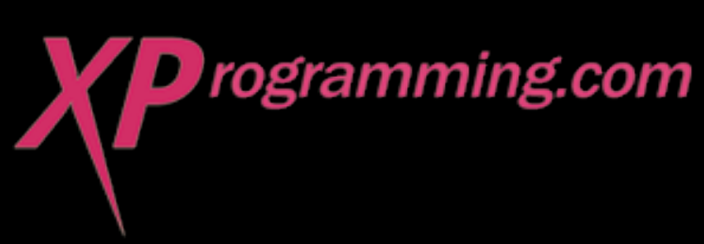
Bowling Game Kata

Object Mentor, Inc.

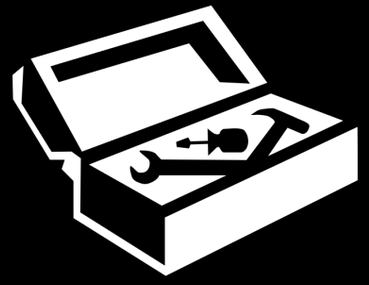
www.objectmentor.com
blog.objectmentor.com

fitnesse.org



www.junit.org

…adapted for Objective-C

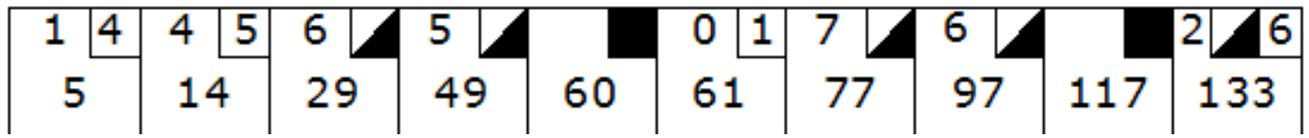


Quality Coding *with Jon Reid*

QualityCoding.org

Last update: 31 Jan 2016

Scoring Bowling



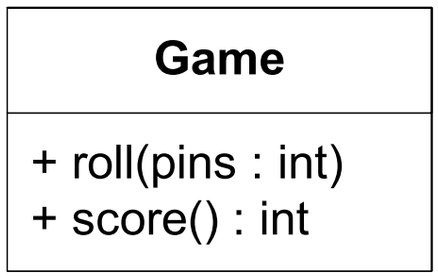
The game consists of 10 frames as shown above. In each frame the player has two opportunities to knock down 10 pins. The score for the frame is the total number of pins knocked down, plus bonuses for strikes and spares.

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.)

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.

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.

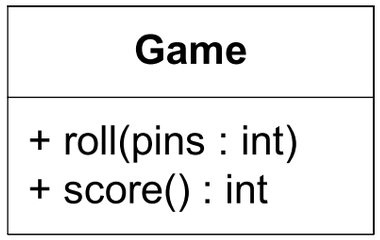
The Requirements



Write a class named “Game” that has two methods:

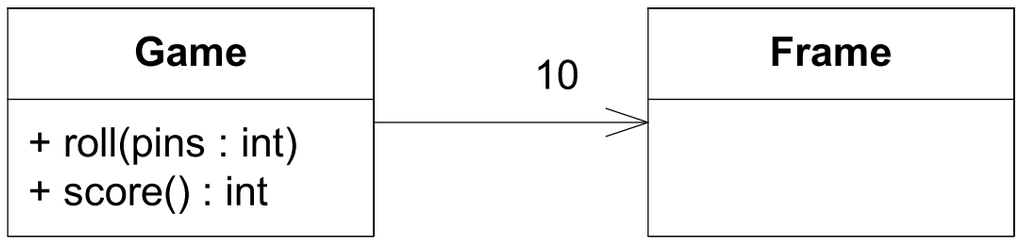
- (void)rollWithPinCount:(NSUInteger)pins is called each time the player rolls a ball. The argument is the number of pins knocked down.
- (NSUInteger)score is called only at the very end of the game. It returns the total score for that game.

A quick design session



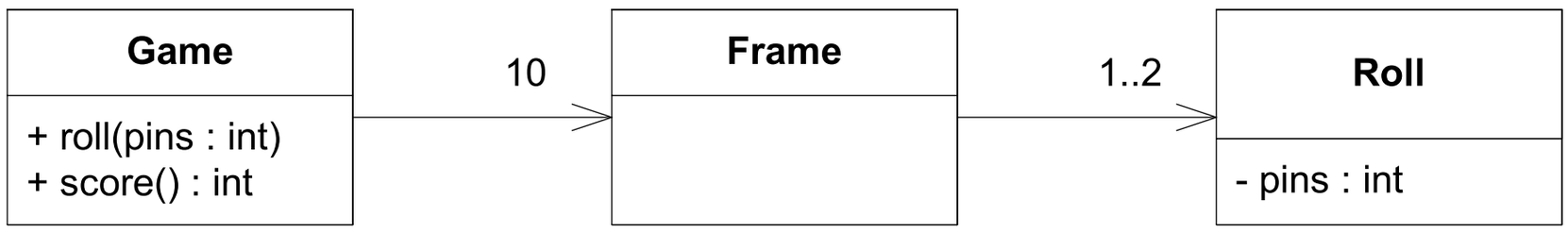
Clearly we need the Game class.

A quick design session



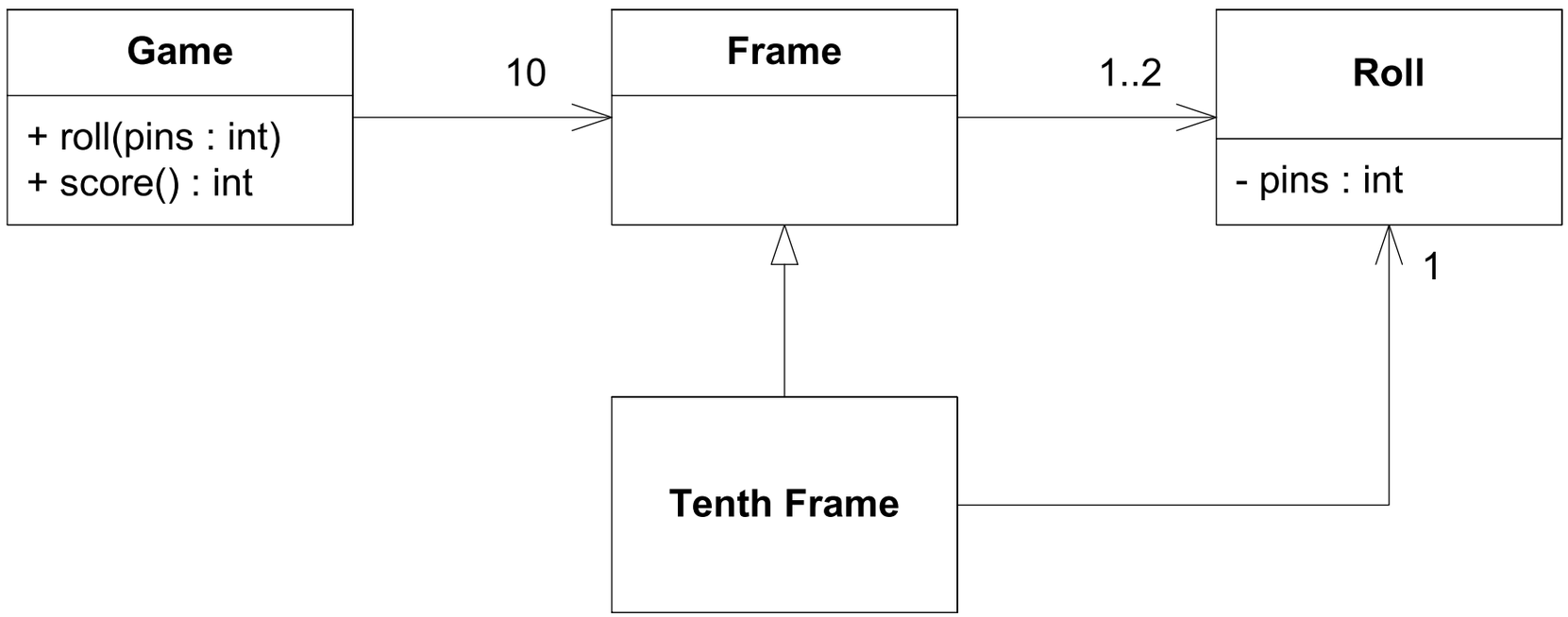
A game has 10 frames.

A quick design session



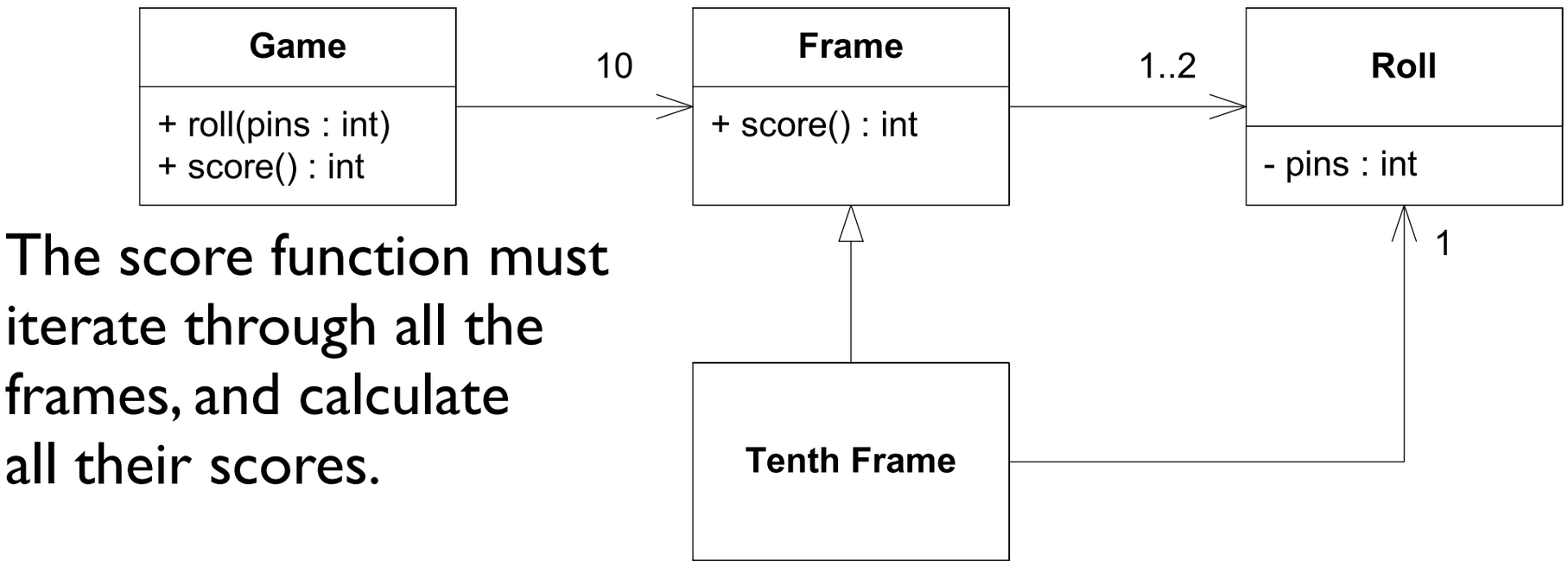
A frame has 1 or 2 rolls.

A quick design session



The tenth frame has 2 or 3 rolls.
It is different from all the other frames.

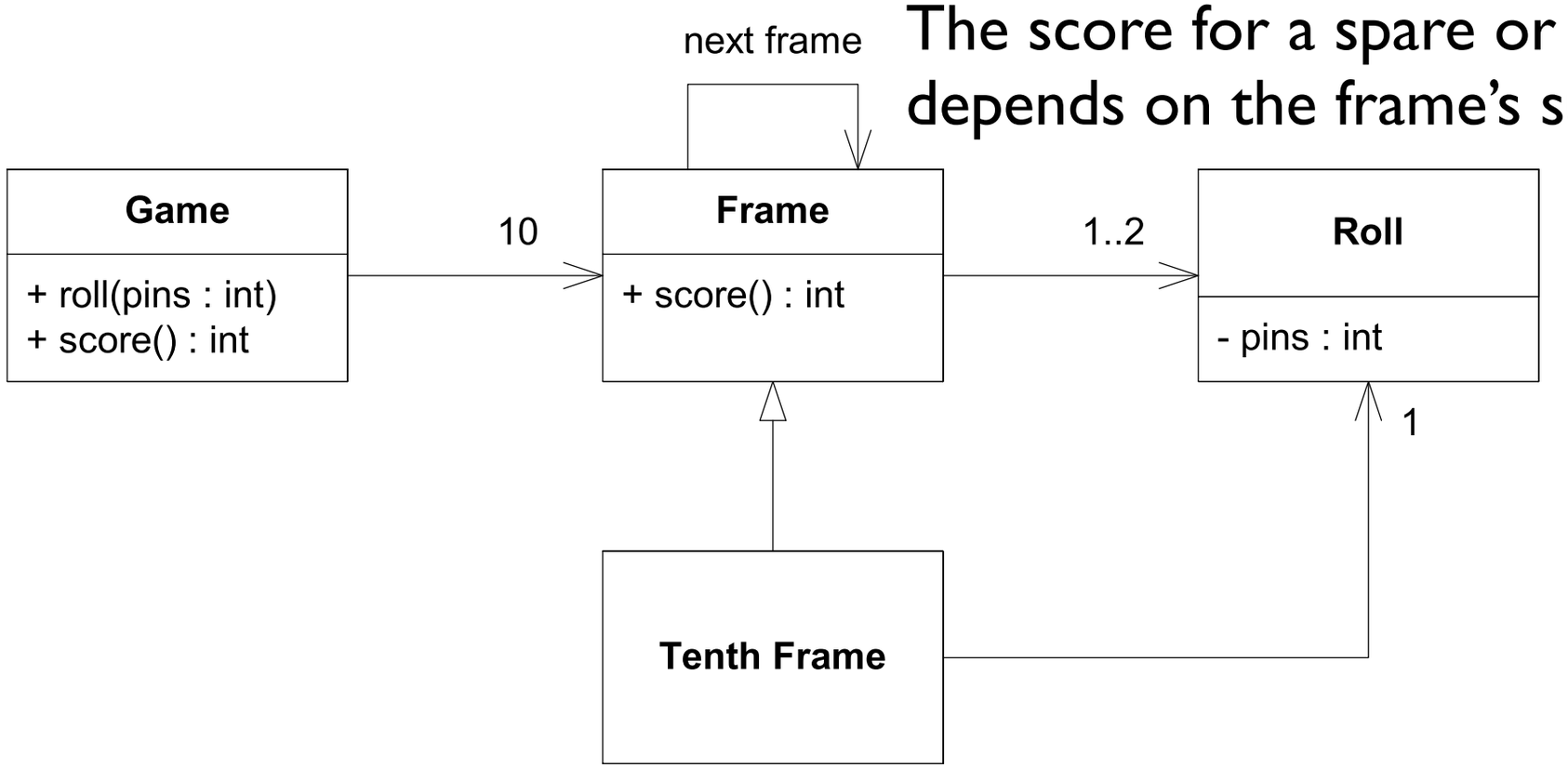
A quick design session



The score function must iterate through all the frames, and calculate all their scores.

A quick design session

The score for a spare or a strike depends on the frame's successor



Begin.

- Create an iOS application of type “Single View Application”. Name it *BowlingGame*. Select Objective-C as the language. Select “Include Unit Tests”
- Select any iOS Simulator in Scheme picker
- ⌘U to run unit tests
- Verify that testExample successfully ran
- Delete setUp, tearDown, testExample and testPerformanceExample, and run tests again

The first test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests

- (void)testGutterGame {
    BowlingGame *game = [[BowlingGame alloc] init];
}

@end
```

```objc
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame
@end
```

The first test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests

- (void)testGutterGame {
    BowlingGame *game = [[BowlingGame alloc] init];
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:0];
}

@end
```

```objc
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame
@end
```

The first test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests

- (void)testGutterGame {
    BowlingGame *game = [[BowlingGame alloc] init];
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:0];
}

@end
```

```objc
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame

- (void)rollWithPinCount:(NSUInteger)pins {
}

@end
```

The first test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests

- (void)testGutterGame {
    BowlingGame *game = [[BowlingGame alloc] init];
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:0];
    XCTAssertEqual([game score], 0);
}

@end
```

```objc
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame

- (void)rollWithPinCount:(NSUInteger)pins {
}

- (NSUInteger)score {
    return 9999;
}

@end
```

("9999") is not equal to ("0")

The first test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests

- (void)testGutterGame {
    BowlingGame *game = [[BowlingGame alloc] init];
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:0];
    XCTAssertEqual([game score], 0);
}

@end
```

```objc
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame

- (void)rollWithPinCount:(NSUInteger)pins {
}

- (NSUInteger)score {
    return 0;
}

@end
```

The second test.

```
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests

- (void)testGutterGame {
    BowlingGame *game = [[BowlingGame alloc] init];
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:0];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    BowlingGame *game = [[BowlingGame alloc] init];
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:1];
    XCTAssertEqual([game score], 20);
}

@end
```

```
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame

- (void)rollWithPinCount:(NSUInteger)pins {
}

- (NSUInteger)score {
    return 0;
}

@end
```

- Game creation is duplicated
- roll loop is duplicated

The second test.

```
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests

- (void)testGutterGame {
    BowlingGame *game = [[BowlingGame alloc] init];
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:0];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    BowlingGame *game = [[BowlingGame alloc] init];
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:1];
    XCTAssertEqual([game score], 20);
}

@end
```

```
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame

- (void)rollWithPinCount:(NSUInteger)pins {
}

- (NSUInteger)score {
    return 0;
}

@end
```

- Game creation is duplicated
- roll loop is duplicated

The second test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests

- (void)testGutterGame {
    BowlingGame *game = [[BowlingGame alloc] init];
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:0];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    BowlingGame *game = [[BowlingGame alloc] init];
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:1];
    XCTAssertEqual([game score], 20);
}

@end
```

```objc
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame

- (void)rollWithPinCount:(NSUInteger)pins {
}

- (NSUInteger)score {
    return 0;
}

@end
```

("0") is not equal to ("20")

- Game creation is duplicated
- roll loop is duplicated

The second test.

```
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests

- (void)testGutterGame {
    BowlingGame *game = [[BowlingGame alloc] init];
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:0];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    BowlingGame *game = [[BowlingGame alloc] init];
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:1];
    XCTAssertEqual([game score], 20);
}

@end
```

```
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _score;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _score += pins;
}

- (NSUInteger)score {
    return _score;
}

@end
```

- Game creation is duplicated
- roll loop is duplicated

The second test.

```
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests

- (void)testGutterGame {
    BowlingGame *game = [[BowlingGame alloc] init];
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:0];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    BowlingGame *game = [[BowlingGame alloc] init];
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:1];
    XCTAssertEqual([game score], 20);
}

@end
```

```
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _score;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _score += pins;
}

- (NSUInteger)score {
    return _score;
}

@end
```

- roll loop is duplicated

The second test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

- (void)setUp {
    [super setUp];
    game = [[BowlingGame alloc] init];
}

- (void)tearDown {
    game = nil;
    [super tearDown];
}

- (void)testGutterGame {
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:0];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:1];
    XCTAssertEqual([game score], 20);
}

@end
```

```objc
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _score;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _score += pins;
}

- (NSUInteger)score {
    return _score;
}

@end
```

- roll loop is duplicated

The second test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

- (void)setUp {
    [super setUp];
    game = [[BowlingGame alloc] init];
}

- (void)tearDown {
    game = nil;
    [super tearDown];
}

- (void)testGutterGame {
    NSUInteger n = 20;
    NSUInteger pins = 0;
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:1];
    XCTAssertEqual([game score], 20);
}

@end
```

Refactor menu / Extract

```objc
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _score;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _score += pins;
}

- (NSUInteger)score {
    return _score;
}

@end
```

- roll loop is duplicated

The second test.

```
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

- (void)setUp {
    [super setUp];
    game = [[BowlingGame alloc] init];
}

- (void)tearDown {
    game = nil;
    [super tearDown];
}

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    NSUInteger n = 20;
    NSUInteger pins = 0;
    [self rollPins:pins times:n];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:1];
    XCTAssertEqual([game score], 20);
}

@end
```

```
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _score;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _score += pins;
}

- (NSUInteger)score {
    return _score;
}

@end
```

- roll loop is duplicated

The second test.

```
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

- (void)setUp {
    [super setUp];
    game = [[BowlingGame alloc] init];
}

- (void)tearDown {
    game = nil;
    [super tearDown];
}

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    for (NSUInteger i = 0; i < 20; ++i)
        [game rollWithPinCount:1];
    XCTAssertEqual([game score], 20);
}

@end
```

```
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _score;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _score += pins;
}

- (NSUInteger)score {
    return _score;
}

@end
```

The second test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

- (void)setUp {
    [super setUp];
    game = [[BowlingGame alloc] init];
}

- (void)tearDown {
    game = nil;
    [super tearDown];
}

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

@end
```

```objc
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _score;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _score += pins;
}

- (NSUInteger)score {
    return _score;
}

@end
```

- ugly comment in test

The third test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

- (void)testOneSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5]; // spare
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

@end
```

```objc
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _score;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _score += pins;
}

- (NSUInteger)score {
    return _score;
}

@end
```

- ugly comment in test

The third test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

- (void)testOneSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5]; // spare
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

@end
```

```objc
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _score;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _score += pins;
}

- (NSUInteger)score {
    return _score;
}

@end
```

'13' should be equal to '16'

- ugly comment in test

The third test.

```
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

- (void)testOneSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5]; // spare
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

@end
```

```
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _score;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _score += pins;
}

- (NSUInteger)score {
    return _score;
}

@end
```

Tempted to use flag to remember previous roll. So design must be wrong.

- ugly comment in test

The third test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

- (void)testOneSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5]; // spare
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

@end
```

```objc
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _score;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _score += pins;
}

- (NSUInteger)score {
    return _score;
}

@end
```

-rollWithPinCount: calculates score, but name does not imply that.

-score does not calculate score, but name implies that it does.

Design is wrong. Responsibilities are misplaced.

- ugly comment in test

The third test.

```
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

//- (void)testOneSpare {
//    [game rollWithPinCount:5];
//    [game rollWithPinCount:5]; // spare
//    [game rollWithPinCount:3];
//    [self rollPins:0 times:17];
//    XCTAssertEqual([game score], 16);
//}

@end
```

```
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _score;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _score += pins;
}

- (NSUInteger)score {
    return _score;
}

@end
```

- ugly comment in test

The third test.

```
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

//- (void)testOneSpare {
//    [game rollWithPinCount:5];
//    [game rollWithPinCount:5]; // spare
//    [game rollWithPinCount:3];
//    [self rollPins:0 times:17];
//    XCTAssertEqual([game score], 16);
//}

@end
```

```
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _score;
    NSUInteger _rolls[21];
    NSUInteger _currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _score += pins;
    _rolls[_currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    for (NSUInteger i = 0; i < 21; ++i)
        score += _rolls[i];
    return score;
}

@end
```

- ugly comment in test

The third test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

//- (void)testOneSpare {
//    [game rollWithPinCount:5];
//    [game rollWithPinCount:5]; // spare
//    [game rollWithPinCount:3];
//    [self rollPins:0 times:17];
//    XCTAssertEqual([game score], 16);
//}

@end
```

```objc
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _score;
    NSUInteger _rolls[21];
    NSUInteger _currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _score += pins;
    _rolls[_currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    for (NSUInteger i = 0; i < 21; ++i)
        score += _rolls[i];
    return score;
}

@end
```

- ugly comment in test

The third test.

```
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

//- (void)testOneSpare {
//    [game rollWithPinCount:5];
//    [game rollWithPinCount:5]; // spare
//    [game rollWithPinCount:3];
//    [self rollPins:0 times:17];
//    XCTAssertEqual([game score], 16);
//}

@end
```

```
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _rolls[21];
    NSUInteger _currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _rolls[_currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    for (NSUInteger i = 0; i < 21; ++i)
        score += _rolls[i];
    return score;
}

@end
```

- ugly comment in test

The third test.

```
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

- (void)testOneSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5]; // spare
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

@end
```

```
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _rolls[21];
    NSUInteger _currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _rolls[_currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    for (NSUInteger i = 0; i < 21; ++i)
        score += _rolls[i];
    return score;
}

@end
```

("13") is not equal to ("16")

- ugly comment in test

The third test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

- (void)testOneSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5]; // spare
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

@end
```

```objc
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _rolls[21];
    NSUInteger _currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _rolls[_currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    for (NSUInteger i = 0; i < 21; ++i) {
        if (_rolls[i] + _rolls[i+1] == 10) // spare
            score += ...
        score += _rolls[i];
    }
    return score;
}

@end
```

This isn't going to work because "i" might not refer to the first ball of the frame.

Design is still wrong.

Need to walk through array two balls (one frame) at a time.

- ugly comment in test

The third test.

```
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

//- (void)testOneSpare {
//    [game rollWithPinCount:5];
//    [game rollWithPinCount:5];  // spare
//    [game rollWithPinCount:3];
//    [self rollPins:0 times:17];
//    XCTAssertEqual([game score], 16);
//}

@end
```

```
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _rolls[21];
    NSUInteger _currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _rolls[_currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    NSUInteger i = 0;
    for (NSUInteger frame = 0; frame < 10; ++frame) {
        score += _rolls[i] + _rolls[i + 1];
        i += 2;
    }
    return score;
}

@end
```

- ugly comment in test

The third test.

```
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

- (void)testOneSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5];  // spare
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

@end
```

```
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _rolls[21];
    NSUInteger _currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _rolls[_currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    NSUInteger i = 0;
    for (NSUInteger frame = 0; frame < 10; ++frame) {
        score += _rolls[i] + _rolls[i + 1];
        i += 2;
    }
    return score;
}

@end
```

("13") is not equal to ("16")

- ugly comment in test

The third test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

- (void)testOneSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5];  // spare
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

@end
```

```objc
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _rolls[21];
    NSUInteger _currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _rolls[_currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    NSUInteger i = 0;
    for (NSUInteger frame = 0; frame < 10; ++frame) {
        // spare
        if (_rolls[i] + _rolls[i + 1] == 10) {
            score += 10 + _rolls[i + 2];
            i += 2;
        } else {
            score += _rolls[i] + _rolls[i + 1];
            i += 2;
        }
    }
    return score;
}

@end
```

- ugly comment in test
- ugly comment in conditional
- i is a bad name for this variable

The third test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

- (void)testOneSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5];  // spare
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

@end
```

```objc
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _rolls[21];
    NSUInteger _currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _rolls[_currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    NSUInteger i = 0;
    for (NSUInteger frame = 0; frame < 10; ++frame) {
        // spare
        if (_rolls[i] + _rolls[i + 1] == 10) {
            score += 10 + _rolls[i + 2];
            i += 2;
        } else {
            score += _rolls[i] + _rolls[i + 1];
            i += 2;
        }
    }
    return score;
}

@end
```

- ugly comment in test
- ugly comment in conditional

The third test.

```
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

- (void)testOneSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5];  // spare
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

@end
```

```
#import <Foundation/Foundation.h>

@interface BowlingGame : NSObject
- (void)rollWithPinCount:(NSUInteger)pins;
- (NSUInteger)score;
@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _rolls[21];
    NSUInteger _currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _rolls[_currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    NSUInteger ballIndex = 0;
    for (NSUInteger frame = 0; frame < 10; ++frame) {
        // spare
        if (_rolls[ballIndex] + _rolls[ballIndex + 1] == 10) {
            score += 10 + _rolls[ballIndex + 2];
            ballIndex += 2;
        } else {
            score += _rolls[ballIndex] + _rolls[ballIndex + 1];
            ballIndex += 2;
        }
    }
    return score;
}

@end
```

Renamed using "Edit All in Scope"

- ugly comment in test

The third test.

```
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

- (void)testOneSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5];  // spare
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _rolls[21];
    NSUInteger _currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _rolls[_currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    NSUInteger ballIndex = 0;
    for (NSUInteger frame = 0; frame < 10; ++frame) {
        if ([self isSpare:ballIndex]) {
            score += 10 + _rolls[ballIndex + 2];
            ballIndex += 2;
        } else {
            score += _rolls[ballIndex] +
                     _rolls[ballIndex + 1];
            ballIndex += 2;
        }
    }
    return score;
}

- (BOOL)isSpare:(NSUInteger)ballIndex {
    return _rolls[ballIndex] + _rolls[ballIndex + 1] == 10;
}

@end
```

The third test.

```
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollPins:(NSUInteger)pins times:(NSUInteger)n {
    for (NSUInteger i = 0; i < n; ++i)
        [game rollWithPinCount:pins];
}

- (void)testGutterGame {
    [self rollPins:0 times:20];
    XCTAssertEqual([game score], 0);
}

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

- (void)rollSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5];
}

- (void)testOneSpare {
    [self rollSpare];
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _rolls[21];
    NSUInteger _currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _rolls[_currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    NSUInteger ballIndex = 0;
    for (NSUInteger frame = 0; frame < 10; ++frame) {
        if ([self isSpare:ballIndex]) {
            score += 10 + _rolls[ballIndex + 2];
            ballIndex += 2;
        } else {
            score += _rolls[ballIndex] +
                     _rolls[ballIndex + 1];
            ballIndex += 2;
        }
    }
    return score;
}

- (BOOL)isSpare:(NSUInteger)ballIndex {
    return _rolls[ballIndex] + _rolls[ballIndex + 1] == 10;
}

@end
```

- ugly comment in test

The fourth test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

- (void)rollSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5];
}

- (void)testOneSpare {
    [self rollSpare];
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

- (void)testOneStrike {
    [game rollWithPinCount:10]; // strike
    [game rollWithPinCount:3];
    [game rollWithPinCount:4];
    [self rollPins:0 times:16];
    XCTAssertEqual([game score], 24);
}

@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger rolls[21];
    NSUInteger currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    rolls[currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    NSUInteger ballIndex = 0;
    for (NSUInteger frame = 0; frame < 10; ++frame) {
        if ([self isSpare:ballIndex]) {
            score += 10 + rolls[ballIndex + 2];
            ballIndex += 2;
        } else {
            score += rolls[ballIndex] +
                     rolls[ballIndex + 1];
            ballIndex += 2;
        }
    }
    return score;
}

- (BOOL)isSpare:(NSUInteger)ballIndex {
    return rolls[ballIndex] + rolls[ballIndex + 1] == 10;
}

@end
```

("17") is not equal to ("24")

- ugly comment in test

The fourth test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

- (void)rollSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5];
}

- (void)testOneSpare {
    [self rollSpare];
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

- (void)testOneStrike {
    [game rollWithPinCount:10]; // strike
    [game rollWithPinCount:3];
    [game rollWithPinCount:4];
    [self rollPins:0 times:16];
    XCTAssertEqual([game score], 24);
}

@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _rolls[21];
    NSUInteger _currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _rolls[_currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    NSUInteger ballIndex = 0;
    for (NSUInteger frame = 0; frame < 10; ++frame) {
        if (_rolls[ballIndex] == 10) { // strike
            score += 10 +
                     _rolls[ballIndex + 1] +
                     _rolls[ballIndex + 2];
            ++ballIndex;
        } else if ([self isSpare:ballIndex]) {
            score += 10 + _rolls[ballIndex + 2];
            ballIndex += 2;
        } else {
            score += _rolls[ballIndex] +
                     _rolls[ballIndex + 1];
            ballIndex += 2;
        }
    }
    return score;
}

- (BOOL)isSpare:(NSUInteger)ballIndex {
    return _rolls[ballIndex] + _rolls[ballIndex + 1] == 10;
}

@end
```

- ugly comment in test
- ugly comment in conditional
- ugly expressions

The fourth test.

```objc
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

- (void)rollSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5];
}

- (void)testOneSpare {
    [self rollSpare];
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

- (void)testOneStrike {
    [game rollWithPinCount:10]; // strike
    [game rollWithPinCount:3];
    [game rollWithPinCount:4];
    [self rollPins:0 times:16];
    XCTAssertEqual([game score], 24);
}

@end
```

```objc
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _rolls[21];
    NSUInteger _currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _rolls[_currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    NSUInteger ballIndex = 0;
    for (NSUInteger frame = 0; frame < 10; ++frame) {
        if (_rolls[ballIndex] == 10) { // strike
            score += 10 +
                     _rolls[ballIndex + 1] +
                     _rolls[ballIndex + 2];
            ++ballIndex;
        } else if ([self isSpare:ballIndex]) {
            score += 10 + _rolls[ballIndex + 2];
            ballIndex += 2;
        } else {
            score += _rolls[ballIndex] +
                     _rolls[ballIndex + 1];
            ballIndex += 2;
        }
    }
    return score;
}

- (BOOL)isSpare:(NSUInteger)ballIndex {
    return _rolls[ballIndex] + _rolls[ballIndex + 1] == 10;
}

@end
```

- ugly comment in test
- ugly comment in conditional

The fourth test.

```
...

- (void)rollSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5];
}

- (void)testOneSpare {
    [self rollSpare];
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

- (void)testOneStrike {
    [game rollWithPinCount:10]; // strike
    [game rollWithPinCount:3];
    [game rollWithPinCount:4];
    [self rollPins:0 times:16];
    XCTAssertEqual([game score], 24);
}

@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _rolls[21];
    NSUInteger _currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _rolls[_currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    NSUInteger ballIndex = 0;
    for (NSUInteger frame = 0; frame < 10; ++frame) {
        if (_rolls[ballIndex] == 10) { // strike
            score += 10 + [self strikeBonus:ballIndex];
            ++ballIndex;
        } else if ([self isSpare:ballIndex]) {
            score += 10 + [self spareBonus:ballIndex];
            ballIndex += 2;
        } else {
            score += [self sumOfBallsInFrame:ballIndex];
            ballIndex += 2;
        }
    }
    return score;
}

- (BOOL)isSpare:(NSUInteger)ballIndex {
    return _rolls[ballIndex] + _rolls[ballIndex + 1] == 10;
}

- (NSUInteger)strikeBonus:(NSUInteger)ballIndex {
    return _rolls[ballIndex + 1] + _rolls[ballIndex + 2];
}

- (NSUInteger)spareBonus:(NSUInteger)ballIndex {
    return _rolls[ballIndex + 2];
}

- (NSUInteger)sumOfBallsInFrame:(NSUInteger)ballIndex {
    return _rolls[ballIndex] + _rolls[ballIndex + 1];
}

@end
```

- ugly comment in test

The fourth test.

```
...

- (void)rollSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5];
}

- (void)testOneSpare {
    [self rollSpare];
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

- (void)testOneStrike {
    [game rollWithPinCount:10];  // strike
    [game rollWithPinCount:3];
    [game rollWithPinCount:4];
    [self rollPins:0 times:16];
    XCTAssertEqual([game score], 24);
}

@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger rolls[21];
    NSUInteger currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    rolls[currentRoll++] = pins;
}
```

```
- (NSUInteger)score {
    NSUInteger score = 0;
    NSUInteger ballIndex = 0;
    for (NSUInteger frame = 0; frame < 10; ++frame) {
        if ([self isStrike:ballIndex]) {
            score += 10 + [self strikeBonus:ballIndex];
            ++ballIndex;
        } else if ([self isSpare:ballIndex]) {
            score += 10 + [self spareBonus:ballIndex];
            ballIndex += 2;
        } else {
            score += [self sumOfBallsInFrame:ballIndex];
            ballIndex += 2;
        }
    }
    return score;
}
}

- (BOOL)isStrike:(NSUInteger)ballIndex {
    return _rolls[ballIndex] == 10;
}

- (BOOL)isSpare:(NSUInteger)ballIndex {
    return rolls[ballIndex] + rolls[ballIndex + 1] == 10;
}

- (NSUInteger)strikeBonus:(NSUInteger)ballIndex {
    return rolls[ballIndex + 1] + rolls[ballIndex + 2];
}

- (NSUInteger)spareBonus:(NSUInteger)ballIndex {
    return rolls[ballIndex + 2];
}

- (NSUInteger)sumOfBallsInFrame:(NSUInteger)ballIndex {
    return rolls[ballIndex] + rolls[ballIndex + 1];
}

@end
```

The fourth test.

```
#import "BowlingGame.h"
#import <XCTest/XCTest.h>

@interface BowlingGameTests : XCTestCase
@end

@implementation BowlingGameTests {
    BowlingGame *game;
}

...

- (void)rollStrike {
    [game rollWithPinCount:10];
}

- (void)rollSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5];
}

- (void)testOneSpare {
    [self rollSpare];
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

- (void)testOneStrike {
    [self rollStrike];
    [game rollWithPinCount:3];
    [game rollWithPinCount:4];
    [self rollPins:0 times:16];
    XCTAssertEqual([game score], 24);
}

@end
```

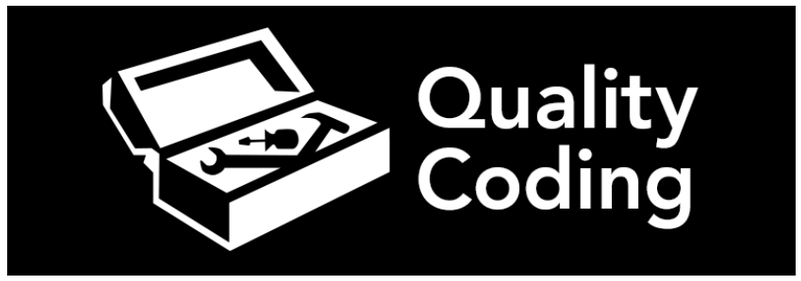
```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _rolls[21];
    NSUInteger _currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _rolls[_currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    NSUInteger ballIndex = 0;
    for (NSUInteger frame = 0; frame < 10; ++frame) {
        if ([self isStrike:ballIndex]) {
            score += 10 + [self strikeBonus:ballIndex];
            ++ballIndex;
        } else if ([self isSpare:ballIndex]) {
            score += 10 + [self spareBonus:ballIndex];
            ballIndex += 2;
        } else {
            score += [self sumOfBallsInFrame:ballIndex];
            ballIndex += 2;
        }
    }
    return score;
}

...
```

The fifth test.

```
...

- (void)testAllOnes {
    [self rollPins:1 times:20];
    XCTAssertEqual([game score], 20);
}

- (void)rollStrike {
    [game rollWithPinCount:10];
}

- (void)rollSpare {
    [game rollWithPinCount:5];
    [game rollWithPinCount:5];
}

- (void)testOneSpare {
    [self rollSpare];
    [game rollWithPinCount:3];
    [self rollPins:0 times:17];
    XCTAssertEqual([game score], 16);
}

- (void)testOneStrike {
    [self rollStrike];
    [game rollWithPinCount:3];
    [game rollWithPinCount:4];
    [self rollPins:0 times:16];
    XCTAssertEqual([game score], 24);
}

- (void)testPerfectGame {
    [self rollPins:10 times:12];
    XCTAssertEqual([game score], 300);
}

@end
```

```
#import "BowlingGame.h"

@implementation BowlingGame {
    NSUInteger _rolls[21];
    NSUInteger _currentRoll;
}

- (void)rollWithPinCount:(NSUInteger)pins {
    _rolls[_currentRoll++] = pins;
}

- (NSUInteger)score {
    NSUInteger score = 0;
    NSUInteger ballIndex = 0;
    for (NSUInteger frame = 0; frame < 10; ++frame) {
        if ([self isStrike:ballIndex]) {
            score += 10 + [self strikeBonus:ballIndex];
            ++ballIndex;
        } else if ([self isSpare:ballIndex]) {
            score += 10 + [self spareBonus:ballIndex];
            ballIndex += 2;
        } else {
            score += [self sumOfBallsInFrame:ballIndex];
            ballIndex += 2;
        }
    }
    return score;
}

...
```

End



QualityCoding.org