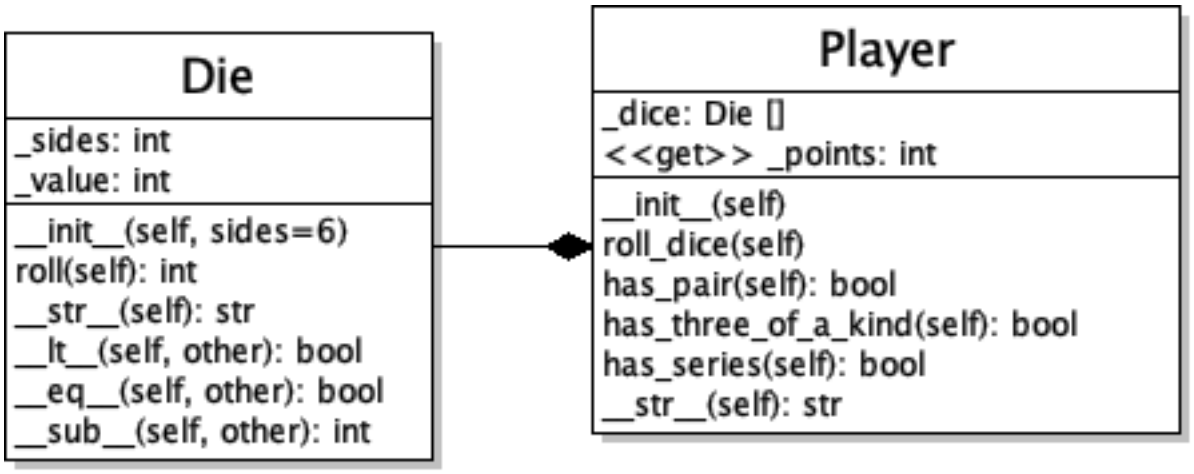
CECS 277 – Lab 6 – Class Relationships

Yahtzee

Create a dice game that awards the user points for a pair, three-of-a-kind, or a series. Use the following class diagram for your program:



Die class (die.py) – has two attributes: the number of sides of the die and the value of the rolled die.

1. `__init__(self, sides=6)` – passes in the number of sides of the die. Assign sides using the parameter and set value to either 0 or to the returned value of roll().
2. `roll(self)` – generate a random number between 1 and the number of sides and assign it to value. Return value.
3. `__str__(self)` – return the Die's value as a string.
4. `__lt__(self, other)` – return true if the value of self is less than the value of other.
5. `__eq__(self, other)` – return true if the value of self is equal to the value of other.
6. `__sub__(self, other)` – return the difference between the value of self and the value of other.

Player class (player.py) – has two attributes: a list of 3 Die objects and the player's points.

1. `__init__(self)` – constructs and sorts the list of three Die objects and then initializes the player's points to 0.
2. `points(self)` – get property that returns the player's points.
3. `roll_dice(self)` – calls roll on each of the Die objects in the list and then sorts the list.
4. `has_pair(self)` – returns true if two dice in the list have the same value (do not access the die's value attribute directly, use ==). Increments points by 1.
5. `has_three_of_a_kind(self)` – returns true if all three dice in the list have the same value (use ==). Increments points by 3.
6. `has_series(self)` – returns true if the values of each of the dice in the list are in a sequence (ex. 1,2,3 or 2,3,4 or 3,4,5 or 4,5,6) (use -). Increments points by 2.
7. `__str__(self)` – returns a string in the format: "D1=2, D2=4, D3=6" (use str()).

Main file (main.py) – has one function named take_turn that passes in a Player object. The take_turn function should: roll the player's dice, display the dice, check for and display any win types (pair, series, three-of-a-kind), and display the updated score. The main function should construct a player object, and then repeatedly call take_turn until the user chooses to end the game. Display the final points at the end of the game. Use the check_input module's get_yes_no function to prompt the user to continue or end the game. Use docstrings to document each class, method, and function.

Example Output (user input in italics)**:**

```
-Yahtzee-

D1=1 D2=4 D3=5
Aww.  Too Bad.
Score = 0
Play again? (Y/N): g
Invalid input - should be a 'Yes'
or 'No'.
Play again? (Y/N): y

D1=3 D2=3 D3=5
You got a pair!
Score = 1
Play again? (Y/N): y

D1=3 D2=4 D3=5
You got a series of 3!
Score = 3
Play again? (Y/N): y

D1=1 D2=1 D3=1
You got 3 of a kind!
Score = 6
Play again? (Y/N): n

Game Over.
Final Score = 6
```

Notes:

1. You should have 4 different files: die.py, player.py, check_input.py, and main.py.
2. Check all user input using the get_yes_no function in the check_input module.
3. Do not create any extra methods or add any extra parameters.
4. Please do not create any global variables or use the attributes globally. Only access the attributes using the class's methods (even in the Player class).
5. The attributes should have a leading underscore to denote that they shouldn't be accessed from outside of their classes.
6. Do not call any of the methods using the double underscores (ex. use == not __eq__).
7. Use docstrings to document the class, each of its methods, and the functions in the main file. See the lecture notes and the Coding Standards reference document for examples.
8. Place your names, date, and a brief description of the program in a comment block at the top of your main file. Place brief comments throughout your code.
9. Thoroughly test your program before submitting:
    a. Make sure that the user input is validated.
    b. Make sure that the dice values are sorted.
    c. Make sure that each of the win types are detected correctly.
    d. Make sure that the user is not awarded for both a pair and a three-of-a-kind simultaneously.
    e. Make sure that each of the win types awards the correct number of points.
    f. Make sure that the game continues and ends correctly.
    g. Make sure that the final score is displayed at the end.

Yahtzee Rubric – Time estimate: 3 hours

Yahtzee **10 points**	Correct. 2 points	A minor mistake. 1.5 points	A few mistakes. 1 point	Several mistakes. 0.5 points	No attempt. 0 points
Die class (in a separate file): 1. Has attributes _sides and _value. 2. Has methods: __init__, roll, __eq__, __lt__, __sub__, and __str__. 3. lt, eq, and sub use the dice values. 4. Did not add a get_value method. 5. Does not access attributes from outside of the class (only use methods).					
Player class (in a separate file): 1. Has attributes: _points and _dice list. 2. Has methods: __init__, point, has_pair, has_three_of_a_kind, has_series, roll_dice, and __str__. 3. init creates 3 dice in a list, sorts them, and then initializes points. 4. roll sorts after rolling the dice. 5. __str__ calls Die's str(). 6. Does not access attributes from outside of the class (only use methods).					
Player class detect win methods: 1. pair and 3-kind use == (not __eq__). 2. series uses – (not __sub__). 3. Returns True/False. 4. Adds the correct number of points.					
Main File: 1. take_turn function rolls and displays the dice, checks and displays if a win condition occurred, and displays points. 2. Does not award points for both 3 of a kind and a pair. 3. main function constructs a Player. 4. main has a loop that repeats until user chooses to quit. 5. main function calls take_turn. 6. main displays final score.					
Code Formatting: 1. Code is in functions/methods. 2. Correct spacing. 3. Meaningful variable names. 4. No global variables. 5. Correct documentation.					