A screenshot of a computer

Description automatically generated with medium confidence

Here as you can see, I didn’t add the value attribute, because when introducing an instance we do npt give the value as a para,eter cause the value is something that is generated randomly.

You see how the instructor has introduced it by giving the input parameter a default value “None”.

Secondly, instead of putting the vale attribute equal to the roll\_dice methode, it puts it equal to the value param,eter. So here she do not give the value parameter its value inside the class itself, but it simply introduces the method and inside the method returns the new valuem but does not put this new\_vale equal to the attribute itself.

I also have an extra method.

A screenshot of a computer

Description automatically generated with medium confidence

Here, I have two differences, firstly is that I have attributes that are public, so I should have defined getters, and secondly is the methods, that I didn’t devised any methods for incrementing or decrementing, instead I directly defined them inside the Die\_Game class.

Text

Description automatically generated

Here, the the first part of the classes are generally the same, except for me having a redundant if condition. If not val, is always true, so whether we put the if or not it always happens. So I must had omit it!

A screenshot of a computer

Description automatically generated with medium confidence

In the second part of the DieGame class everything is the same, except the teacher has defined another method for showing the dice, in this case the code is much more readable.

Text

Description automatically generated

Here again the instructor has introduced many new methods to make the code more concise and readable.

First, she introduced method to show the value of the dice. I did it simply by writing the code directly. The logic is the same with the both of us.

def show\_dice(self, player\_value, computer\_value):  
 print(f"Your die: {player\_value}")  
 print(f"Computer die: {computer\_value}\n")

Second, she updated the code by introducing a new method, which has another new method **from the player class** in it. I don’t have these methods inside the player class.

def update\_counters(self, winner, loser): #new method  
 winner.decrement\_counter()#these two methods are coming from the player class  
 loser.increment\_counter()

the other last thing is that I have a **repetition** in my code, which is the same for the three if clauses, which could be avoided, by bring it out the if clauses.

print(f"player 1 counter is: {self.\_player\_one.counter}")  
print(f"player 2 counter is : {self.\_player\_two.counter}")

in this case also the instructor has defined a new method for it:

def show\_counters(self):  
 print(f"\nYour counter: {self.player.counter}")  
 print(f"Computer counter: {self.computer.counter}")

Text

Description automatically generated

In the last part, the game over method varies significantly from mine. She has a new method which is called in the game over method. In this new method, if the winner is the computer, (so winner.is\_computer == True) the corresponding message would be called. While if not, the message related to the player would be called. The reason that I do not have these methods are that I have introduced the player number two as a general player and not the computer.