**Ревью кода.**

Общий обзор

Код представляет собой реализацию игры в теннис с использованием трех классов: `TennisGameDefactored1`, `TennisGameDefactored2`, и `TennisGameDefactored3`. Каждый класс реализует методы для отслеживания очков и определения текущего состояния игры. Однако код имеет несколько проблем, которые могут затруднить его поддержку, тестирование и понимание.

**1. Нарушение стандартов кодирования**

Имена переменных и методов не соответствуют стандартам PEP 8. Например:

- Методы `P1Score` и `P2Score` должны быть названы в нижнем регистре с подчеркиванием, например, `p1\_score` и `p2\_score`.

- Переменные, такие как `player1Name`, `player2Name`, `p1points`, и `p2points`, также не соответствуют стандартам.

Решение:

Переименуйте переменные и методы для соблюдения стандартов PEP 8:

python

**class TennisGame:**

**def \_\_init\_\_(self, player1\_name, player2\_name):**

**self.player1\_name = player1\_name**

**self.player2\_name = player2\_name**

**self.p1\_points = 0**

**self.p2\_points = 0**

**2. Неоправданная цикломатическая сложность**

Метод `score` в классе `TennisGameDefactored2` имеет высокую цикломатическую сложность из-за множества вложенных условий и ветвлений. Это делает метод трудным для понимания и тестирования.

Решение:

Разделите метод `score` на несколько вспомогательных методов, чтобы уменьшить его сложность. Например, можно выделить обработку равного счета, преимущества и выигрыша в отдельные методы.

python

**def \_get\_equal\_score(self):**

**# Обработка равного счета**

**...**

**def \_get\_winner(self):**

**# Определение победителя**

**...**

**def score(self):**

**if self.p1\_points == self.p2\_points:**

**return self.\_get\_equal\_score()**

**winner = self.\_get\_winner()**

**if winner:**

**return winner**

**...**

**3. Нарушение принципа DRY (Don't Repeat Yourself)**

В методах `score` различных классов повторяются одни и те же логические блоки, что приводит к избыточности. Например, определение текстового представления очков для каждого игрока.

Решение:

Создайте вспомогательный метод для получения текстового представления очков, чтобы избежать дублирования кода:

python

**def \_get\_score\_description(self, points):**

**score\_map = {0: "Love", 1: "Fifteen", 2: "Thirty", 3: "Forty"}**

**return score\_map.get(points, "Unknown")**

**4. Мертвый код**

Методы `SetP1Score` и `SetP2Score` в классе `TennisGameDefactored2` не используются, что делает их избыточными.

Решение:

Удалите неиспользуемые методы, чтобы уменьшить количество строк кода и улучшить его читаемость:

python

Удалите эти методы, если они не используются

**def SetP1Score(self, number):**

**...**

**def SetP2Score(self, number):**

**...**

**5. Комбинаторный взрыв**

В классе `TennisGameDefactored2` большое количество условий для определения результата игры (победа, преимущество и т.д.) делает код сложным для понимания и увеличивает количество возможных комбинаций.

Решение:

Упрощение логики определения результата игры может помочь. Например, можно использовать более простую логику для определения выигрыша и преимущества:

python

**def \_get\_winner(self):**

**if self.p1\_points >= 4 and (self.p1\_points - self.p2\_points) >= 2:**

**return f"Win for {self.player1\_name}"**

**if self.p2\_points >= 4 and (self.p2\_points - self.p1\_points) >= 2:**

**return f"Win for {self.player2\_name}"**

**if self.p1\_points > self.p2\_points and self.p2\_points >= 3:**

**return f"Advantage {self.player1\_name}"**

**if self.p2\_points > self.p1\_points and self.p1\_points >= 3:**

**return f"Advantage {self.player2\_name}"**

**return None**

Лист 1: Исходный код

# -\*- coding: utf-8 -\*-

class TennisGameDefactored1:

def \_\_init\_\_(self, player1Name, player2Name):

self.player1Name = player1Name

self.player2Name = player2Name

self.p1points = 0

self.p2points = 0

def won\_point(self, playerName):

if playerName == self.player1Name:

self.p1points += 1

else:

self.p2points += 1

def score(self):

result = ""

tempScore=0

if (self.p1points==self.p2points):

result = {

0 : "Love-All",

1 : "Fifteen-All",

2 : "Thirty-All",

3 : "Forty-All",

}.get(self.p1points, "Deuce")

elif (self.p1points>=4 or self.p2points>=4):

minusResult = self.p1points-self.p2points

if (minusResult==1):

result ="Advantage " + self.player1Name

elif (minusResult ==-1):

result ="Advantage " + self.player2Name

elif (minusResult>=2):

result = "Win for " + self.player1Name

else:

result ="Win for " + self.player2Name

else:

for i in range(1,3):

if (i==1):

tempScore = self.p1points

else:

result+="-"

tempScore = self.p2points

result += {

0 : "Love",

1 : "Fifteen",

2 : "Thirty",

3 : "Forty",

}[tempScore]

return result

class TennisGameDefactored2:

def \_\_init\_\_(self, player1Name, player2Name):

self.player1Name = player1Name

self.player2Name = player2Name

self.p1points = 0

self.p2points = 0

def won\_point(self, playerName):

if playerName == self.player1Name:

self.P1Score()

else:

self.P2Score()

def score(self):

result = ""

if (self.p1points == self.p2points and self.p1points < 4):

if (self.p1points==0):

result = "Love"

if (self.p1points==1):

result = "Fifteen"

if (self.p1points==2):

result = "Thirty"

if (self.p1points==3):

result = "Forty"

result += "-All"

if (self.p1points==self.p2points and self.p1points>3):

result = "Deuce"

P1res = ""

P2res = ""

if (self.p1points > 0 and self.p2points==0):

if (self.p1points==1):

P1res = "Fifteen"

if (self.p1points==2):

P1res = "Thirty"

if (self.p1points==3):

P1res = "Forty"

P2res = "Love"

result = P1res + "-" + P2res

if (self.p2points > 0 and self.p1points==0):

if (self.p2points==1):

P2res = "Fifteen"

if (self.p2points==2):

P2res = "Thirty"

if (self.p2points==3):

P2res = "Forty"

P1res = "Love"

result = P1res + "-" + P2res

if (self.p1points>self.p2points and self.p1points < 4):

if (self.p1points==2):

P1res="Thirty"

if (self.p1points==3):

P1res="Forty"

if (self.p2points==1):

P2res="Fifteen"

if (self.p2points==2):

P2res="Thirty"

result = P1res + "-" + P2res

if (self.p2points>self.p1points and self.p2points < 4):

if (self.p2points==2):

P2res="Thirty"

if (self.p2points==3):

P2res="Forty"

if (self.p1points==1):

P1res="Fifteen"

if (self.p1points==2):

P1res="Thirty"

result = P1res + "-" + P2res

if (self.p1points > self.p2points and self.p2points >= 3):

result = "Advantage " + self.player1Name

if (self.p2points > self.p1points and self.p1points >= 3):

result = "Advantage " + self.player2Name

if (self.p1points>=4 and self.p2points>=0 and (self.p1points-self.p2points)>=2):

result = "Win for " + self.player1Name

if (self.p2points>=4 and self.p1points>=0 and (self.p2points-self.p1points)>=2):

result = "Win for " + self.player2Name

return result

def SetP1Score(self, number):

for i in range(number):

self.P1Score()

def SetP2Score(self, number):

for i in range(number):

self.P2Score()

def P1Score(self):

self.p1points +=1

def P2Score(self):

self.p2points +=1

class TennisGameDefactored3:

def \_\_init\_\_(self, player1Name, player2Name):

self.p1N = player1Name

self.p2N = player2Name

self.p1 = 0

self.p2 = 0

def won\_point(self, n):

if n == self.p1N:

self.p1 += 1

else:

self.p2 += 1

def score(self):

if (self.p1 < 4 and self.p2 < 4):

p = ["Love", "Fifteen", "Thirty", "Forty"]

s = p[self.p1]

return s + "-All" if (self.p1 == self.p2) else s + "-" + p[self.p2]

else:

if (self.p1 == self.p2):

return "Deuce"

s = self.p1N if self.p1 > self.p2 else self.p2N

return "Advantage " + s if ((self.p1-self.p2)\*(self.p1-self.p2) == 1) else "Win for " + s

# NOTE: You must change this to point at the one of the three examples that you're working on!

TennisGame = TennisGameDefactored1

Лист 2: Улучшение класса `TennisGameDefactored1`:

python

class TennisGame:

def \_\_init\_\_(self, player1\_name, player2\_name):

self.player1\_name = player1\_name

self.player2\_name = player2\_name

self.p1\_points = 0

self.p2\_points = 0

def won\_point(self, player\_name):

if player\_name == self.player1\_name:

self.p1\_points += 1

else:

self.p2\_points += 1

def \_get\_score\_description(self, points):

score\_map = {0: "Love", 1: "Fifteen", 2: "Thirty", 3: "Forty"}

return score\_map.get(points, "Unknown")

def \_get\_equal\_score(self):

if self.p1\_points < 4:

return f"{self.\_get\_score\_description(self.p1\_points)}-All"

return "Deuce"

def \_get\_winner(self):

if self.p1\_points >= 4 and (self.p1\_points - self.p2\_points) >= 2:

return f"Win for {self.player1\_name}"

if self.p2\_points >= 4 and (self.p2\_points - self.p1\_points) >= 2:

return f"Win for {self.player2\_name}"

return None

def score(self):

if self.p1\_points == self.p2\_points:

return self.\_get\_equal\_score()

winner = self.\_get\_winner()

if winner:

return winner

score\_description = f"{self.\_get\_score\_description(self.p1\_points)}-{self.\_get\_score\_description(self.p2\_points)}"

return score\_description