Bowling Game Kata

Object Mentor, Inc.

www.objectmentor.com
blog.objectmentor.com

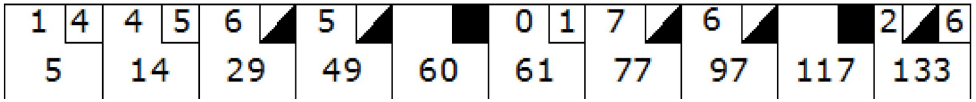
fitnesse.org

XProgramming.com

JU.org

www.junit.org

Scoring Bowling.



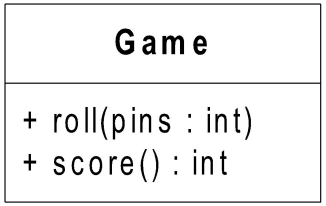
The game consists of 10 frames as shown above. In each frame the player has two opportunities to knock down 10 pins. The score for the frame is the total number of pins knocked down, plus bonuses for strikes and spares.

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.

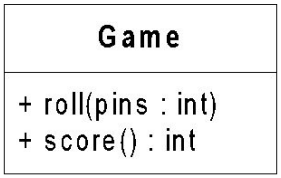
In the tenth frame a player who rolls a spare or strike is allowed to roll the extra balls to complete the frame. However no more than three balls can be rolled in tenth frame.

The Requirements.



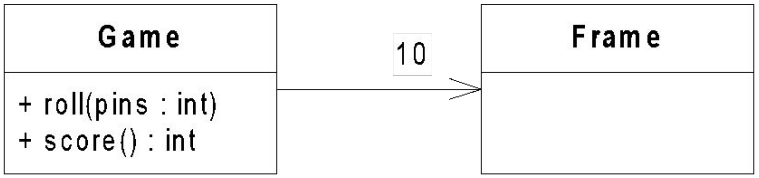
- Write a class named “Game” that has two methods
  - roll(pins : int) is called each time the player rolls a ball. The argument is the number of pins knocked down.
  - score() : int is called only at the very end of the game. It returns the total score for that game.

A quick design session



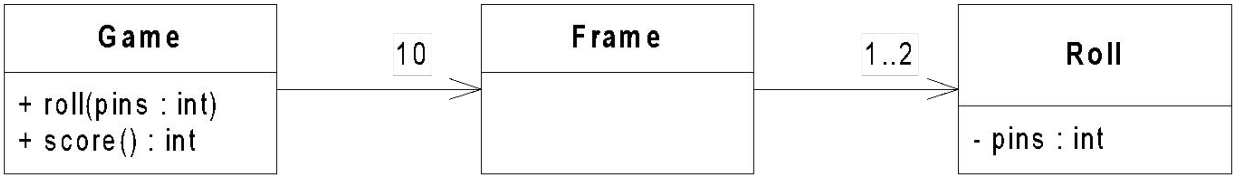
Clearly we need the Game class.

A quick design session



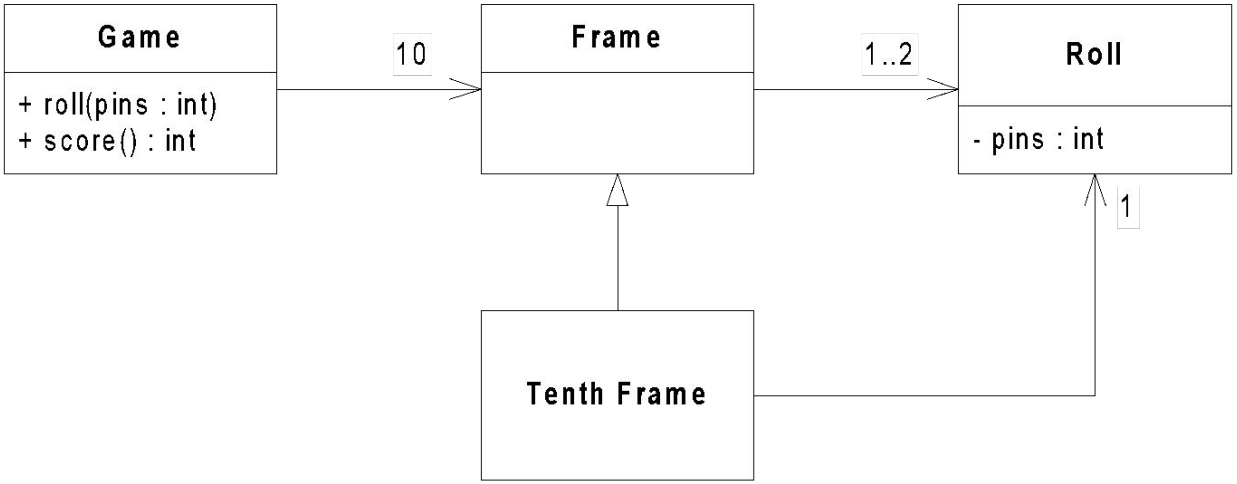
A game has 10 frames.

A quick design session



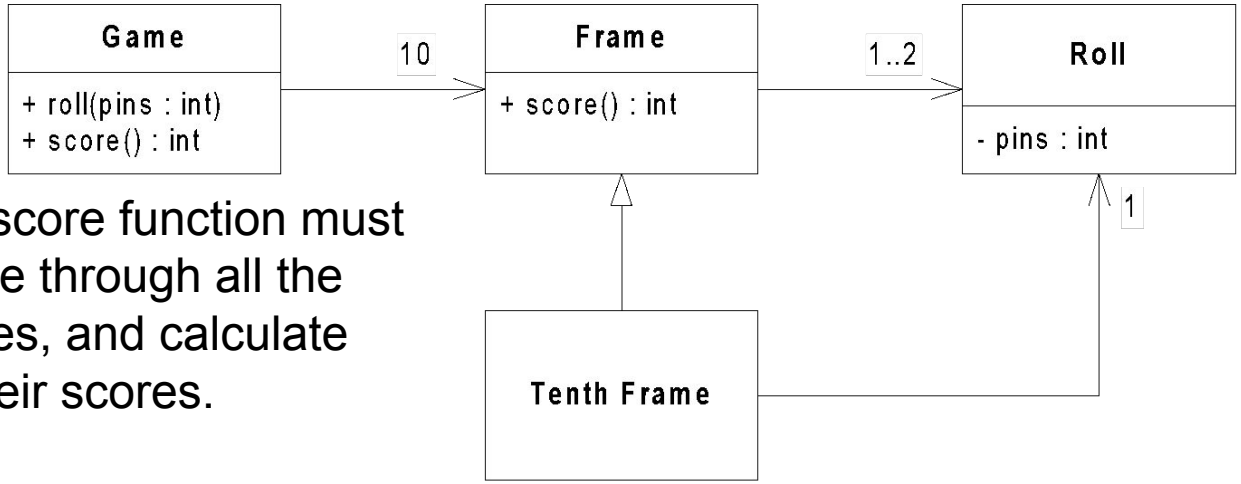
A frame has 1 or two rolls.

A quick design session



The tenth frame has two or three rolls.
It is different from all the other frames.

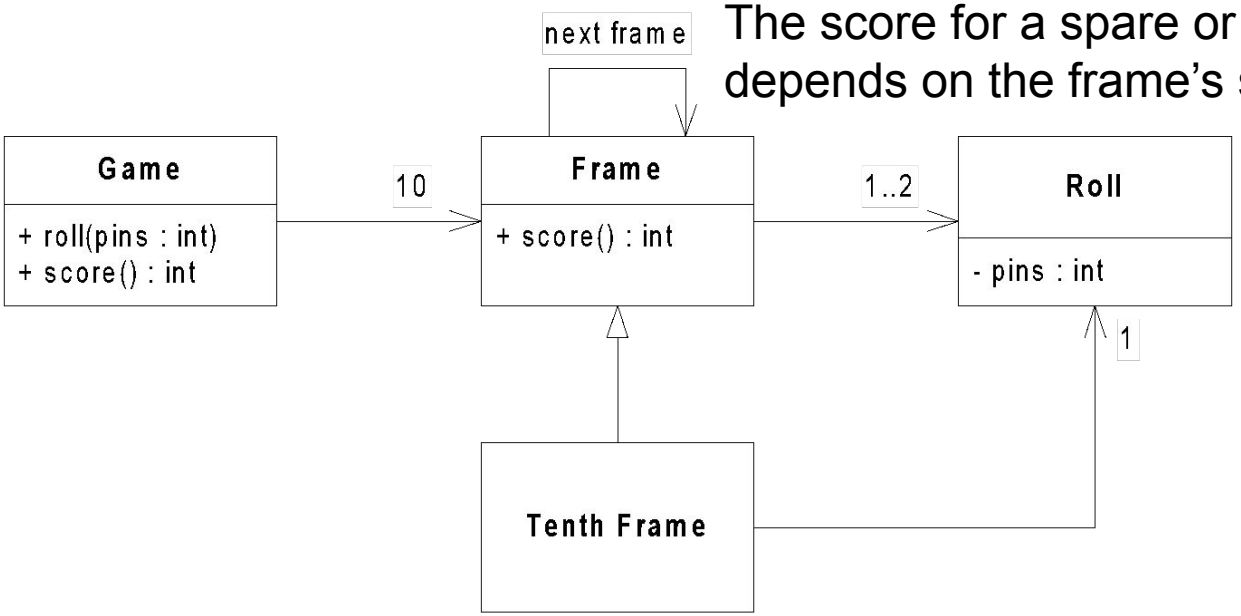
A quick design session



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 a project named BowlingGame
- Create a unit test named BowlingGameTest

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
}
```

Begin.

- Create a project named BowlingGame
- Create a unit test named BowlingGameTest

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
}
```

Execute this program and verify that you get the following error:

```
No tests found in BowlingGameTest
```

The first test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  public void testGutterGame() throws Exception {
    Game g = new Game();
  }
}
```

The first test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  public void testGutterGame() throws Exception {
    Game g = new Game();
  }
}
```

```
public class Game {
}
```

The first test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  public void testGutterGame() throws Exception {
    Game g = new Game();
  }
}
```

```
public class Game {
}
```

The first test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  public void testGutterGame() throws Exception {
    Game g = new Game();
    for (int i=0; i<20; i++)
      g.roll(0);
  }
}
```

```
public class Game {
}
```

The first test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  public void testGutterGame() throws Exception {
    Game g = new Game();
    for (int i=0; i<20; i++)
      g.roll(0);
  }
}
```

```
public class Game {
  public void roll(int pins) {
  }
}
```

The first test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  public void testGutterGame() throws Exception {
    Game g = new Game();
    for (int i=0; i<20; i++)
      g.roll(0);
    assertEquals(0, g.score());
  }
}
```

```
public class Game {
  public void roll(int pins) {
  }
}
```

The first test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  public void testGutterGame() throws Exception {
    Game g = new Game();
    for (int i=0; i<20; i++)
      g.roll(0);
    assertEquals(0, g.score());
  }
}
```

```
public class Game {
  public void roll(int pins) {
  }

  public int score() {
    return -1;
  }
}
```

```
expected:<0> but was:<-1>
```

The first test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  public void testGutterGame() throws Exception {
    Game g = new Game();
    for (int i=0; i<20; i++)
      g.roll(0);
    assertEquals(0, g.score());
  }
}
```

```
public class Game {
  public void roll(int pins) {
  }

  public int score() {
    return 0;
  }
}
```

The Second test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  public void testGutterGame() throws Exception {
    Game g = new Game();
    for (int i = 0; i < 20; i++)
      g.roll(0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    Game g = new Game();
    for (int i = 0; i < 20; i++)
      g.roll(1);
    assertEquals(20, g.score());
  }
}
```

```
public class Game {
  public void roll(int pins) {
  }

  public int score() {
    return 0;
  }
}
```

- Game creation is duplicated
- roll loop is duplicated

The Second test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  public void testGutterGame() throws Exception {
    Game g = new Game();
    for (int i = 0; i < 20; i++)
      g.roll(0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    Game g = new Game();
    for (int i = 0; i < 20; i++)
      g.roll(1);
    assertEquals(20, g.score());
  }
}
```

```
public class Game {
  public void roll(int pins) {
  }

  public int score() {
    return 0;
  }
}
```

- Game creation is duplicated
- roll loop is duplicated

The Second test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  public void testGutterGame() throws Exception {
    Game g = new Game();
    for (int i = 0; i < 20; i++)
      g.roll(0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    Game g = new Game();
    for (int i = 0; i < 20; i++)
      g.roll(1);
    assertEquals(20, g.score());
  }
}
```

```
public class Game {
  public void roll(int pins) {
  }

  public int score() {
    return 0;
  }
}
```

```
expected:<20> but was:<0>
```

- roll loop is duplicated

The Second test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  public void testGutterGame() throws Exception {
    for (int i = 0; i < 20; i++)
      g.roll(0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    for (int i = 0; i < 20; i++)
      g.roll(1);
    assertEquals(20, g.score());
  }
}
```

```
public class Game {
  private int score = 0;

  public void roll(int pins) {
    score += pins;
  }

  public int score() {
    return score;
  }
}
```

- roll loop is duplicated

The Second test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  public void testGutterGame() throws Exception {
    int n = 20;
    int pins = 0;
    for (int i = 0; i < n; i++) {
      g.roll(pins);
    }
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    for (int i = 0; i < 20; i++)
      g.roll(1);
    assertEquals(20, g.score());
  }
}
```

```
public class Game {
  private int score = 0;

  public void roll(int pins) {
    score += pins;
  }

  public int score() {
    return score;
  }
}
```

- roll loop is duplicated

The Second test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  public void testGutterGame() throws Exception {
    int n = 20;
    int pins = 0;
    rollMany(n, pins);
    assertEquals(0, g.score());
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testAllOnes() throws Exception {
    for (int i = 0; i < 20; i++)
      g.roll(1);
    assertEquals(20, g.score());
  }
}
```

```
public class Game {
  private int score = 0;

  public void roll(int pins) {
    score += pins;
  }

  public int score() {
    return score;
  }
}
```

- roll loop is duplicated

The Second test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testAllOnes() throws Exception {
    for (int i = 0; i < 20; i++)
      g.roll(1);
    assertEquals(20, g.score());
  }
}
```

```
public class Game {
  private int score = 0;

  public void roll(int pins) {
    score += pins;
  }

  public int score() {
    return score;
  }
}
```

- roll loop is duplicated

The Second test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }
}
```

```
public class Game {
  private int score = 0;

  public void roll(int pins) {
    score += pins;
  }

  public int score() {
    return score;
  }
}
```

The Second test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }
}
```

```
public class Game {
  private int score = 0;

  public void roll(int pins) {
    score += pins;
  }

  public int score() {
    return score;
  }
}
```

- ugly comment in test.

The Third test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    g.roll(5);
    g.roll(5); // spare
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }
}
```

```
public class Game {
  private int score = 0;

  public void roll(int pins) {
    score += pins;
  }

  public int score() {
    return score;
  }
}
```

- ugly comment in test.

The Third test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    g.roll(5);
    g.roll(5); // spare
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }
}
```

```
public class Game {
  private int score = 0;

  public void roll(int pins) {
    score += pins;
  }

  public int score() {
    return score;
  }
}
```

```
expected:<16> but was:<13>
```

- ugly comment in test.

The Third test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    g.roll(5);
    g.roll(5); // spare
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }
}
```

```
public class Game {
  private int score = 0;

  public void roll(int pins) {
    score += pins;
  }

  public int score() {
    return score;
  }
}
```

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

- ugly comment in test.

The Third test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    g.roll(5);
    g.roll(5); // spare
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }
}
```

```
public class Game {
  private int score = 0;

  public void roll(int pins) {
    score += pins;
  }

  public int score() {
    return score;
  }
}
```

roll() 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 junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

//  public void testOneSpare() throws Exception {
//    g.roll(5);
//    g.roll(5); // spare
//    g.roll(3);
//    rollMany(17,0);
//    assertEquals(16,g.score());
//  }
}
```

```
public class Game {
  private int score = 0;

  public void roll(int pins) {
    score += pins;
  }

  public int score() {
    return score;
  }
}
```

- ugly comment in test.

The Third test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

//  public void testOneSpare() throws Exception {
//    g.roll(5);
//    g.roll(5); // spare
//    g.roll(3);
//    rollMany(17,0);
//    assertEquals(16,g.score());
//  }
}
```

```
public class Game {
  private int score = 0;
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    score += pins;
    rolls[currentRoll++] = pins;
  }

  public int score() {
    return score;
  }
}
```

- ugly comment in test.

The Third test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

//  public void testOneSpare() throws Exception {
//    g.roll(5);
//    g.roll(5); // spare
//    g.roll(3);
//    rollMany(17,0);
//    assertEquals(16,g.score());
//  }
}
```

```
public class Game {
  private int score = 0;
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    score += pins;
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    for (int i = 0; i < rolls.length; i++)
      score += rolls[i];
    return score;
  }
}
```

- ugly comment in test.

The Third test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

//  public void testOneSpare() throws Exception {
//    g.roll(5);
//    g.roll(5); // spare
//    g.roll(3);
//    rollMany(17,0);
//    assertEquals(16,g.score());
//  }
}
```

```
public class Game {
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    for (int i = 0; i < rolls.length; i++)
      score += rolls[i];
    return score;
  }
}
```

- ugly comment in test.

The Third test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    g.roll(5);
    g.roll(5); // spare
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }
}
```

```
public class Game {
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    for (int i = 0; i < rolls.length; i++)
      score += rolls[i];
    return score;
  }
}
```

```
expected:<16> but was:<13>
```

- ugly comment in test.

The Third test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    g.roll(5);
    g.roll(5); // spare
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }
}
```

```
public class Game {
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    for (int i = 0; i < rolls.length; i++)  {
      if (rolls[i] + rolls[i+1] == 10) // spare
        score += ...
      score += rolls[i];
    }
    return score;
  }
}
```

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 junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

//  public void testOneSpare() throws Exception {
//    g.roll(5);
//    g.roll(5); // spare
//    g.roll(3);
//    rollMany(17,0);
//    assertEquals(16,g.score());
//  }
}
```

```
public class Game {
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    for (int i = 0; i < rolls.length; i++)
      score += rolls[i];
    return score;
  }
}
```

- ugly comment in test.

The Third test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

//  public void testOneSpare() throws Exception {
//    g.roll(5);
//    g.roll(5); // spare
//    g.roll(3);
//    rollMany(17,0);
//    assertEquals(16,g.score());
//  }
}
```

```
public class Game {
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    int i = 0;
    for (int frame = 0; frame < 10; frame++)  {
      score += rolls[i] + rolls[i+1];
      i += 2;
    }
    return score;
  }
}
```

- ugly comment in test.

The Third test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    g.roll(5);
    g.roll(5); // spare
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }
}
```

```
public class Game {
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    int i = 0;
    for (int frame = 0; frame < 10; frame++)  {
      score += rolls[i] + rolls[i+1];
      i += 2;
    }
    return score;
  }
}
```

```
expected:<16> but was:<13>
```

- ugly comment in test.

The Third test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    g.roll(5);
    g.roll(5); // spare
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }
}
```

```
public class Game {
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    int i = 0;
    for (int frame = 0; frame < 10; frame++) {
      if (rolls[i] + rolls[i + 1] == 10) // spare
      {
        score += 10 + rolls[i + 2];
        i += 2;
      } else {
        score += rolls[i] + rolls[i + 1];
        i += 2;
      }
    }
    return score;
  }
}
```

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

The Third test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    g.roll(5);
    g.roll(5); // spare
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }
}
```

```
public class Game {
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    int i = 0;
    for (int frame = 0; frame < 10; frame++) {
      if (rolls[i] + rolls[i + 1] == 10) // spare
      {
        score += 10 + rolls[i + 2];
        i += 2;
      } else {
        score += rolls[i] + rolls[i + 1];
        i += 2;
      }
    }
    return score;
  }
}
```

-ugly comment in test.
-ugly comment in conditional.

The Third test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    g.roll(5);
    g.roll(5); // spare
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }
}
```

```
public class Game {
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    int frameIndex = 0;
    for (int frame = 0; frame < 10; frame++) {
      if (rolls[frameIndex] +
          rolls[frameIndex + 1] == 10) // spare
      {
        score += 10 + rolls[frameIndex + 2];
        frameIndex += 2;
      } else {
        score += rolls[frameIndex] +
                 rolls[frameIndex + 1];
        frameIndex += 2;
      }
    }
    return score;
  }
}
```

-ugly comment in test.

The Third test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    g.roll(5);
    g.roll(5); // spare
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }
}
```

```
public class Game {
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    int frameIndex = 0;
    for (int frame = 0; frame < 10; frame++) {
      if (isSpare(frameIndex))
      {
        score += 10 + rolls[frameIndex + 2];
        frameIndex += 2;
      } else {
        score += rolls[frameIndex] +
                 rolls[frameIndex + 1];
        frameIndex += 2;
      }
    }
    return score;
  }

  private boolean isSpare(int frameIndex) {
    return rolls[frameIndex] +
           rolls[frameIndex + 1] == 10;
  }
}
```

-

The Third test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  private Game g;

  protected void setUp() throws Exception {
    g = new Game();
  }

  private void rollMany(int n, int pins) {
    for (int i = 0; i < n; i++)
      g.roll(pins);
  }

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    rollSpare();
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }

  private void rollSpare() {
    g.roll(5);
    g.roll(5);
  }
}
```

```
public class Game {
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    int frameIndex = 0;
    for (int frame = 0; frame < 10; frame++) {
      if (isSpare(frameIndex))
      {
        score += 10 + rolls[frameIndex + 2];
        frameIndex += 2;
      } else {
        score += rolls[frameIndex] +
                 rolls[frameIndex + 1];
        frameIndex += 2;
      }
    }
    return score;
  }

  private boolean isSpare(int frameIndex) {
    return rolls[frameIndex] +
           rolls[frameIndex + 1] == 10;
  }
}
```

- ugly comment in testOneStrike

The Fourth test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  ...

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    rollSpare();
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }

  public void testOneStrike() throws Exception {
    g.roll(10); // strike
    g.roll(3);
    g.roll(4);
    rollMany(16, 0);
    assertEquals(24, g.score());
  }

  private void rollSpare() {
    g.roll(5);
    g.roll(5);
  }
}
```

```
public class Game {
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    int frameIndex = 0;
    for (int frame = 0; frame < 10; frame++) {
      if (isSpare(frameIndex))
      {
        score += 10 + rolls[frameIndex + 2];
        frameIndex += 2;
      } else {
        score += rolls[frameIndex] +
                 rolls[frameIndex + 1];
        frameIndex += 2;
      }
    }
    return score;
  }

  private boolean isSpare(int frameIndex) {
    return rolls[frameIndex] +
           rolls[frameIndex + 1] == 10;
  }
}
```

```
expected:<24> but was:<17>
```

-ugly comment in testOneStrike.
-ugly comment in conditional.
-ugly expressions.

The Fourth test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  ...

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    rollSpare();
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }

  public void testOneStrike() throws Exception {
    g.roll(10); // strike
    g.roll(3);
    g.roll(4);
    rollMany(16, 0);
    assertEquals(24, g.score());
  }

  private void rollSpare() {
    g.roll(5);
    g.roll(5);
  }
}
```

```
public class Game {
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    int frameIndex = 0;
    for (int frame = 0; frame < 10; frame++) {
      if (rolls[frameIndex] == 10) // strike
      {
        score += 10 +
                 rolls[frameIndex+1] +
                 rolls[frameIndex+2];
        frameIndex++;
      }
      else if (isSpare(frameIndex))
      {
        score += 10 + rolls[frameIndex + 2];
        frameIndex += 2;
      } else {
        score += rolls[frameIndex] +
                 rolls[frameIndex + 1];
        frameIndex += 2;
      }
    }
    return score;
  }

  private boolean isSpare(int frameIndex) {
    return rolls[frameIndex] +
           rolls[frameIndex + 1] == 10;
  }
}
```

-ugly comment in testOneStrike.
-ugly comment in conditional.

The Fourth test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  ...

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    rollSpare();
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }

  public void testOneStrike() throws Exception {
    g.roll(10); // strike
    g.roll(3);
    g.roll(4);
    rollMany(16, 0);
    assertEquals(24, g.score());
  }

  private void rollSpare() {
    g.roll(5);
    g.roll(5);
  }
}
```

```
public class Game {
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    int frameIndex = 0;
    for (int frame = 0; frame < 10; frame++) {
      if (rolls[frameIndex] == 10) // strike
      {
        score += 10 + strikeBonus(frameIndex);
        frameIndex++;
      } else if (isSpare(frameIndex)) {
        score += 10 + spareBonus(frameIndex);
        frameIndex += 2;
      } else {
        score += sumOfBallsInFrame(frameIndex);
        frameIndex += 2;
      }
    }
    return score;
  }

  private int sumOfBallsInFrame(int frameIndex) {
    return rolls[frameIndex]+rolls[frameIndex+1];
  }

  private int spareBonus(int frameIndex) {
    return rolls[frameIndex + 2];
  }

  private int strikeBonus(int frameIndex) {
    return rolls[frameIndex+1]+rolls[frameIndex+2];
  }

  private boolean isSpare(int frameIndex) {
    return rolls[frameIndex]+rolls[frameIndex+1] == 10;
  }
}
```

-ugly comment in testOneStrike.

The Fourth test.

```
import junit.framework.TestCase;

public class BowlingGameTest extends TestCase {
  ...

  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    rollSpare();
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }

  public void testOneStrike() throws Exception {
    g.roll(10); // strike
    g.roll(3);
    g.roll(4);
    rollMany(16, 0);
    assertEquals(24, g.score());
  }

  private void rollSpare() {
    g.roll(5);
    g.roll(5);
  }
}
```

```
public class Game {
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    int frameIndex = 0;
    for (int frame = 0; frame < 10; frame++) {
      if (isStrike(frameIndex)) {
        score += 10 + strikeBonus(frameIndex);
        frameIndex++;
      } else if (isSpare(frameIndex)) {
        score += 10 + spareBonus(frameIndex);
        frameIndex += 2;
      } else {
        score += sumOfBallsInFrame(frameIndex);
        frameIndex += 2;
      }
    }
    return score;
  }

  private boolean isStrike(int frameIndex) {
    return rolls[frameIndex] == 10;
  }

  private int sumOfBallsInFrame(int frameIndex) {
    return rolls[frameIndex] + rolls[frameIndex+1];
  }

  private int spareBonus(int frameIndex) {
    return rolls[frameIndex+2];
  }

  private int strikeBonus(int frameIndex) {
    return rolls[frameIndex+1] + rolls[frameIndex+2];
  }

  private boolean isSpare(int frameIndex) {
    return rolls[frameIndex]+rolls[frameIndex+1] == 10;
  }
}
```

The Fourth test.

```
  ...
  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    rollSpare();
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }

  public void testOneStrike() throws Exception {
    rollStrike();
    g.roll(3);
    g.roll(4);
    rollMany(16, 0);
    assertEquals(24, g.score());
  }

  private void rollStrike() {
    g.roll(10);
  }

  private void rollSpare() {
    g.roll(5);
    g.roll(5);
  }
}
```

```
public class Game {
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    int frameIndex = 0;
    for (int frame = 0; frame < 10; frame++) {
      if (isStrike(frameIndex)) {
        score += 10 + strikeBonus(frameIndex);
        frameIndex++;
      } else if (isSpare(frameIndex)) {
        score += 10 + spareBonus(frameIndex);
        frameIndex += 2;
      } else {
        score += sumOfBallsInFrame(frameIndex);
        frameIndex += 2;
      }
    }
    return score;
  }

  private boolean isStrike(int frameIndex) {
    return rolls[frameIndex] == 10;
  }

  private int sumOfBallsInFrame(int frameIndex) {
    return rolls[frameIndex] + rolls[frameIndex+1];
  }

  private int spareBonus(int frameIndex) {
    return rolls[frameIndex+2];
  }

  private int strikeBonus(int frameIndex) {
    return rolls[frameIndex+1] + rolls[frameIndex+2];
  }

  private boolean isSpare(int frameIndex) {
    return rolls[frameIndex]+rolls[frameIndex+1] == 10;
  }
}
```

The Fifth test.

```
  ...
  public void testGutterGame() throws Exception {
    rollMany(20, 0);
    assertEquals(0, g.score());
  }

  public void testAllOnes() throws Exception {
    rollMany(20,1);
    assertEquals(20, g.score());
  }

  public void testOneSpare() throws Exception {
    rollSpare();
    g.roll(3);
    rollMany(17,0);
    assertEquals(16,g.score());
  }

  public void testOneStrike() throws Exception {
    rollStrike();
    g.roll(3);
    g.roll(4);
    rollMany(16, 0);
    assertEquals(24, g.score());
  }

  public void testPerfectGame() throws Exception {
    rollMany(12,10);
    assertEquals(300, g.score());
  }

  private void rollStrike() {
    g.roll(10);
  }

  private void rollSpare() {
    g.roll(5);
    g.roll(5);
  }
}
```

```
public class Game {
  private int rolls[] = new int[21];
  private int currentRoll = 0;

  public void roll(int pins) {
    rolls[currentRoll++] = pins;
  }

  public int score() {
    int score = 0;
    int frameIndex = 0;
    for (int frame = 0; frame < 10; frame++) {
      if (isStrike(frameIndex)) {
        score += 10 + strikeBonus(frameIndex);
        frameIndex++;
      } else if (isSpare(frameIndex)) {
        score += 10 + spareBonus(frameIndex);
        frameIndex += 2;
      } else {
        score += sumOfBallsInFrame(frameIndex);
        frameIndex += 2;
      }
    }
    return score;
  }

  private boolean isStrike(int frameIndex) {
    return rolls[frameIndex] == 10;
  }

  private int sumOfBallsInFrame(int frameIndex) {
    return rolls[frameIndex] + rolls[frameIndex+1];
  }

  private int spareBonus(int frameIndex) {
    return rolls[frameIndex+2];
  }

  private int strikeBonus(int frameIndex) {
    return rolls[frameIndex+1] + rolls[frameIndex+2];
  }

  private boolean isSpare(int frameIndex) {
    return rolls[frameIndex]+rolls[frameIndex+1] == 10;
  }
}
```

End