Player Status

**Grade settings**: Maximum grade: 100  
**Run**: Yes **Evaluate**: Yes  
**Automatic grade**: Yes

England Carom Committee needs to know the points scored by each player for the upcoming league matches. The committee selects the players as per their points. Points are based on the number of matches won by each player. Create a Python application and help them to calculate the points.

**Requirement 1: Define a ‘Player’ class and its required members.**All the attributes should be private attributes.

**Component Specification:** **Player (Model Class)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type(Class)** | **Attributes** | **Type of attributes** | **Values** |
| **Player** | player\_name  no\_of\_matches  points  status\_category | string  integer  float  string | User input  User input  User input  Should be a calculated value |

|  |  |  |
| --- | --- | --- |
| **Type(Class)** | **Methods** | **Responsibilities** |
| **Player** | A parameterized constructor with 3 arguments:  player\_name, no\_of\_matches, and points respectively | To initialize the private member variables like player\_name, no\_of\_matches, points and status\_category.  The status\_category should be initialized with its default value. |
| Included necessary getters and setters for the private attributes |  |
| find\_status\_category(self) | This method is used to find the **status category**of a player and set the same to the instance variable **'status\_category'.**  If the points are l**ess than or equal to 0**, then set the **status\_category** as **'Miserable'**  If the points are between **1 and 50 (both Inclusive)**, then set the **status\_category** as **'Tolerable'**  If the points are between **51 and 75 (both Inclusive)**, then set the **status\_category** as **'Satisfactory'**  If the points are**above 75**, then set the **status\_category** as **'Excellent'** |

**Requirement 2: Store the Player Details**

The committee now wants to store each player’s point details for their future reference.  For this, you have to create a utility class called **PlayerManagement’** to perform few functionalities.

The method ‘**add\_player\_details()**’  should create a ‘**Player’** object using a parameterized constructor.  Then calculate and set the **status\_category**instance variable by invoking the ‘**find\_status\_category ()**'  method and add this ‘**Player**’ object to the ‘**player\_list’**.

**Component Specification:** **PlayerManagement** **(Utility Class)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Component Name** | **Type(Class)** | **Attributes** | **Type of attributes** | **Methods** | **Responsibilities** |
|  | PlayerManagement | player\_list | List |  |  |
| Add Player Object | PlayerManagement |  |  | add\_player\_details (self,player\_name, no\_of\_matches,points) | This method has to create a **Player** object, identify the status\_category of that player and set the values to the corresponding variable.  Then add this object to the **player\_list**. This method need not return anything. |

**Requirement 3: View the player details based on the status category**

The committee wants to verify the details of their players over a period of time-based on the status category they belong to.

The method ‘**view\_player\_details()’** should identify the**Player** objects from **player\_list**based on the status category that passed as arguments to this method.  Fetch the player’s objects those belong to the status category  specified, store in another list, and return the same.

**Component Specification:** **PlayerManagement(Utility Class)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Type(Class)** | **Methods** | **Returns** | **Responsibilities** |
| View player details of a particular type | PlayerManagement | view\_player\_details (self,status\_category) | List of  Player objects that are within the status category specified. | This method has to identify the Player objects from the player\_list based on the status\_category entered. Fetch those player objects and add them to another list and return the same.  If no players are found under this category, then return an empty list. |

The file **‘main.py’**  is to get the inputs like the **number of players** and**each player’s point details** from the user**.**

The player details will be in the form of a string in the following format:**player\_name:no\_of\_mathes: points .**

Parse these player details by invoking the **add\_player\_details()**method in the PlayerManagement class  which will perform necessary operations and then add the ‘Player’ object to the list.

After adding the object to the list, get the input for the**status category to search**from the user, andinvoke the ‘**view\_player\_details()’** method by passing the status category entered by the user.  If the list returned by this method is empty then display the message “**No players found in this category”** elsedisplay the player details as shown in the sample input/output statements.

**Note:**

* In the sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represent the output.
* Ensure to provide the names for classes, attributes, and methods as specified in the question description.
* Adhere to the code template, if provided and do not edit or delete the codes provided in the code template.

**Sample Input 1:**

Enter the no. of players:**4**Enter player 1 details: **Shane:450:40**Enter player 2 details: **Esther:30:76**Enter player 3 details: **Jane:1500:250**

Enter player 3 details: **Tom:100:70**

Enter the status category to search: **Excellent**

**Sample Output1:**

No. of players in the specified status category:2

Player Name: Esther  
No of matches: 30  
Points: 76.0

Player Name: Jane  
No of matches: 1500  
Points: 250.0

**Sample Input 2:**

Enter the no. of players:**2**

Enter player 1 details: **John:50:50**Enter player 2 details: **Harry:250:400**

Enter the status category to search:**Miserable**

**Sample Output2:**

No players found in this category

### Automatic evaluation[[+]](javascript:void(0);)

#### player.py

1 # Please do not change the given template. Fill your code only in the provided places.

2

3 *class* Player:

4

5 # Write the code for the parameterized constructor here

6 *def* *\_\_init\_\_*(self,player\_name,no\_of\_matches,points):

7 self.*\_\_player\_name*=player\_name

8 self.*\_\_no\_of\_matches*=no\_of\_matches

9 self.*\_\_points*=points

10 self.*\_\_status\_category*=""

11

12 *def* get\_player\_name(self):

13 *return* self.*\_\_player\_name*

14

15 *def* set\_player\_name(self,player\_name):

16 self.*\_\_player\_name*=player\_name

17

18 *def* get\_no\_of\_matches(self):

19 *return* self.*\_\_no\_of\_matches*

20

21 *def* set\_no\_of\_matches(self,no\_of\_matches):

22 self.*\_\_no\_of\_matches*=no\_of\_matches

23

24 *def* get\_points(self):

25 *return* self.*\_\_points*

26

27 *def* set\_points(self,points):

28 self.*\_\_points*=points

29

30 *def* get\_status\_category(self):

31 *return* self.*\_\_status\_category*

32

33 *def* set\_status\_category(self,status\_category):

34 self.*\_\_status\_category*=status\_category

35

36

37

38 *def* find\_status\_category(self):

39 # Fill your code here

40 *if* self.*\_\_points*<=0:

41 self.*\_\_status\_category*="Miserable"

42 *elif* self.*\_\_points*>=1 *and* self.*\_\_points*<=60:

43 self.*\_\_status\_category*="Tolerable"

44 *elif* self.*\_\_points*>=51 *and* self.*\_\_points*<=75:

45 self.*\_\_status\_category*="Satisfactory"

46 *else*:

47 self.*\_\_status\_category*="Excellent"

48

49

50 *return*

51

52

53

54

55

56

57

58

59

60

61

62

63

64

#### player\_management.py

1 # Please do not alter the given template

2 # You can add any number of methods and attributes as you required without changing the given template

3

4 *import* player *as* pl

5

6 *class* PlayerManagement:

7 *def* *\_\_init\_\_*(self):

8 self.*\_\_player\_list*=[]

9

10

11 *def* add\_player\_details(self,player\_name,no\_of\_matches,points):

12 #Fill your code here

13 p\_obj=pl.Player(player\_name,no\_of\_matches,points)

14 p\_obj.find\_status\_category()

15 self.*\_\_player\_list*.append(p\_obj)

16 *return*

17

18 *def* view\_player\_details(self,status\_category):

19 #Fill your code here

20 lst=[]

21 *for* i *in* self.*\_\_player\_list*:

22 *if* status\_category==i.get\_status\_category():

23 lst.append(i)

24 *return* lst #TODO CHANGE THIS RETURN VALUE

25

26

#### main.py

1 # Please do not alter the given template

2

3 *import* player\_management *as* pm

4

5 *def* main():

6 no\_players=int(input("Enter the no. of players:"))

7

8 # Fill your code here for getting inputs, creating appropriate object and

9 # invoke necessary methods as per the requirement specification.

10 player\_obj=pm.PlayerManagement()

11 *for* i *in* range(no\_players):

12 print("Enter the player details ",i+1,":")

13 name,no\_of\_matches,points=input().split(':')

14 player\_obj.add\_player\_details(name,no\_of\_matches,float(points))

15

16 status\_category=input("Enter the status category to search:")

17

18 player\_rec=player\_obj.view\_player\_details(status\_category)

19 # Fill your code for invoking the appropriate method(s) as per the requirements

20 *if* len(player\_rec)!=0:

21 print("The no. of player in this category:",len(player\_rec))

22 *for* i *in* player\_rec:

23 print("Player Name:",i.get\_player\_name())

24 print("No of matches:",i.get\_no\_of\_matches())

25 print("Points:",i.get\_points())

26 # print("Status category:",i.get\_status\_category())

27

28 *else*:

29 print("No players found in this category")

30

31

32

33

34 *if* *\_\_name\_\_*=='\_\_main\_\_':

35 main()

## Grade

Reviewed on Tuesday, 30 November 2021, 12:17 PM by Automatic grade  
**Grade** 100 / 100  
**Assessment report**

*All testcases passed. Your code has been submitted for evaluation.*