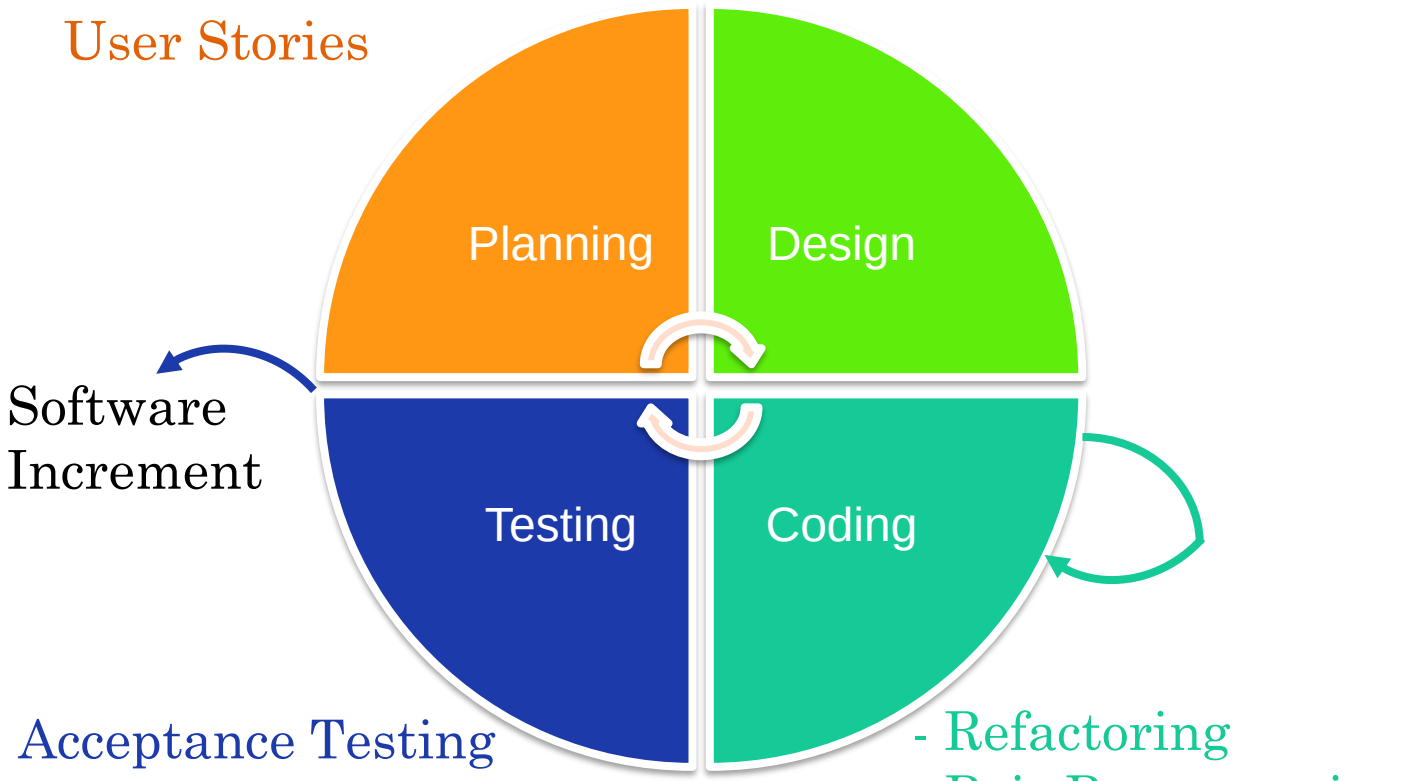
Test-driven Development

Test-driven Development (TDD)

- TDD was introduced in support of **agile methods** such as XP. However, it can also be used in **plan-driven** development process.

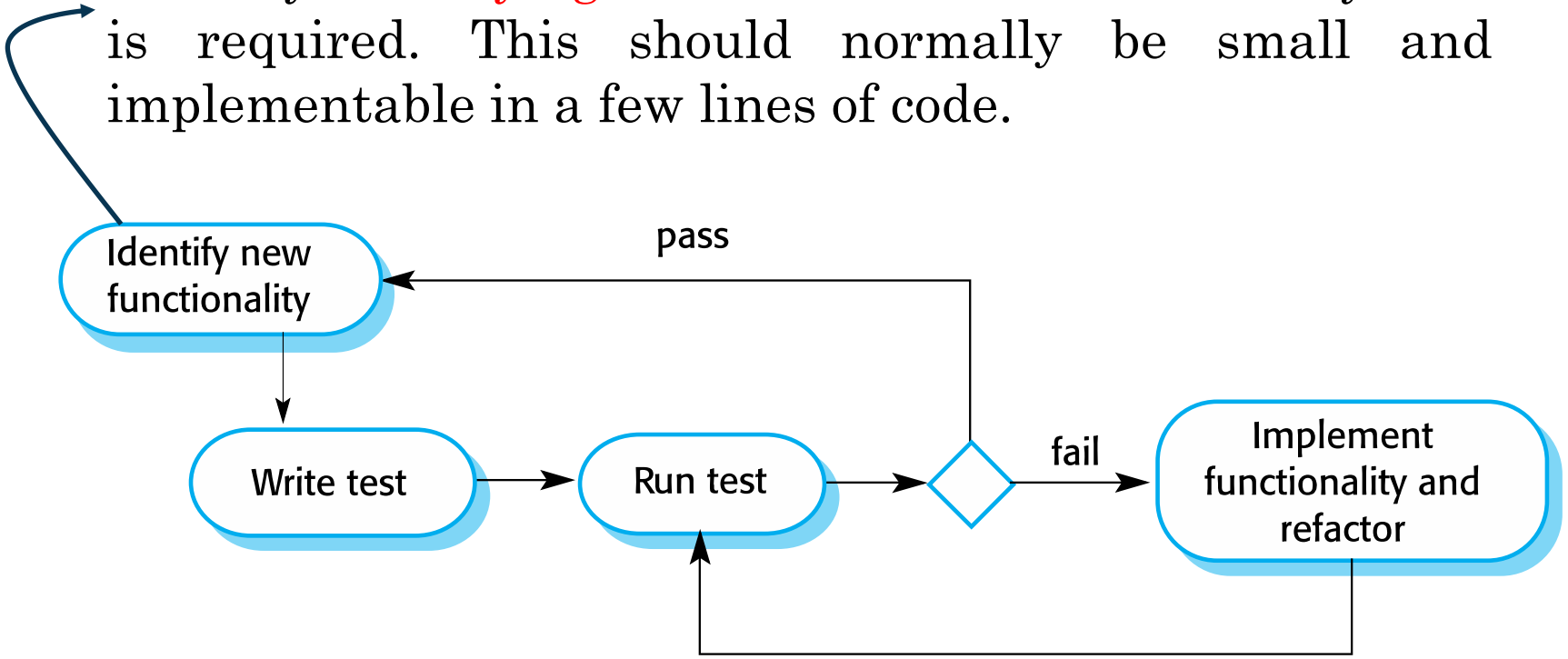
- Code is developed incrementally, along with a test for that increment. You don't move on to the next increment until the code that you have developed passes its test

Components of XP

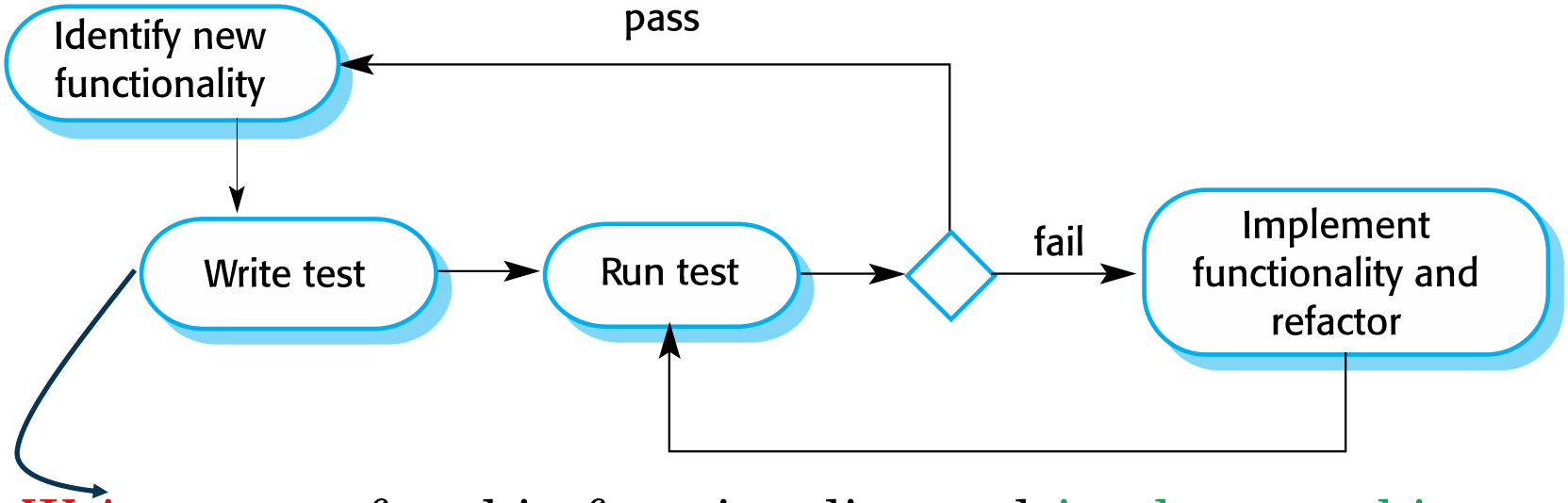


TDD process activities

Start by identifying the increment of functionality that is required. This should normally be small and implementable in a few lines of code.



TDD process activities

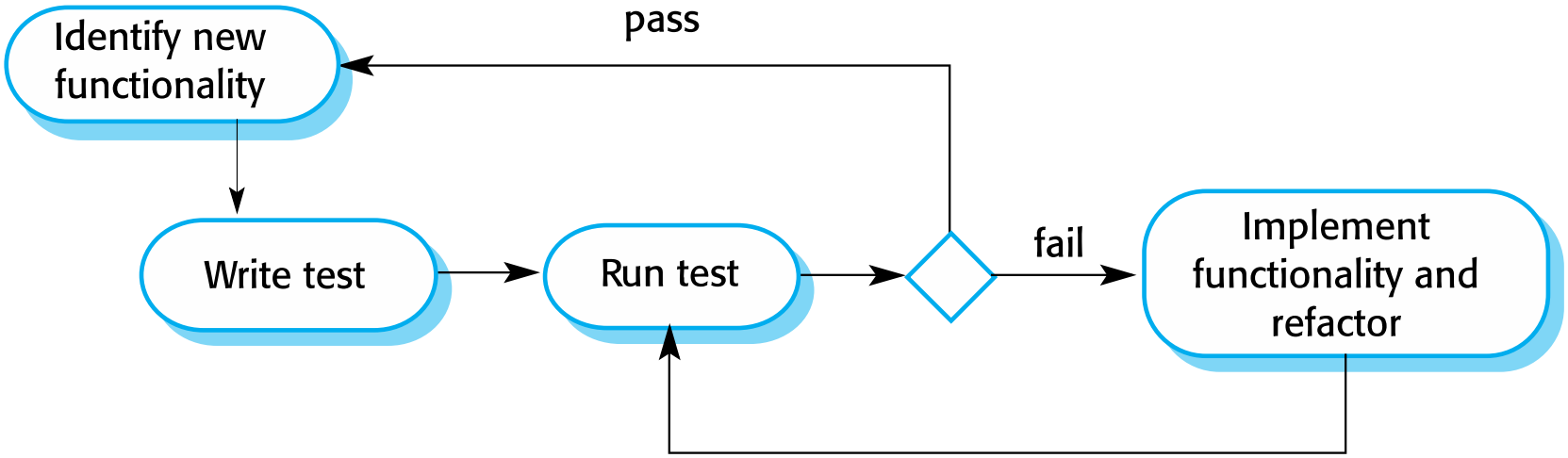


- Write a test for this functionality and implement this as an automated test.

TDD process activities

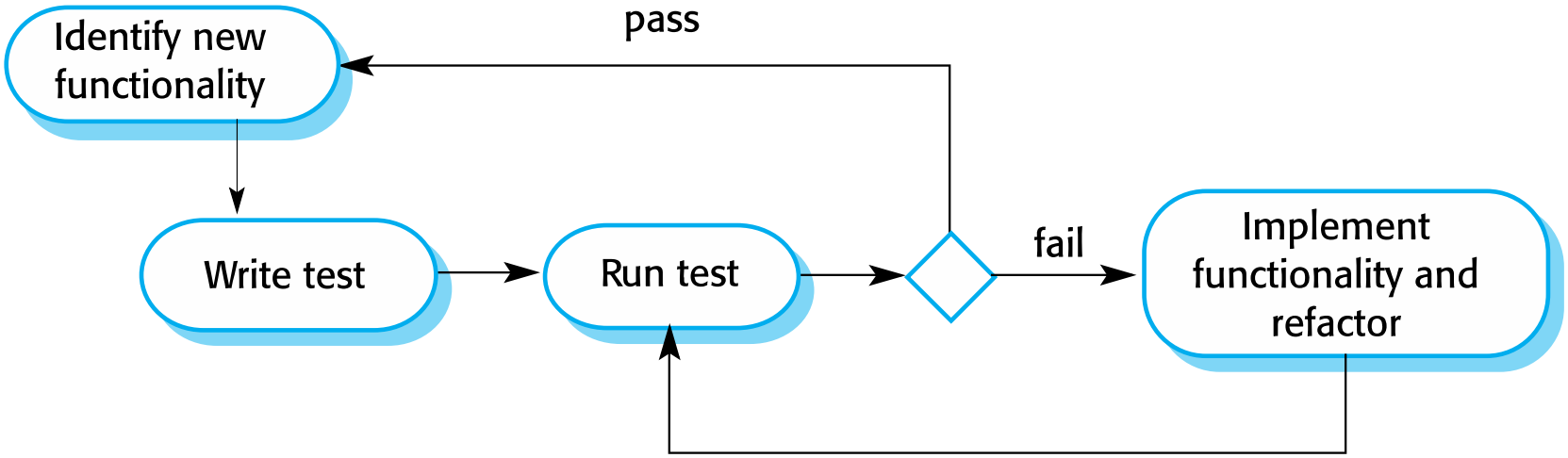
- Run the test, along with all other tests that have been implemented. Initially, you have not implemented the functionality so the new test will fail.

(Reinforces “test-first” mentality)



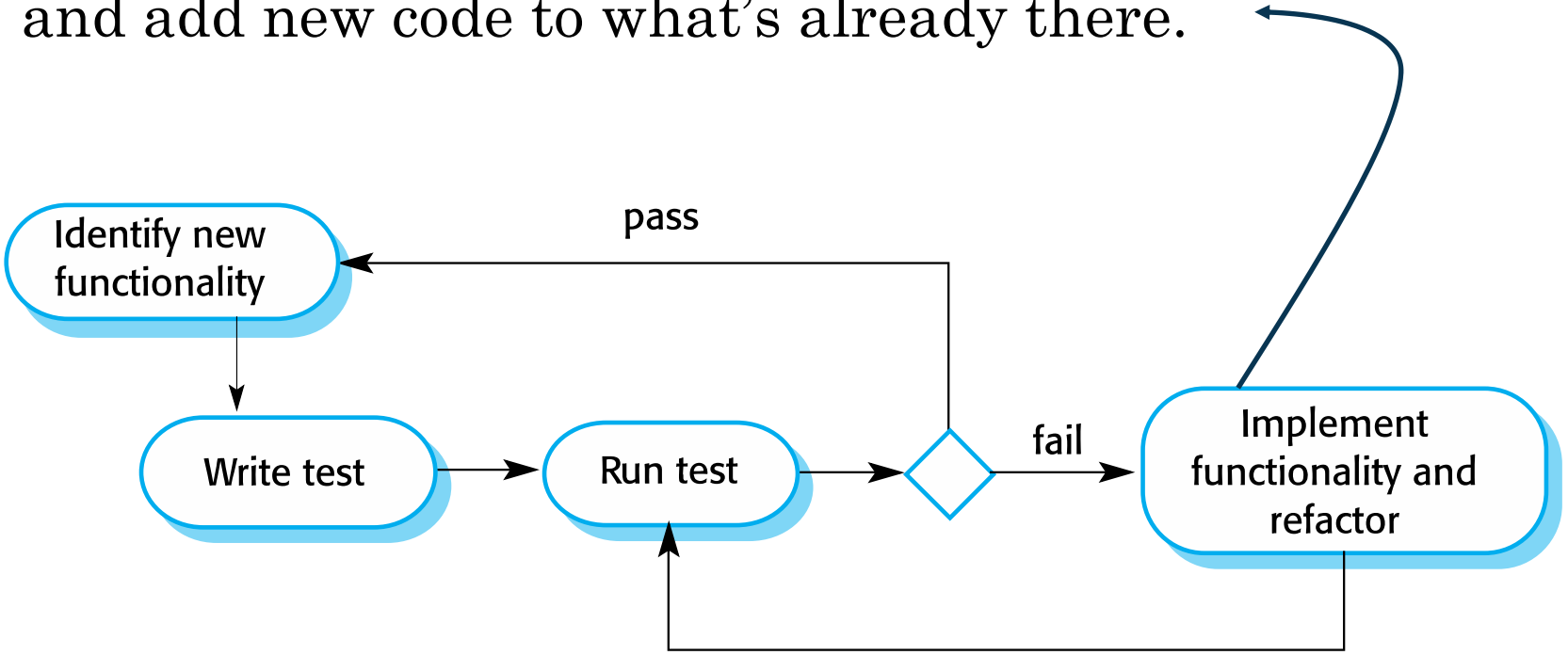
TDD process activities

- Implement the functionality and re-run the test(s)…
- Once all tests run successfully, you move on to implementing the next increment of functionality.



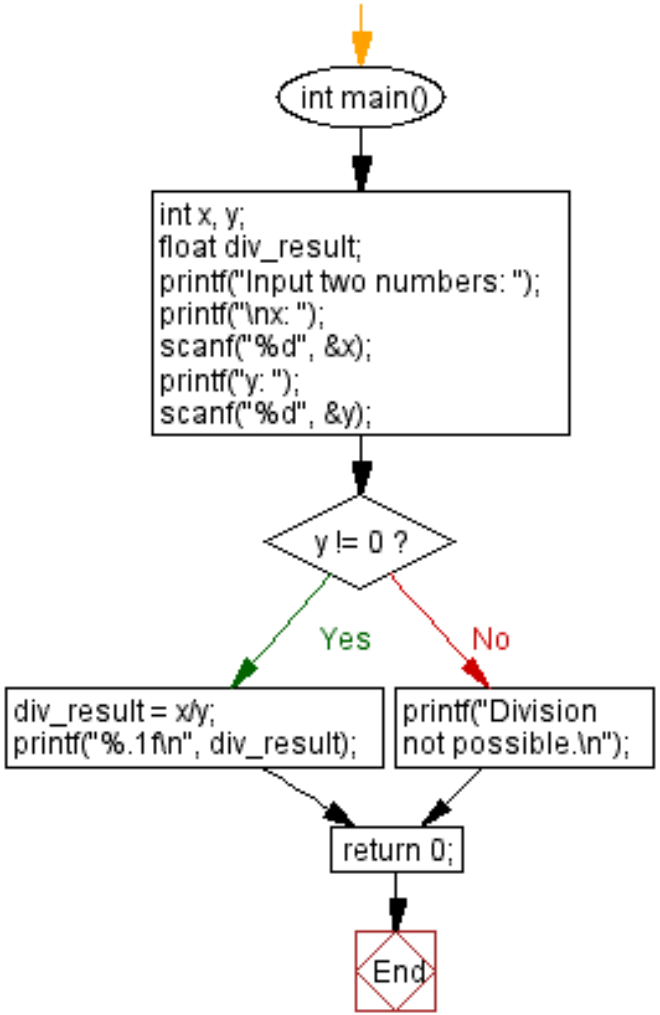
TDD process activities

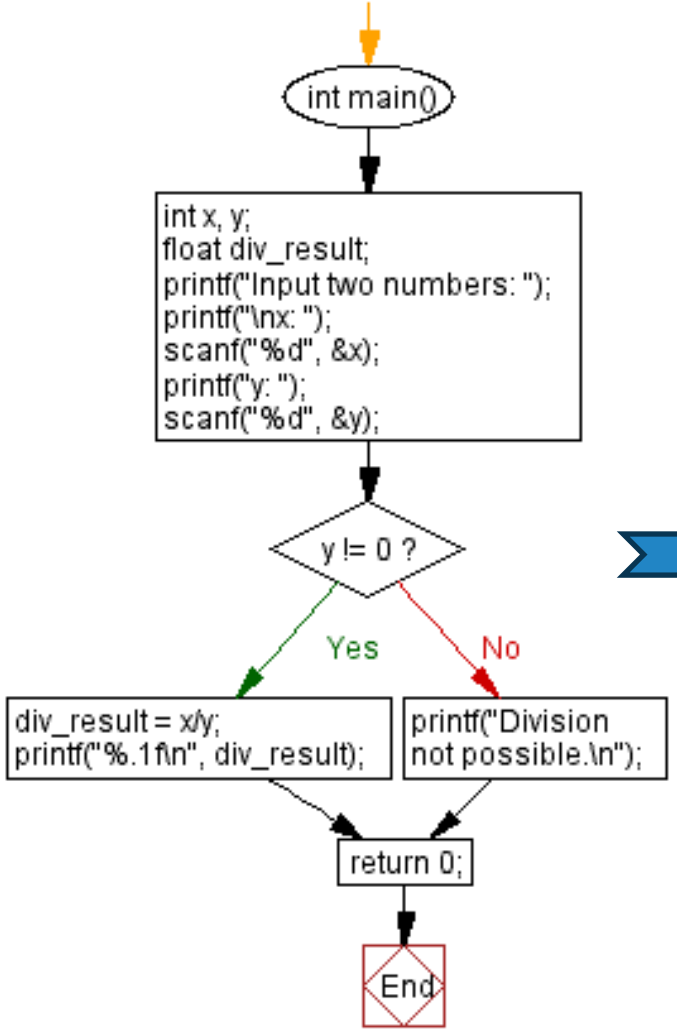
- This step involve **refactoring** existing code to improve it and add new code to what's already there.



Benefits of TDD

- better problem understanding
- If you don't know enough to write the tests, you won't develop the required code
- **Example: Division of two integers**






```c
int main() {
    int x, y;
    float div_result;

    // Prompt for user input
    printf("Input two numbers: ");
    printf("\nx: ");
    scanf("%d", &x);
    printf("y: ");
    scanf("%d", &y);

    // Check if division is possible
    if(y != 0) {
        div_result = x/y;
        printf("%.1f\n", div_result);
    } else {
        printf("Division not possible.\n");
    }

    return 0;
}
```

Benefits of TDD

- **Code coverage** : Every code segment that you write has at least one associated test so all code written has at least one test.
- **Regression testing** : A regression test suite is developed incrementally as a program is developed.
- **Simplified debugging** : When a test fails, it should be obvious where the problem lies. The newly written code needs to be checked and modified.
- **System documentation** : The tests themselves are a form of documentation that describe what the code should be doing.

Benefits of TDD

"When you test first, you capture your intent in an automatable and executable form. You focus on what you are about to write in a way that works to prevent defects rather than create them. The tests you write serve as a persistent reinforcement of that intent going forward. In addition to helping you do the *thing right,* it helps you to do the *right thing.*"

— Stephen Vance, in Quality Code, 2014

Test-first Development - Limitations

- Programmers prefer programming to testing and sometimes they take short cuts when writing tests
- tests can be very difficult to write incrementally.
- difficult to judge the completeness of a set of tests.

Regression Testing

- Re-running of one or more test cases, after some program change, that ran without revealing faults prior to the change
- It is used to determine if the changes have not 'broken' previously working code
- This can be expensive in a manual testing environment, but automated testing, it may be possible to run **all** previous (together with new) tests with each increment
- Tests must run 'successfully' before the change is committed

Applications of TDD

- Software with new codes
- Functions from standard libraries

- Reuse of legacy system
- Multi threaded system

Thanks!

Any comment/questions?