The Open/Closed Principle Dojo

Matteo Vaccari & Antonio Carpentieri
matteo.vaccari@xpeppers.com, antonio.carpentieri@xpeppers.com
www.xpeppers.com

XP Days Benelux 2010

The FizzBuzz Game

1, 2, Fizz!, 4, Buzz!, Fizz!, 7,
8, Fizz!, Buzz!, 11, Fizz!, 13,
14, FizzBuzz!, 16, 17, Fizz!...

If the number is a multiple of 3, say "Fizz"
If it is a multiple of 5, say "Buzz"
If it is a multiple of 3 and 5, say "FizzBuzz"
Otherwise, just say the number.

It's not hard...

```
public String say(Integer n) {
    if (isFizz(n) && isBuzz(n)) {
        return "FizzBuzz";
    }
    if (isFizz(n)) {
        return "Fizz";
    }
    if (isBuzz(n)) {
        return "Buzz";
    }
    return n.toString();
}

public boolean isFizz(Integer n) {
    return 0 == n % 3;
}

// ...
```

New requirement

If it is a multiple of 7, say "Bang"

No problem!

```
public String say(Integer n) {
    if (isBang(n)) {
        return "Bang";
    }
    if (isFizz(n) && isBuzz(n)) {
        return "FizzBuzz";
    }
    if (isFizz(n)) {
        return "Fizz";
    }
    if (isBuzz(n)) {
        return "Buzz";
    }
    return n.toString();
}
```

Wait, that's not what I meant!

If it is a multiple of 3 and 7, say "FizzBang"
If it is a multiple of 5 and 7, say "BuzzBang"
If it is a multiple of 3, 5 and 7, say "FizzBuzzBang"

Hmmm....

```
public String say(Integer n) {
    if (isFizz(n) && isBuzz(n) && isBang(n)) {
        return "FizzBuzzBang";
    }
    if (isBang(n) && isBuzz(n)) {
        return "BuzzBang";
    }
    if (isBang(n) && isFizz(n)) {
        return "FizzBang";
    }
    if (isBang(n)) {
        return "Bang";
    }
    if (isFizz(n) && isBuzz(n)) {
        return "FizzBuzz";
    }
    if (isFizz(n)) {
        return "Fizz";
    }
    if (isBuzz(n)) {
        return "Buzz";
    }
    return n.toString();
}
```

Hmmm....

```
public String say(Integer n) {
    if (isFizz(n) && isBuzz(n) && isBang(n)) {
        return "FizzBuzzBang";
    }
    if (isBang(n) && isBuzz(n)) {
        return "BuzzBang";
    }
    if (isFizz(n) && isBuzz(n)) {
        return "FizzBuzz";
    }
    if (isFizz(n)) {
        return "Fizz";
    }
    if (isBuzz(n)) {
        return "Buzz";
    }
    return n.toString();
}
```

Not so simple anymore. What is gonna happen when the customer adds a new requirement?

OK. Nobody told you before but...

Adding IFs is evil.

easy ≠ effective

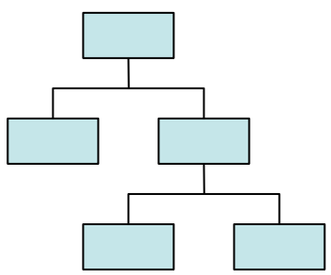
http://pierg.wordpress.com/2009/08/05/anti-if-campaign/

The Open/Closed Principle

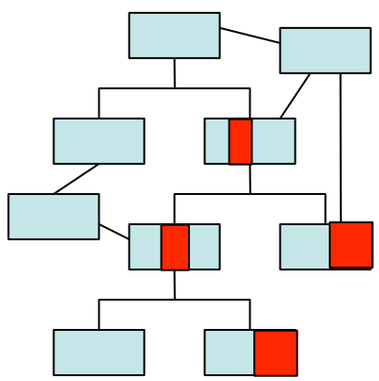
Software entities
(classes, modules, functions, etc.)
should be *open for extension*, but
closed for modification

How do we implement features?

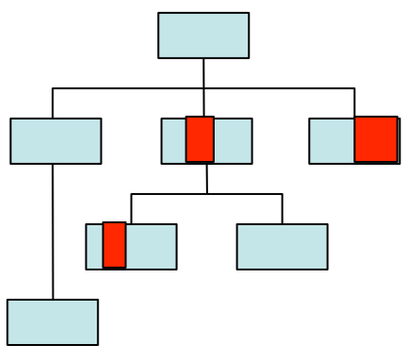
Usual way:



Starting code base

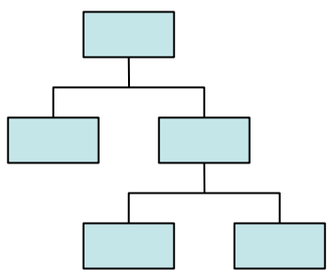


Changes implemented
red == code changed

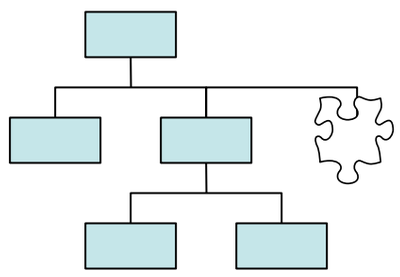


(Hopefully) Code cleaned up

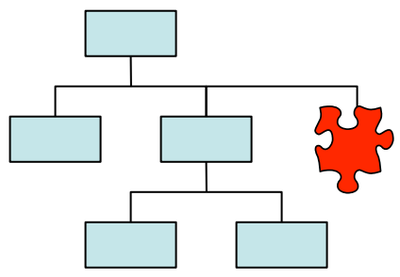
OCP:



Starting code base



Change design to make room for new feature



Implement feature

From a slide by Dave Nicolette

When I must add functionality:

- Can I do it by changing *only construction code* and creating *new classes*?
- If I can, I rock! ➪ €€€€
- If I can't, I *refactor* until I can

Rules for the OCP dojo

1. Write a failing test
2. Write a setup that builds an object (or aggregate) that makes the test pass
   - Factory only creates and links, no conditionals
3. Write next failing test
4. Can you make it pass by changing factory and/or creating new classes?
   - Yes: great! go back to step 3
   - No: refactor until you can

Refactoring should bring the system in a state where it's possible to implement the next test just by composing objects in the setup method

No new functionality! Current test should still fail

First test: Say the number

Just say the number

say(1) returns "1"
say(2) returns "2"

Second test: Say "Fizz"

When a number is a multiple of 3, say "Fizz"

say(3) returns "Fizz"
say(6) returns "Fizz"

Third test: say "Buzz"

When a number is a multiple of 5, say "Buzz"

say(5) returns "Buzz"
say(10) returns "Buzz"

Fourth test: say "FizzBuzz"

When a number is a multiple of
3 and 5, say "FizzBuzz"

say(3*5) returns "FizzBuzz"

Fifth test: say Bang

When a number is a multiple of 7, say "Bang"

say(7) returns "Bang"
say(14) returns "Bang"

Sixth, Seventh, Eighth test:
say FizzBang, BuzzBang, FizzBuzzBang

say(3*7) returns "FizzBang"
say(5*7) returns "BuzzBang"
say(3*5*7) returns "FizzBuzzBang"

The Bowling Score

By Robert Martin "Uncle Bob"

http://butunclebob.com/ArticleS.UncleBob.TheBowlingGameKata

The requirements

- Write class "Game" with two methods:
  - void roll(int pins); call when the player rolls a ball. The argument is the number of pins knocked down.
  - int score(); called when the game is ended. Returns the final score.

http://butunclebob.com/ArticleS.UncleBob.TheBowlingGameKata

The solution

```
int score() {
    int score = 0;
    int currentRoll = 0;
    for (int frame=0; frame<10; frame++) {
        if (isStrike(currentRoll)) {
            score += 10 + sumOfTwoRolls(currentRolls+1);
            currentRoll++;
        } else if (isSpare(currentRoll)) {
            score += 10 + rolls[currentRolls+1];
            currentRoll += 2;
        } else {
            score = sumOfTwoRolls(currentRolls);
            currentRoll += 2;
        }
    }
    return score;
}
```

http://butunclebob.com/ArticleS.UncleBob.TheBowlingGameKata

What happens next?

A new story

To support our customers on the Mars colony, we should implement *Martian Bowling*
This is the same as regular bowling, except for:

- 12 frames
- 3 balls per frame

```
int score() {
    int score = 0;
    int currentRoll = 0;
    for (int frame=0; frame<10; frame++) {
        if (isStrike(currentRoll)) {
            score += 10 + sumOfTwoRolls(currentRolls+1);
            currentRoll++;
        } else if (isSpare(currentRoll)) {
            score += 10 + rolls[currentRolls+1];
            currentRoll += 2;
        } else {
            score = sumOfTwoRolls(currentRolls);
            currentRoll += 2;
        }
    }
    return score;
}
```

```
int score() {
    int score = 0;
    int currentRoll = 0;
    int numFrames = isMartian() ? 12 : 10;
    for (int frame=0; frame<numFrames; frame++) {
        if (isStrike(currentRoll)) {
            score += 10 + sumOfTwoRolls(currentRolls+1);
            currentRoll++;
        } else if (isSpare(currentRoll)) {
            score += 10 + rolls[currentRolls+1];
            currentRoll += 2;
        } else if (isMartian()) {
            score = sumOfThreeRolls(currentRolls);
            currentRoll += 3;
        } else {
            score = sumOfTwoRolls(currentRolls);
            currentRoll += 2;
        }
    }
    return score;
}
```

And another!

The scientists on Callisto play the *Callisto Variant*
This is the same as regular bowling, except for:

- ✳ As long as the last roll is 10, you may keep rolling

This may be played with *either the Terran or Martian rules*

```
int score() {
    int score = 0;
    int currentRoll = 0;
    int numFrames = isMartian() ? 12 : 10;
    for (int frame=0; frame<numFrames; frame++) {
        if (callistoVariant() && isLastFrame(frame)) {
            while (isStrike(currentRoll)) {
                score += 10 + sumOfTwoRolls(currentRolls+1);
                currentRoll++;
            }
        } else {
            if (isStrike(currentRoll)) {
                score += 10 + sumOfTwoRolls(currentRolls+1);
                currentRoll++;
            }
            if (isSpare(currentRoll)) {
                score += 10 + rolls[currentRolls+1];
                currentRoll += 2;
            } else if (isMartian()) {
                score = sumOfThreeRolls(currentRolls);
                currentRoll += 3;
            } else {
                score = sumOfTwoRolls(currentRolls);
                currentRoll += 2;
            }
        }
    return score;
}
```

Meanwhile, on Venus...

... people play Venusian Bowling, where the number of pins is variable. It starts with 1 and increases by one until frame 11

```
int score() {
    int score = 0;
    int currentRoll = 0;
    int numFrames = isMartian() ? 12 : (isVenusian() ? 11 : 10);
    for (int frame=0; frame<numFrames; frame++) {
        if (callistoVariant() && isLastFrame(frame)) {
            while (isStrike(currentRoll, frame)) {
                score += 10 + sumOfTwoRolls(currentRolls+1);
                currentRoll++;
            }
        } else {
            if (isStrike(currentRoll)) {
                score += 10 + sumOfTwoRolls(currentRolls+1);
                currentRoll++;
            }
        }
        if (isSpare(currentRoll)) {
            score += 10 + rolls[currentRolls+1];
            currentRoll += 2;
        } else if (isMartian()) {
            score = sumOfThreeRolls(currentRolls);
            currentRoll += 3;
        } else {
            score = sumOfTwoRolls(currentRolls);
            currentRoll += 2;
        }
    }
    return score;
}
```

```
boolean isStrike(int currentRoll, int frame) {
    if (isVenusian()) {
        return rolls[currentRoll] == frame;
    }
    return rolls[currentRoll] == 10;
}
```

Help!

Another way?

```
int terranScore() {
    int score = 0;
    int currentRoll = 0;
    for (int frame=0; frame<10; frame++) {
        if (isStrike(currentRoll)) {
            score += 10 +
                  sumOfTwoRolls(currentRolls+1);
            currentRoll++;
        } else if (isSpare(currentRo
            score += 10 + rolls[cur
            currentRoll += 2;
        } else {
            score = sumOfTwoRolls(currentRolls);
            currentRoll += 2;
        }
    }
    return score;
}

int martianScoreWithCallistoVariant() {
//     ...
}

int venusianScore() {
//     ...
}
```

```
int martianScore() {
    int score = 0;
    int currentRoll = 0;
    for (int frame=0; frame<12; frame++) {
        if (isStrike(currentRoll)) {
            score += 10 +
                sumOfTwoRolls(currentRolls+1);
            currentRoll++;
        } else if (isSpare(currentRoll)) {
            score += 10 + rolls[currentRolls+1];
            currentRoll += 2;
        } else {
            score = sumOfThreeRolls(currentRolls);
            currentRoll += 3;
        }
    }
    return score;
}

int terranScoreWithCallistoVariant() {
//     ...
}
```

Duplication!!!

The challenge

Can we implement ***all*** the various scoring rules *with no IFs* and *without duplication*?

The Bowling Score stories

Sum of rolls

When the player does not strike or spare, the score is the sum of the two rolls.

Sum of rolls

Acceptance Criteria

scenario 0 - all zeroes.
  Player rolls 0 for 20 times.
  The application reports score is 0.

scenario 1 - all twos.
  Player rolls 2 for 20 times. The application reports score is 40.

scenario 2 - up and down.
  Player rolls 0,1,2,3,4,5,4,3,2,1,0,1,2,3,4,5,4,3,2,1. The application reports score is 50.

Spare

When the players knocks down all pins in two rolls, the score for that frame is 10 plus the next roll.

Spare

Acceptance Criteria

scenario 0 - one spare.
  Player rolls 3, 7, 4 and then rolls 0 for 17 times.
  The application reports score is 10 + 4 + 4.

scenario 1 - spare in the last frame.
  Player rolls 0 for 18 times, then 2, 8, 3. The application reports score is 10 + 3.

Strike

When the players knocks down all pins in one roll, the score for that frame is 10 plus the next two rolls.

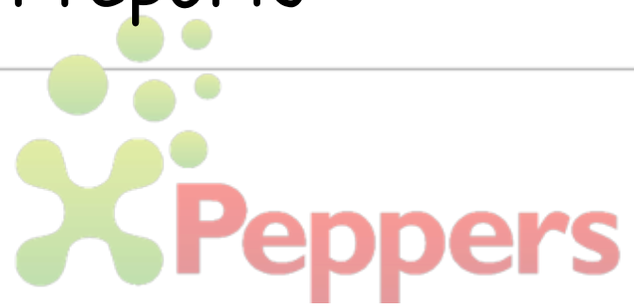
Strike

Acceptance Criteria

scenario 0 - one strike.
  Player rolls 10, 2, 4 and then rolls 0 for 16 times.
  The application reports score is 10 + 6 + 6.

scenario 1 - strike in the last frame.
  Player rolls 0 for 18 times, then 10, 8, 3. The application reports score is 10 + 11.

scenario 2 - perfect game
  Player rolls 10 for 12 times. The application reports score is 300.



A new story

To support our customers on the Mars colony, we should implement *Martian Bowling*
This is the same as regular bowling, except for:

- 12 frames
- 3 balls per frame

Martian Bowling

When playing Martian bowling, there are 3 balls per frame, and 12 frames.

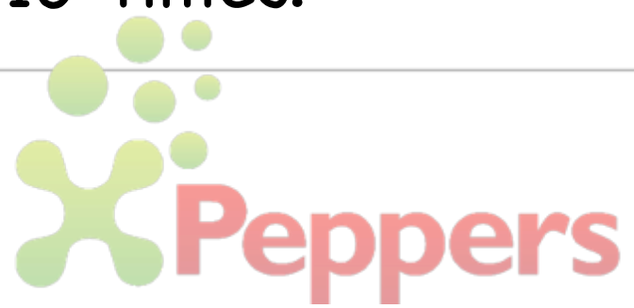
Martian Bowling

Acceptance Criteria

scenario 0 - sum of three rolls.
  Player rolls 1, 2, 3 and then rolls 0 for 3 * 11 times.
  The application reports score is 1 + 2 + 3.

scenario 1 - martian spare.
  Player rolls 1, 2, 7, 3, then 0 for 2 + 3*10 times. The application reports score is 10 + 3 + 3.

scenario 2 - martian strike.
  Player rolls 10, then 2, 3, then 0 for 1 + 3*10 times.
The application reports score is 10 + 5 + 5.



And another!

The scientists on Callisto play the *Callisto Variant*
This is the same as regular bowling, except for:

- ✳ As long as the last roll is 10, you may keep rolling

This may be played with *either the Terran or Martian rules*

Callisto Variant

As long as the last roll is 10, you may keep rolling

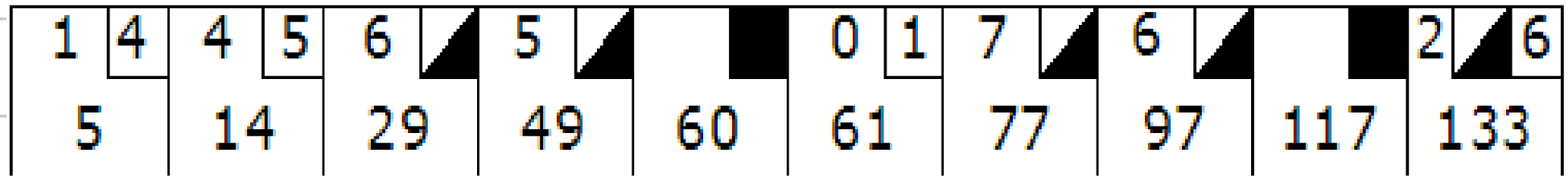
Callisto Variant

Acceptance Criteria

Scenario 0 - Terran + Callisto.
Player rolls 0 for 2*9 times, then 10 for 5 times.
The application reports score is 10*5.

Scenario 1 - Martian + Callisto.
Player rolls 0 for 3*11 times, then 10 for 7 times.
The application reports score is 10*7.

Table display



The application displays a table with results for each frame

Table display

Acceptance Criteria

The player rolls

1, 4, 4, 5, 6, 4, 5, 5, 10, 0, 1, 7, 3, 6, 4, 10, 2, 8, 6

The application reports score is

```
|1 4|4 5|6 /|5 /|  X|0 1|7 /|6 /|  X|2/6|
|  5| 14| 29| 49| 60| 61| 77| 97|117|133|
```

Things to remember

- Before starting to code, refactor to make implementing the feature easier
- Before refactoring, think and plan
- Always refactor on a green bar
- If you mess up, ctrl-Z until back to green (whew!)
- Only add an extension point when a new feature requires it

Want to know more?

http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod

http://www.antiifcampaign.com/

http://matteo.vaccari.name/blog/archives/293

This presentation can be downloaded from
http://slideshare.net/xpmatteo

Grazie dell'attenzione!

Peppers

Extreme Programming: development & mentoring