Clean Code

Homework-Week4

Aluno:
Francisco Ribeiro - PG27719

Introdução

Proposto pela equipa docente da unidade curricular *Engenharia Web* este trabalho tem como objectivo melhorar a qualidade do código escrito no homework-week3.

Arquitetura

Diagrama de classes presente no homework-week3

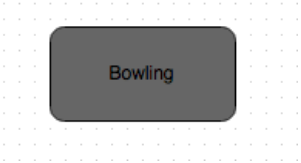
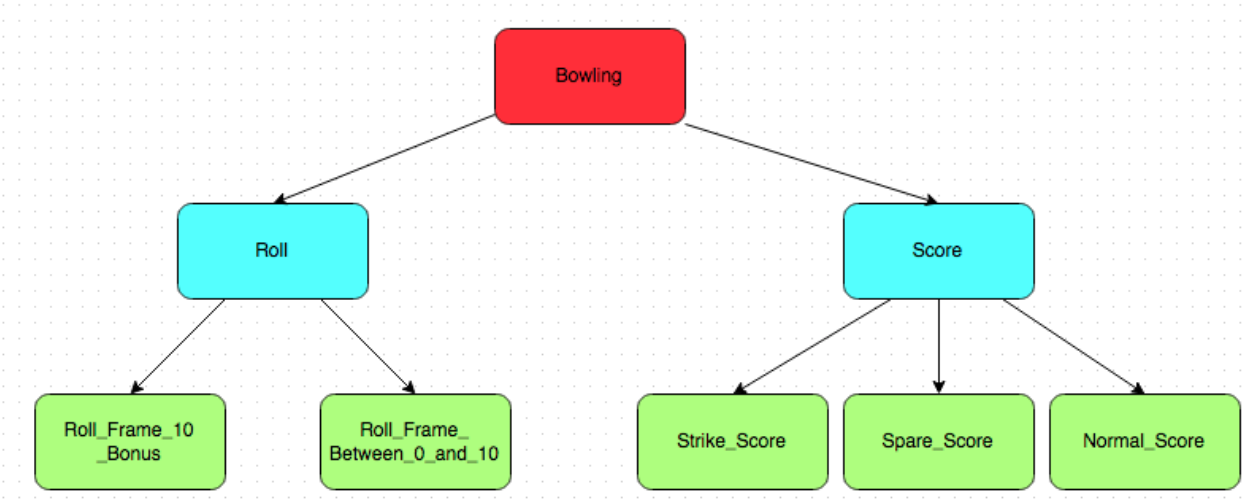
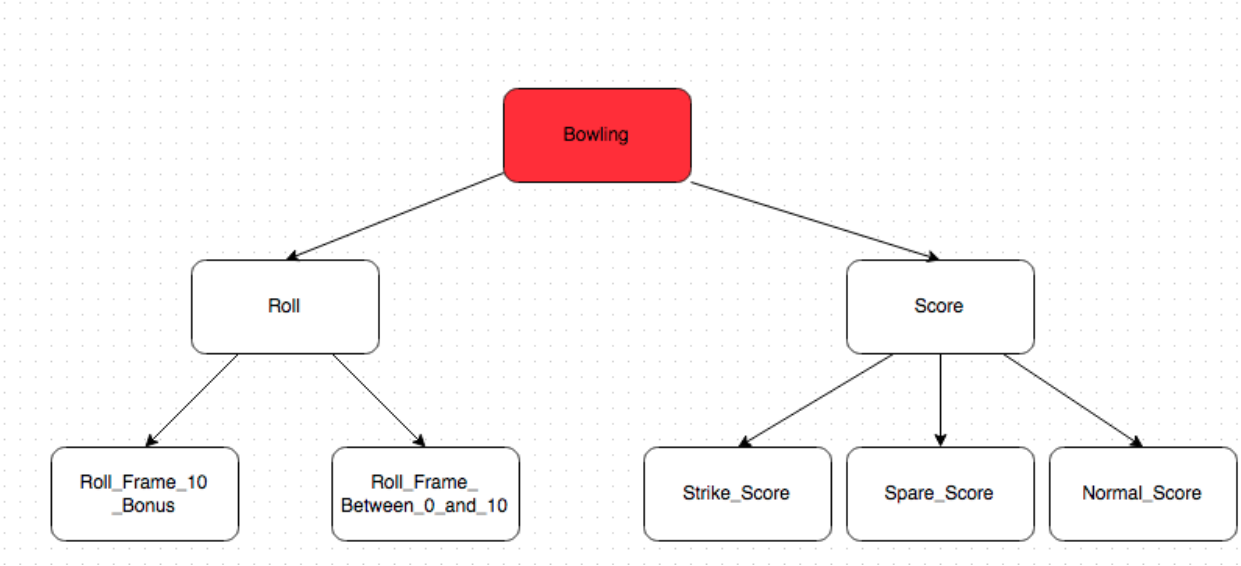


Diagrama de classes presente no homework-week4



Descrição

Class Bowling



- **Responsabilidade:**

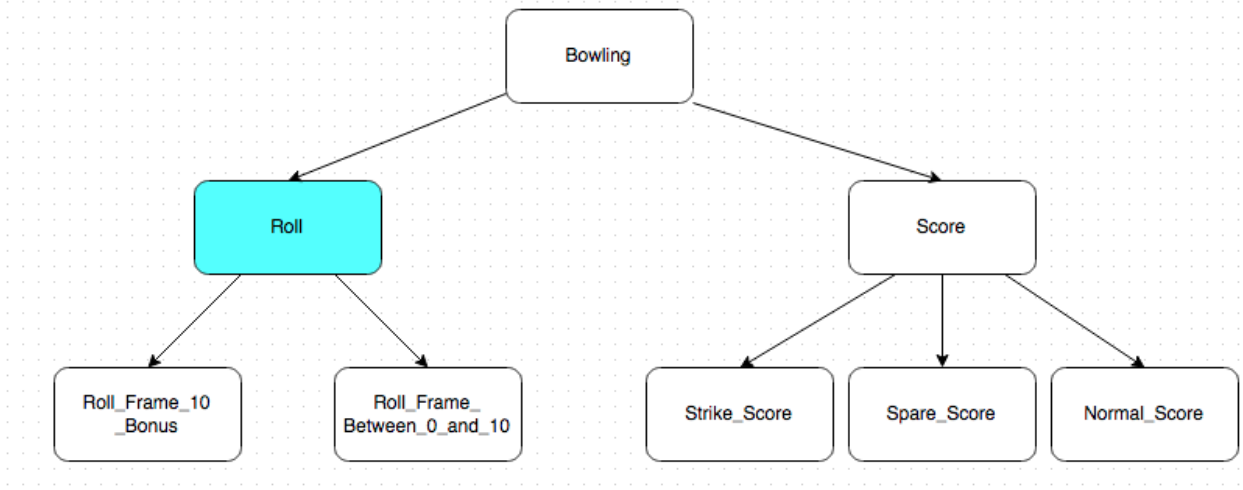
A classe Bowling é a classe principal do programa.
Aqui são guardadas todas as jogadas no *array "game".*
A classe bowling distribui a responsabilidade de introduzir jogadas no *array* à classe *Roll e distribui a responsabilidade de calcular a pontuação do jogo à classe Score.*
É utilizado o padrão Decorator.

- **Código**

bowling.rb

```ruby
load 'roll.rb'
load 'score.rb'

class Bowling

  attr_accessor :game

  def initialize()
    @game = Array.new
    @r = Roll.new(self)
    @s = Score.new(self)
  end

  def roll(pins)
    @r.roll(pins)
  end

  def score
    @s.score
  end
end

```

Class Roll



- **Responsabilidade:**

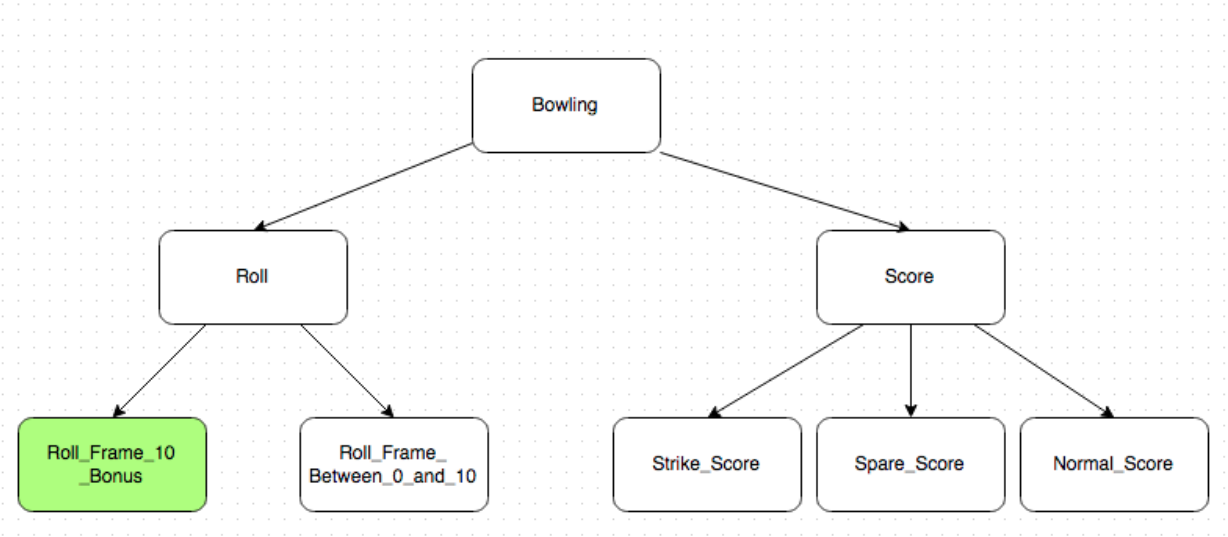
A classe Roll tem como responsabilidade verificar se a jogada é valida e, se sim, determinar se essa jogada é um strike, um spare, ou uma jogada normal (não é strike nem spare).
A classe Roll envia para as classes Roll_Frame_10_Bonus (caso de bónus - decima frame) e Roll_Frame_Between_0_and_10 a responsabilidade de adicionar essa jogada válida à lista de jogadas.

- **Código**

roll.rb

```ruby
load 'roll_frame_10_bonus.rb'
load 'roll_frame_between_0_and_10.rb'


class Roll
  attr_accessor :bowling
  attr_accessor :game
  attr_accessor :first_roll
  attr_accessor :second_roll
  attr_accessor :frame          # number of frames
  attr_accessor :valid_roll     #is a valid roll ??  (if no then -1)

  def initialize(bowling)
    @bowling=bowling
    @game = bowling.game
    @first_roll = -1
    @second_roll = 0
    @frame = 0
    @valid_roll= 0
  end

  def roll_verification(pins)
    if (@valid_roll != -1)
    then
      @valid_roll = 0
      frame_10_is_Strike_case(pins)
      frame_10_is_Spare_case(pins)
      frame_between_0_and_10_case(pins)
      is_valid_roll?
    end
  end
```

Class Roll_Frame_10_Bonus



- **Responsabilidade:**

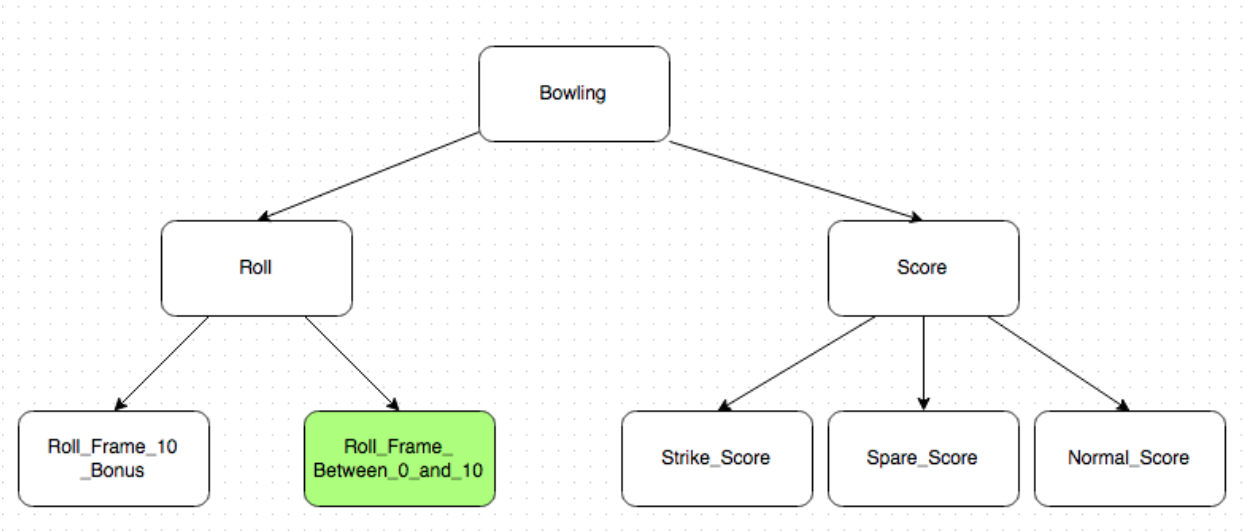
A classe Roll_Frame_10_Bonus tem a responsabilidade de adicionar à lista de jogadas uma jogada de Bonus na frame 10.

- **Código**

roll_frame_10_bonus.rb

```ruby
class Roll_Frame_10_Bonus

  def initialize(roll)
    @roll = roll
  end

  def add_bonus(pins)
    if (pins==10)
    then
      @roll.game << 'strike'
    else
      @roll.game << pins
    end
  end

end
```

Class Roll_Frame_Between_0_and_10



- **Responsabilidade:**

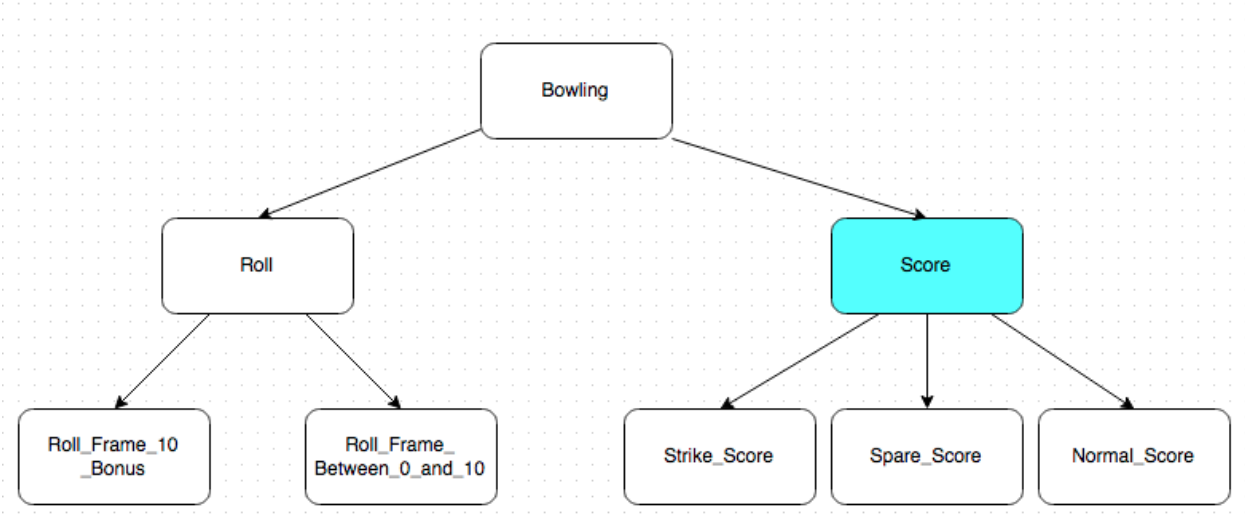
A classe Roll_Frame_Between_0_and_10 tem como responsabilidade adicionar uma jogada, entre as frames 0 e 10, à lista de jogadas.

- **Código**

roll_frame_between_0_and_10.rb

```
1  class Roll_Frame_Between_0_and_10
2
3    def initialize(roll)
4      @roll = roll
5    end
6
7    def add_roll(pins)
8      first_roll_is_strike(pins)
9      first_roll_is_not_strike(pins)
10     second_roll_is_spare(pins)
11     second_roll_is_not_spare(pins)
12     invalid_roll(pins)
13     second_roll_increment(pins)
14   end
15
```

Class Score:



- **Responsabilidade:**

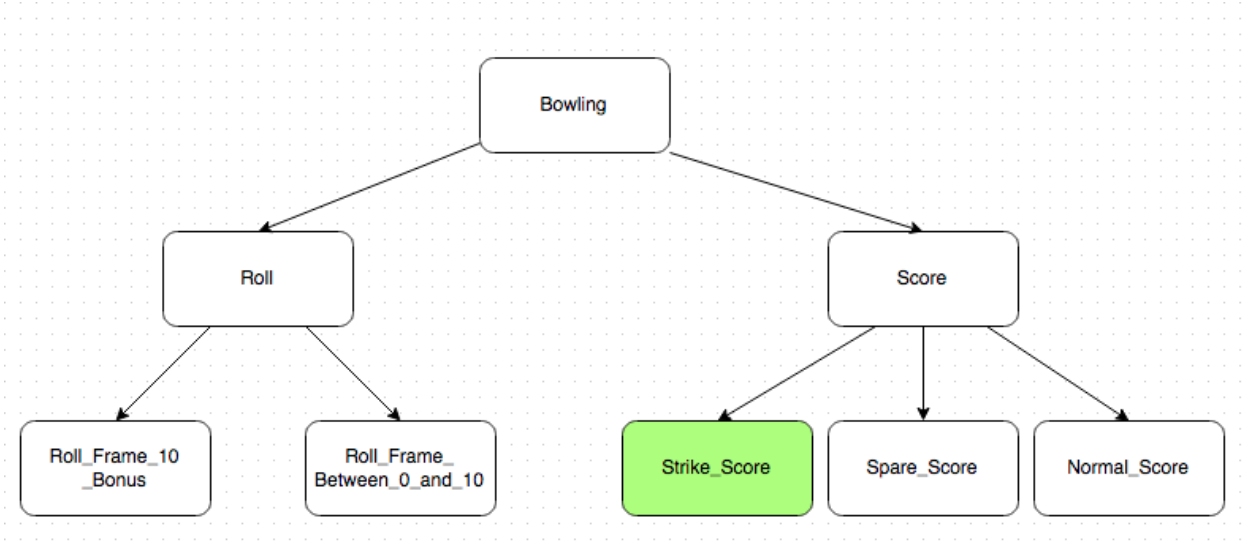
A classe Score tem como responsabilidade percorrer a lista de jogadas e verificar se a jogada é um strike, um spare ou uma jogada normal. Para cada um destes casos envia a responsabilidade de calcular o score para as classes Strike_Score, Spare_Score e Normal_Score.

- **Código**

score.rb

```ruby
load 'strike_score.rb'
load 'spare_score.rb'
load 'normal_score.rb'


class Score

  attr_accessor :game
  attr_accessor :total
  attr_accessor :position
  attr_accessor :frame
  attr_accessor :is_valid

  def initialize(bowling)
    @bowling=bowling
    @game=bowling.game
    @position=0       #Array position
    @total=0          #total score
    @frame = 1        #frame number
    @is_valid = 0     #is a valid count??  (if no then -1)
  end

  def score_verification
    @bowling.game.each do |roll|
      is_strike?(roll)
      is_spare?(roll)
      is_normal_point_score?(roll)
      invalid_score(roll)
      @is_valid += 1;
    end
    @total
  end

```

Class Strike_Score



- **Responsabilidade:**

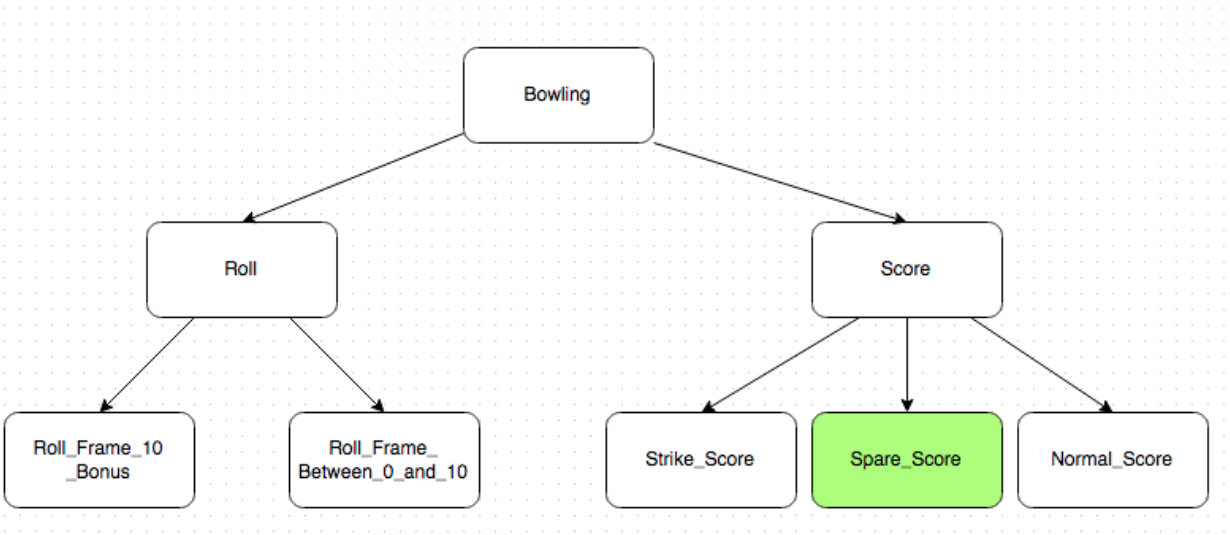
A responsabilidade da classe Strike_Score é calcular o score de uma jogada strike.

- **Código**

strike_score.rb

```ruby
class Strike_Score
  def initialize(score)
    @score=score
  end

  def strike_score
    if @score.game[@score.position+1]!= nil and @score.game[@score.position+2]!=nil and @score.frame<10
    then
      @score.total += 10 + string_to_score(@score.position+1) + string_to_score(@score.position+2)
      @score.is_valid=0
    else
      @score.total += 10
      @score.is_valid=0
    end
  end

```

Class Spare_Score



- **Responsabilidade:**

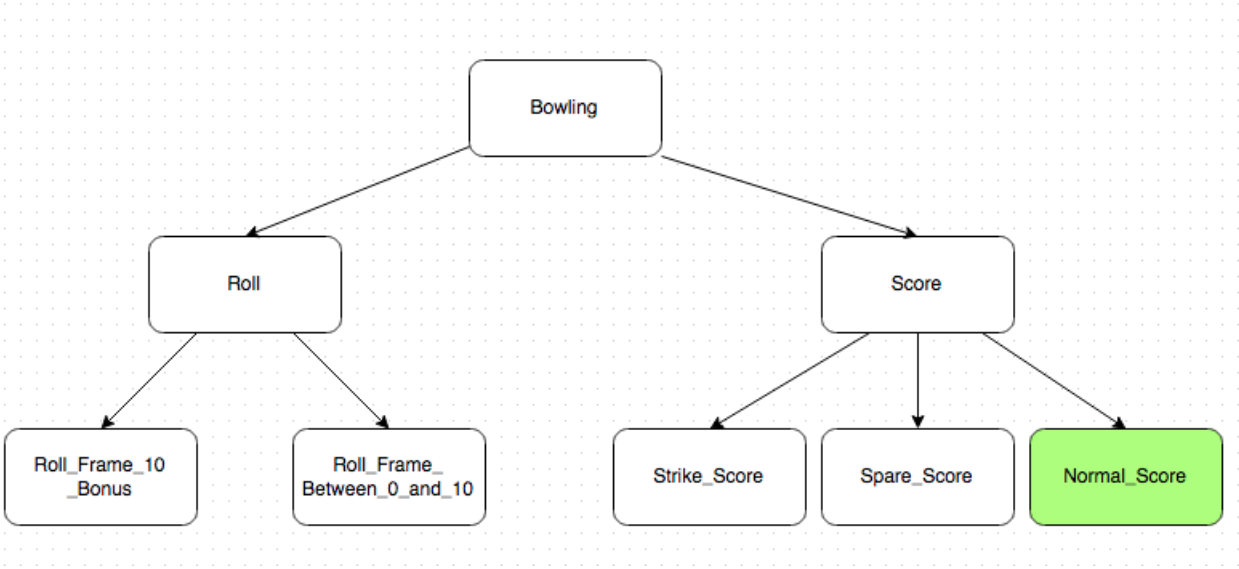
A responsabilidade da classe Spare_Score é calcular o score de uma jogada spare.

- **Código**

spare_score.rb

```ruby
class Spare_Score
  def initialize(score)
    @score=score
  end

  def spare_score
    if @score.game[@score.position+1]!=nil and @score.frame<10
    then
      @score.total += (10-@score.game[@score.position-1]) + string_to_score(@score.position+1)
      @score.is_valid=0
    else
      @score.total+=(10-@score.game[@score.position-1])
      @score.is_valid=0
    end
  end

```

Class Normal_Score



- **Responsabilidade:**

A responsabilidade da classe Normal_Score é calcular o score de uma jogada normal, ou seja, nao é strike nem spare.

- **Código**

normal_score.rb

```ruby
class Normal_Score
  def initialize (score)
    @score=score
  end

  def normal_score(roll)
    @score.total += roll
  end

end
```

Clean Code

Formatação

- Foram removidos quase todos os comentários do código.
- Foram substituídas *TABS* por dois espaços.
- Foram removidos os espaços antes e depois dos símbolos (, [,) e].
- Foram corrigidos os erros ortográficos.
- Não existem classes com muitos metódos.
- Não existem métodos com muita complexidade.
- Não são usados métodos com o mesmo nome da classe.

Nomes

- Foram usados nomes adequados para variáveis e métodos.

Template method

- Foram divididos métodos com grande complexidade em “submétodos” para tornar o código mais legível e com menos complexidade.

Single responsibility

- Uma classe tem apenas uma responsabilidade, uma razão para mudar.
- Tornou-se mais fácil a reutilização de código.

Padrão *Decorator*

- Em quase todo o programa foi utilizado o padrão *Decorator*. Ele permite dividir a responsabilidade entre as diferentes classes.

Open closed

- Foi estendido o comportamento de uma classe sem ser alterado o código.(Injeção de dependência).
- Exemplo: A classe Roll_Frame_10_Bonus pode ser utilizada quando o bónus é referente a um strike na décima frame ou quando o bónus é um spare no décimo frame.

roll.rb

```ruby
def frame_10_is_Strike_case(pins)
  if pins<=10 and pins>=0 and @frame>=10 and @frame<12 and (@game.last=='strike' or (@game.take(@game.size-1)).last=='strike')
  then
    @valid_roll =1
    frame_10_strike_bonus = Roll_Frame_10_Bonus.new(self)
    frame_10_strike_bonus.add_bonus(pins)
    @frame += 1
  end
end

def frame_10_is_Spare_case(pins)
  if pins<=10 and pins>=0 and @frame==10 and @game.last=='spare'
  then
    @valid_roll =1
    frame_10_spare_bonus = Roll_Frame_10_Bonus.new(self)
    frame_10_spare_bonus.add_bonus(pins)
    @frame += 1
  end
end
```

roll_frame_10_bonus.rb

```ruby
class Roll_Frame_10_Bonus

  def initialize (roll)
    @roll = roll
  end

  def add_bonus(pins)
    if (pins==10)
    then
      @roll.game << 'strike'
    else
      @roll.game << pins
    end
  end

end
```

Law of demeter

- Tendo A->B->C. “A” não sabe nada sobre “C” atraves de B.