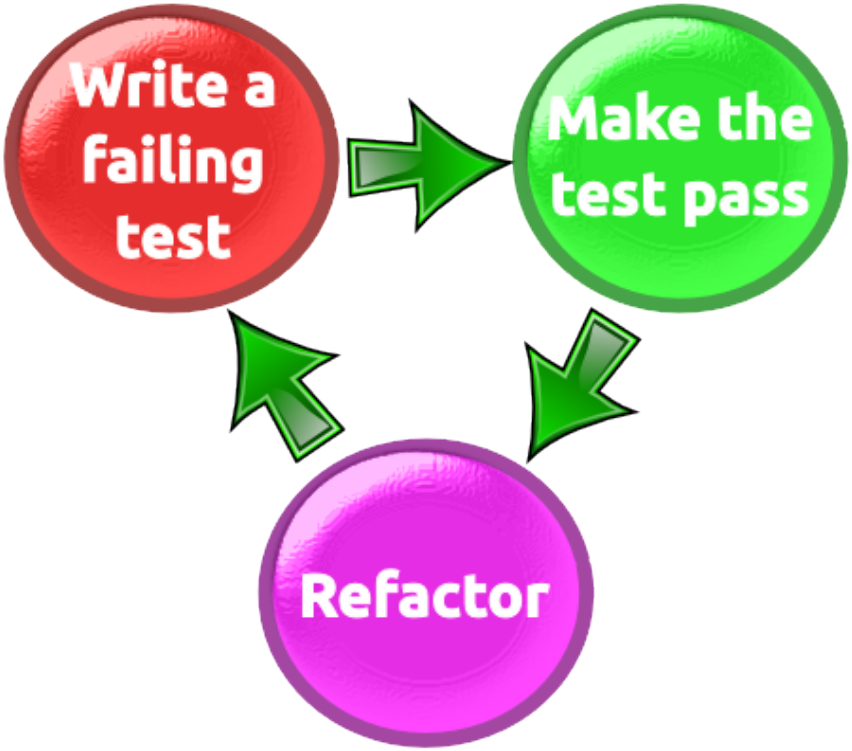
# TDD Kata

Jun Chen

February 21, 2014

# TDD Cycle: Red-Green-Refactor




- Write a test ("red")
- Get the test to pass ("green")
- Optimize the design ("refactor")

Each test should represent the smallest meaningful increment you can think of.

# Three Rules of TDD

# Three Rules of TDD

1. Write NO production code except to pass a failing test.

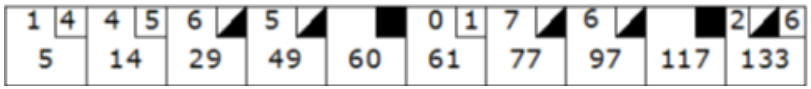
# Three Rules of TDD

1. Write NO production code except to pass a failing test.
2. Write only enough of a test to demonstrate a failure.

# Three Rules of TDD

1. Write NO production code except to pass a failing test.
2. Write only enough of a test to demonstrate a failure.
3. Write only enough production code to pass the test.

# Scoring Bowling




The game consists of 10 frames as shown above. In each frame the player has two opportunities to knock down 10 pins. The score for the frame is the total number of pins knocked down, plus bonuses for strikes and spares.

1. A spare is when the player knocks down all 10 pins in two tries. The bonus for that frame is the number of pins knocked down by the next roll. So in frame 3 above, the score is 10 (the total number knocked down) plus a bonus of 5 (the number of pins knocked down on the next roll.)
2. A strike is when the player knocks down all 10 pins on his first try. The bonus for that frame is the value of the next two balls rolled.
3. In the tenth frame a player who rolls a spare or strike is allowed to roll the extra balls to complete the frame. However no more than three balls can be rolled in tenth frame.

# Requirement

Write a class named Game that has two methods

1. roll(pins : int) is called each time the player rolls a ball. The argument is the number of pins knocked down.
2. score() : int is called only at the very end of the game. It returns the total score for that game.