Telerik Academy

Test-Driven Development (TDD)

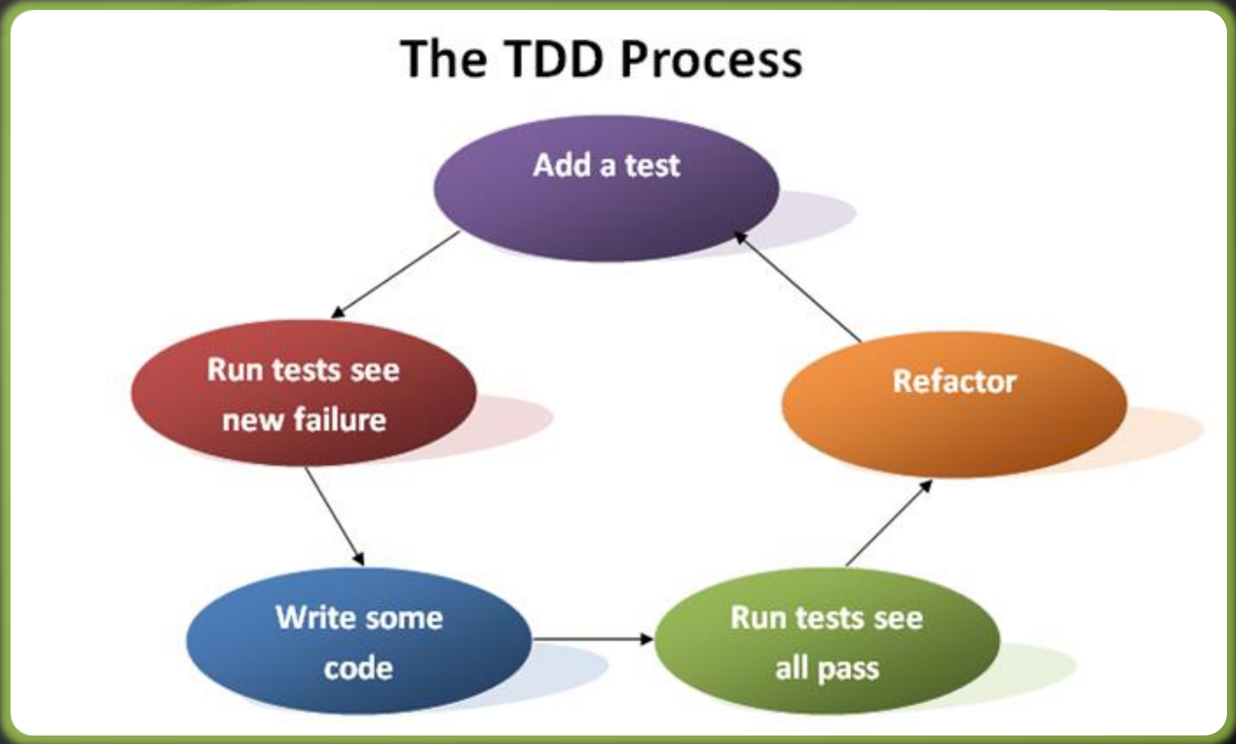
Learn the "Test First" Approach to Coding

.Net Ninja
Red
Refactor!
Green

Telerik Software Academy

Learning & Development

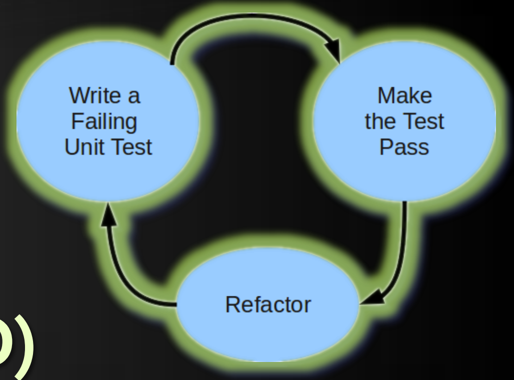
http://academy.telerik.com

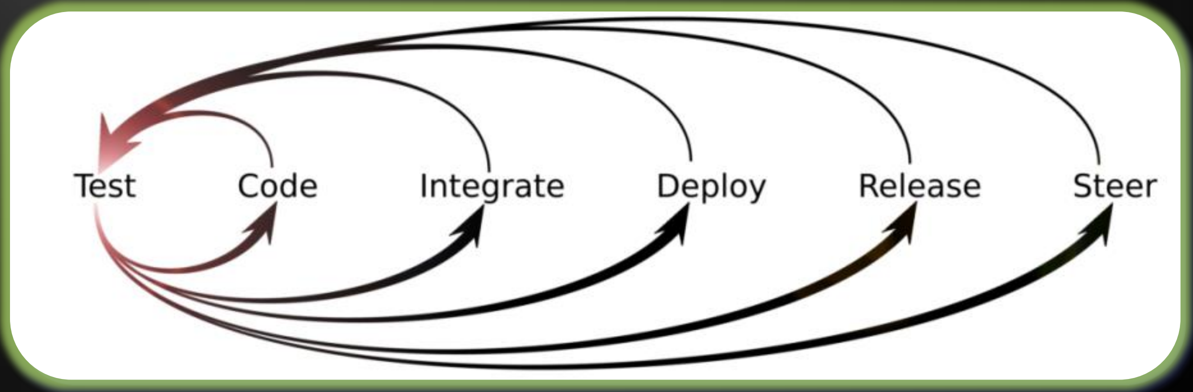


Code and Test vs. Test Driven Development

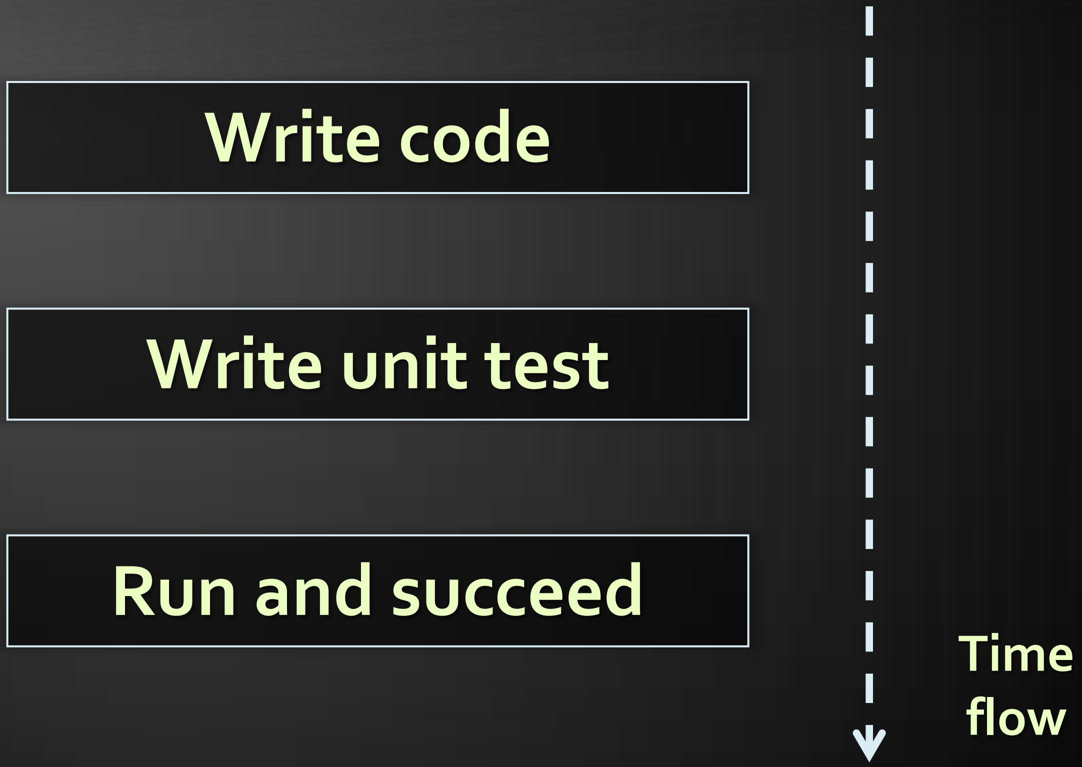
Unit Testing Approaches

- "Code First" (code and test) approach
  - Classical approach
- "Test First" approach
  - Test-driven development (TDD)





Code and Test Approach



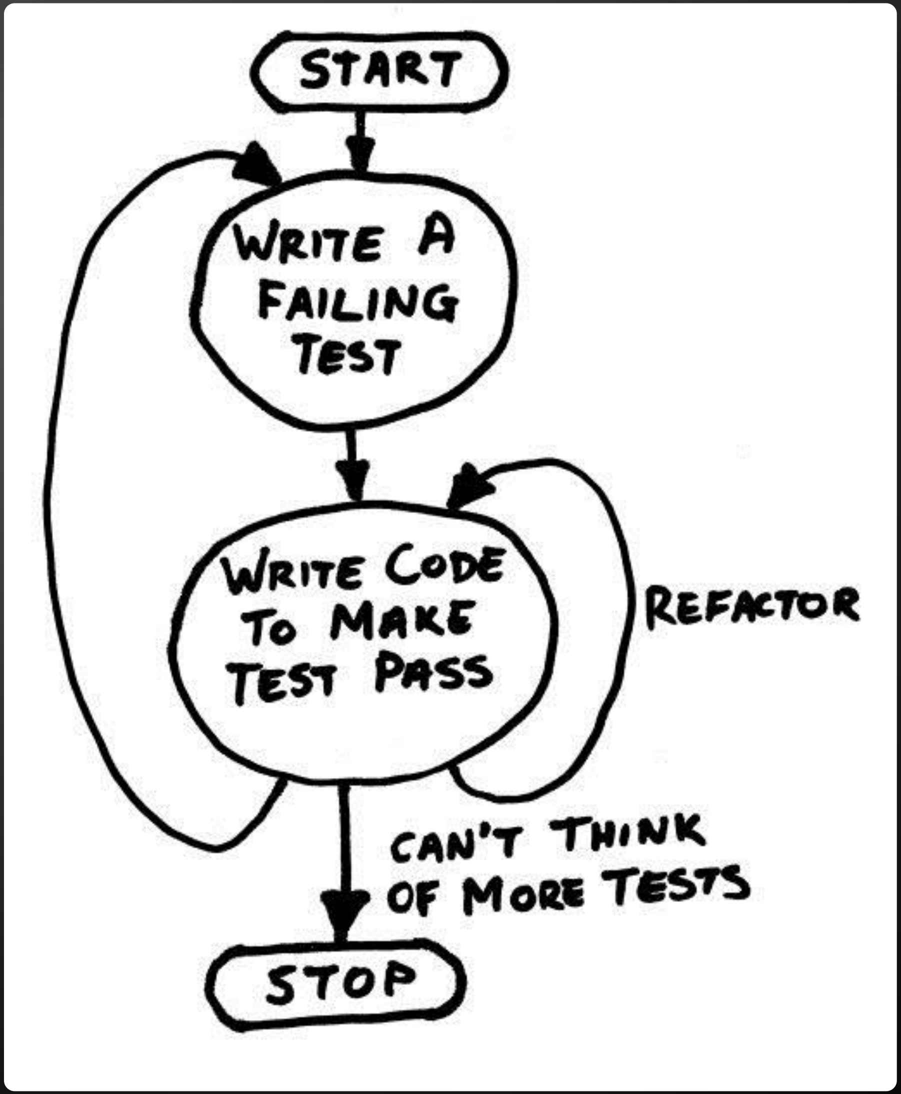
Write code

Write unit test

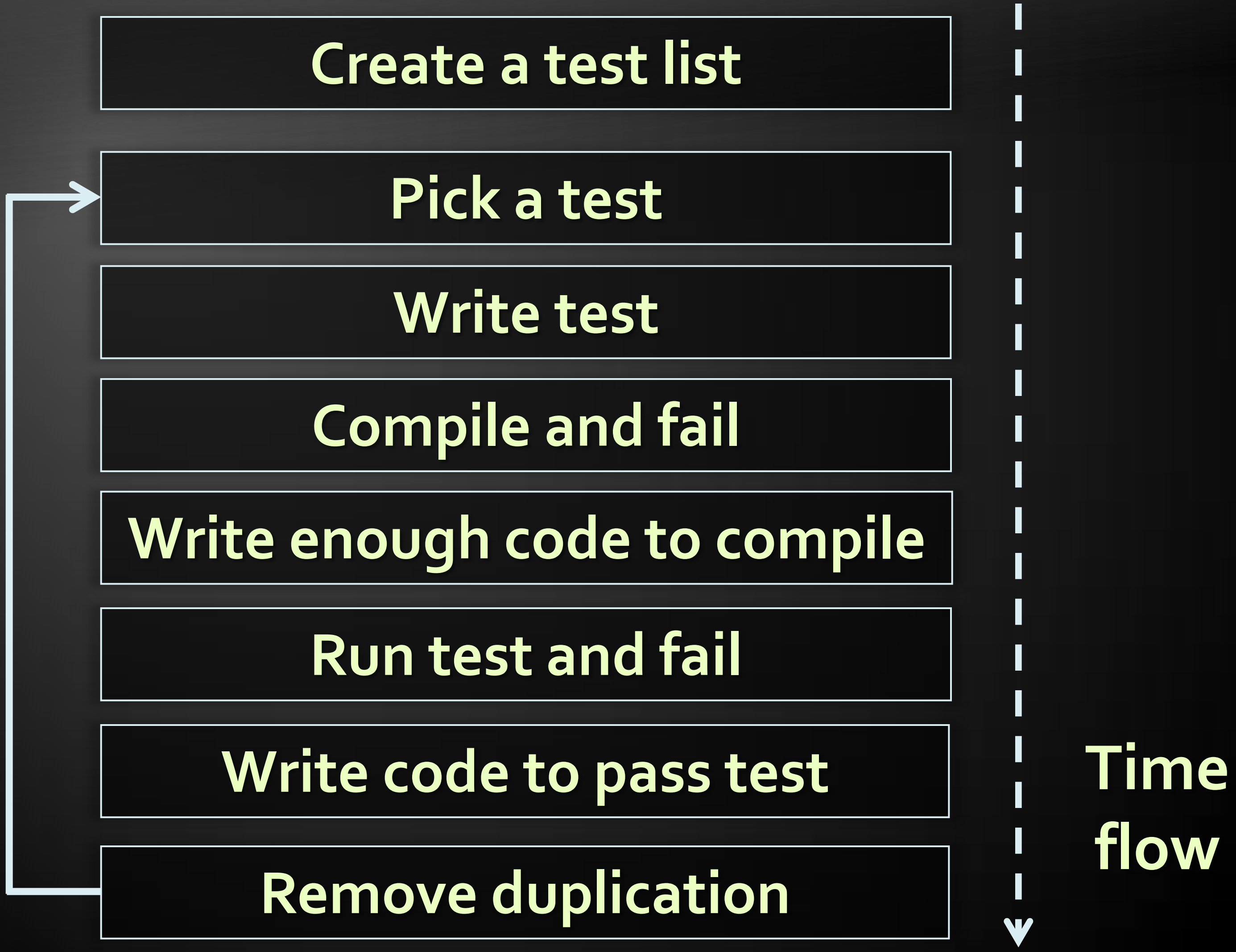
Run and succeed

Time flow

TDD in One Slide



Test-Driven Development (TDD)



Create a test list

Pick a test

Write test

Compile and fail

Write enough code to compile

Run test and fail

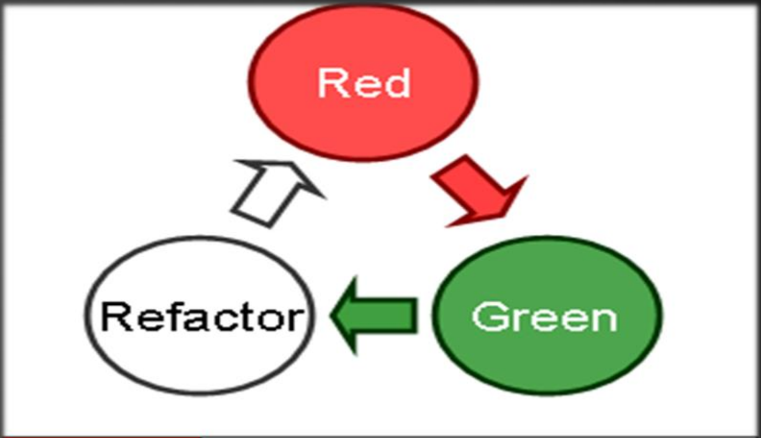
Write code to pass test

Remove duplication

Time flow

Why TDD?

- TDD helps find design issues early
  - Avoids rework
- Writing code to satisfy a test is a focused activity
  - Less chance of error
- Tests will be more comprehensive than when written after code



Red	Write a test that fails.
Green	Write code to pass the test.
Refactor	Clean up your code and the test.

Test-Driven Development

Live Demo: Poker Hands Checker

Telerik Academy

Test-Driven Development (TDD)

Questions?

http://academy.telerik.com

Homework

- Finish the "Poker" project given in the Visual Studio Solution "11. Test-Driven-Development-Demo+Homework.zip" using TDD.

1. Write a class `Card` implementing the `ICard` interface. Implement the properties. Write a constructor. Implement the `ToString()` method. Test all cases.
2. Write a class `Hand` implementing the `IHand` interface. Implement the properties. Write a constructor. Implement the `ToString()` method. Test all cases.
3. Write a class `PokerHandsChecker` (+ tests) and start implementing the `IPokerHandsChecker` interface. Implement the `IsValidHand(IHand)`. A hand is valid when it consists of exactly 5 different cards.

Homework (2)

4. Implement `IPokerHandsChecker.IsFlush(IHand)` method. Follow the official poker rules from Wikipedia: http://en.wikipedia.org/wiki/List_of_poker_hands
5. Implement `IsFourOfAKind(IHand)` method. Did you test all the scenarios?
6. * Implement the other check for poker hands: `IsHighCard(IHand hand)`, `IsOnePair(IHand hand)`, `IsTwoPair(IHand hand)`, `IsThreeOfAKind(IHand hand)`, `IsFullHouse(IHand hand)`, `IsStraight(IHand hand)` and `IsStraightFlush(IHand hand)`. Did you test all the scenarios well?
7. * Implement a card comparison logic for Poker hands (+ tests). `CompareHands(…)` should return `-1`, `0` or `1`.

Free Trainings @ Telerik Academy

- C# Programming @ Telerik Academy
  - csharpfundamentals.telerik.com

- Telerik Software Academy
  - academy.telerik.com

- Telerik Academy @ Facebook
  - facebook.com/TelerikAcademy

- Telerik Software Academy Forums
  - forums.academy.telerik.com