

Class Design Strategy In Python

Boat Earphone Class

```
1
2
3 -----
4 """
5     @Author:Kunal Narkhede
6     @Date:27/12/2023
7     @Goal:To implement class Earphone
8     Capture Real Life Product on Amazon
9     http://surl.li/oqujf
10
11 """
12 -----
13 import sys
14 class ProductDimension:
15     """
16     This class implement the Dimension of Earphone
17     @__init__(self, length: float, width: float, height: float, weight: float):
18     Constructor
19     @get_length(self)
20     getter of attribute length
21     @get_width(self)
22     getter of attribute width
23     @get_height(self)
24     getter of attribute height
25     @get_weight(self)
26     getter of attribute weight
27     -----
28     @set_length(self):
29     setter of attribute length
30     @set_width(self):
31     setter of attribute width
32     @set_height(self):
33     setter of attribute height
34     @set_weight(self):
```

```
35         setter of attribute weight
36     """
37     def __init__(
38         self,
39         length:float,
40         width:float,
41         height:float,
42         weight:float
43     ):
44
45     """
46     Constructor of ProductDimension class:
47     @__init__(self, length: float, width: float, height: float, weight: float):
48
49     @self:newly created class object of ProductDimension
50     @length:Client specified value for attribute length
51     @width:Client specified value for attribute width
52     @height:Client specified value for attribute height
53     @weight:Client specified value for attribute weight
54
55     """
56     if type(length)!=float:
57         raise TypeError("Bad type:length")
58     if type(width)!=float:
59         raise TypeError("Bad type:width")
60     if type(height)!=float:
61         raise TypeError("Bad type:height")
62     if type(weight)!=float:
63         raise TypeError("Bad type:weight")
64     if length<=0.0:
65         raise ValueError("Length must be positive")
66     if width<=0.0:
67         raise ValueError("Width must be positive")
68     if height<=0.0:
69         raise ValueError("Height must be positive")
```

```
70     if weight<=0.0:
71         raise ValueError("Weight must be positive")
72
73     self.length=length
74     self.width=width
75     self.height=height
76     self.weight=weight
77
78     #getter method
79
80     def get_length(self) -> float:
81         """
82         Returns the length attribute of the calling object
83         """
84         return self.length
85
86     def get_width(self) -> float:
87         """
88         Returns the width attribute of the calling object
89         """
90         return self.width
91
92     def get_height(self) -> float:
93         """
94         Returns the height attribute of the calling object
95         """
96         return self.height
97
98     def get_weight(self) -> float:
99         """
100         Returns the weight attribute of the calling object
101         """
102         return self.weight
103
104     #setter method
```

```
105
106 def set_length(self,new_length:float):
107     """
108     Sets the length attribute of the calling object to @new_length
109     Before setting, TypeCheck and ValueCheck is performed.
110     """
111     if type(new_length)!=float:
112         raise TypeError("new_length must be an float")
113     if new_length <= 0.0:
114         raise TypeError("new_length must be positive")
115     self.length=new_length
116
117 def set_width(self,new_width:float):
118     """
119     Sets the width attribute of the calling object to @new_width
120     Before setting, TypeCheck and ValueCheck is performed.
121     """
122     if type(new_width)!=float:
123         raise TypeError("new_width must be an float")
124     if new_width <= 0.0:
125         raise ValueError("new_width must be positive")
126     self.width=new_width
127
128 def set_height(self,new_height:float):
129     """
130     Sets the height attribute of the calling object to @new_height
131     Before setting, TypeCheck and ValueCheck is performed.
132     """
133     if type(new_height)!=float:
134         raise TypeError("new_height must be an float")
135     if new_height <= 0.0 :
136         raise ValueError("new_height must be positive")
137     self.height=new_height
138
139 def set_weight(self,new_weight:float):
```

```
140     """
141     Sets the weight attribute of the calling object to @new_weight
142     Before setting, TypeCheck and ValueCheck is performed.
143     """
144     if type(new_weight)!=float:
145         raise TypeError("new_weight must be an float")
146     if new_weight <= 0.0:
147         raise ValueError("new_weight must be positive")
148     self.weight=new_weight
149
150
151 class Earphone:
152     def __init__(self,
153         er_brand:str,
154         er_cable_feature:str,
155         er_net_quantity_in_piece:int,
156         er_colour:str,
157         er_battery_backup_in_hrs:int,
158         er_control_method:str,
159         er_nr_of_items:int,
160         er_manufacturer:str,
161         er_model:str,
162         er_prod_dimensions:ProductDimension,
163         er_batteries:str,
164         er_model_nr:str,
165         er_special_feature:str,
166         er_mounting_hardware:[str],
167         er_batteries_included:bool,
168         er_batteries_required:bool,
169         er_batteries_cell_composition:str,
170         er_from_factor:str,
171         er_rechargeable_battery:bool,
172         er_country_origin:str
173     ):
174
```

```
175     if type(er_brand)!=str:
176         raise TypeError("er_brand must be in str")
177     if type(er_cable_feature)!=str:
178         raise TypeError("ep_cable_feature must be in str")
179     if type(er_net_quantity_in_piece)!=int:
180         raise TypeError("er_net_quantity_in_piece must be in str")
181     if er_net_quantity_in_piece<=0:
182         raise ValueError("er_net_quantity_in_piece must be positive")
183     if type(er_colour)!=str:
184         raise TypeError("er_colour must be in str")
185     if type(er_battery_backup_in_hrs)!=int:
186         raise TypeError("er_battery_backup_in_hrs must be in int")
187     if er_battery_backup_in_hrs<=0:
188         raise TypeError("er_battery_backup_in_hrs must be positive")
189     if type(er_control_method)!=str:
190         raise TypeError("er_control_method must be str")
191     if type(er_nr_of_items)!=int:
192         raise TypeError("er_nr_of_items must be int")
193     if er_nr_of_items<=0:
194         raise ValueError("er_nr_of_items must be positive")
195     if type(er_manufacturer)!=str:
196         raise TypeError("er_manufacturer must be in str")
197     if type(er_model)!=str:
198         raise TypeError("er_model must be str")
199     if type(er_prod_dimensions)!=ProductDimension:
200         raise TypeError("er_prod_dimensions must be in ProductDimension")
201     if type(er_batteries)!=str:
202         raise TypeError("er_batteries must be in str")
203     if type(er_model_nr)!=str:
204         raise TypeError("er_model_nr must be in str")
205     if type(er_special_feature)!=str:
206         raise TypeError("er_special_feature must be in str")
207     if '__iter__' not in dir(type(er_mounting_hardware)):
208         raise TypeError("er_mounting_hardware must be iterable")
209     for hardware in er_mounting_hardware:
```

```
210         if type(hardware)!=str:
211             raise TypeError("hardware must be in str")
212     if type(er_batteries_included)!=bool:
213         raise TypeError("er_batteries_included must be bool")
214     if type(er_batteries_required)!=bool:
215         raise TypeError("er_batteries_required must be bool")
216     if type(er_batteries_cell_composition)!=str:
217         raise TypeError("er_batteries_cell_composition must be str")
218     if type(er_from_factor)!=str:
219         raise TypeError("er_from_factor must be str")
220     if type(er_rechargeble_battery)!=bool:
221         raise TypeError("er_rechargeble_battery must be bool")
222     if type(er_country_origin)!=str:
223         raise TypeError("er_country_origin must be str")
224
225     self.er_brand=er_brand
226     self.er_cable_feature=er_cable_feature
227     self.er_net_quantity_in_piece=er_net_quantity_in_piece
228     self.er_colour=er_colour
229     self.er_battery_backup_in_hrs=er_battery_backup_in_hrs
230     self.er_control_method=er_control_method
231     self.er_nr_of_items=er_nr_of_items
232     self.er_manufacturer=er_manufacturer
233     self.er_model=er_model
234     self.er_prod_dimensions=er_prod_dimensions
235     self.er_batteries=er_batteries
236     self.er_model_nr=er_model_nr
237     self.er_special_feature=er_special_feature
238     self.er_mounting_hardware=er_mounting_hardware
239     self.er_batteries_included=er_batteries_included
240     self.er_batteries_required=er_batteries_required
241     self.er_batteries_cell_composition=er_batteries_cell_composition
242     self.er_from_factor=er_from_factor
243     self.er_rechargeble_battery=er_rechargeble_battery
244     self.er_country_origin=er_country_origin
```

```
245
246 #Getter Method
247
248 def get_er_brand(self)->str:
249     """
250     Returns the er_brand attribute of the calling object
251     """
252     return self.er_brand
253 def get_er_cable_feature(self)->str:
254     """
255     Returns the er_cable_feature attribute of the calling object
256     """
257     return self.er_cable_feature
258
259 def get_er_net_quantity_in_piece(self)->int:
260     """
261     Returns the er_net_quantity_in_piece attribute of the calling object
262     """
263     return self.er_net_quantity_in_piece
264
265 def get_er_colour(self)->str:
266     """
267     Returns the er_colour attribute of the calling object
268     """
269     return self.er_colour
270
271 def get_er_battery_backup_in_hrs(self)->int:
272     """
273     Returns the er_battery_backup_in_hrs attribute of the calling object
274     """
275     return self.er_battery_backup_in_hrs
276
277 def get_er_control_method(self)->str:
278     """
279     Returns the er_control_method attribute of the calling object
```



```
280     """
281     return self.er_control_method
282
283     def get_er_nr_of_items(self)->int:
284         """
285         Returns the er_nr_of_items attribute of the calling object
286         """
287         return self.er_nr_of_items
288
289     def get_er_manufacturer(self)->str:
290         """
291         Returns the er_manufacturer attribute of the calling object
292         """
293         return self.er_manufacturer
294
295     def get_er_model(self)->str:
296         """
297         Returns the er_model attribute of the calling object
298         """
299         return self.er_model
300
301     def get_er_prod_dimensions(self)->ProductDimension:
302         """
303         Returns the er_prod_dimensions attribute of the calling object
304         """
305         return self.er_prod_dimensions
306
307     def get_er_batteries(self):
308         """
309         Returns the er_batteries attribute of the calling object
310         """
311         return self.er_batteries
312
313     def get_er_model_nr(self)->str:
314         """
```

```
315         Returns the er_model_nr attribute of the calling object
316     """
317     return self.er_model_nr
318
319     def get_er_special_feature(self)->str:
320         """
321         Returns the er_special_feature attribute of the calling object
322         """
323         return self.er_special_feature
324
325     def get_er_mounting_hardware(self)->[str]:
326         """
327         Returns the er_special_feature attribute of the calling object
328         """
329         return self.er_mounting_hardware
330
331     def get_er_batteries_included(self)->bool:
332         """
333         Returns the er_batteries_included attribute of the calling object
334         """
335         return self.er_batteries_included
336
337     def get_er_batteries_required(self)->bool:
338         """
339         Returns the er_batteries_required attribute of the calling object
340         """
341         return self.er_batteries_required
342
343     def get_er_batteries_cell_composition(self)->str:
344         """
345         Returns the er_batteries_cell_composition attribute of the calling object
346         """
347         return self.er_batteries_cell_composition
348     def get_er_from_factor(self)->str:
349         """
```

```

350         Returns the er_from_factor attribute of the calling object
351     """
352     return self.er_from_factor
353
354     def get_er_rechargeable_battery(self)->bool:
355         """
356         Returns the er_rechargeable_battery attribute of the calling object
357         """
358         return self.er_rechargeable_battery
359
360     def get_er_country_origin(self)->str:
361         """
362         Returns the er_country_origin attribute of the calling object
363         """
364         return self.er_country_origin
365
366     #Setter Method
367
368     def set_er_brand(self,new_er_brand):
369         """
370         Sets the er_brand attribute of the calling object to @new_er_cable_feature
371         Before setting, TypeCheck is performed and this function returns nothing
372         """
373         if type(new_er_brand)!=str:
374             raise TypeError("new_er_brand must be in str")
375         self.er_brand=new_er_brand
376
377     def set_er_cable_feature(self,new_er_cable_feature)->None:
378         """
379         Sets the er_cable_feature attribute of the calling object to @new_er_cable_feature
380         Before setting, TypeCheck is performed and this function returns nothing
381         """
382         if type(new_er_cable_feature)!=str:
383             raise TypeError("new_er_cable_feature must be str")
384         self.er_cable_feature=new_er_cable_feature

```

```

385     def set_er_net_quantity_in_piece(self,new_er_net_quantity_in_piece)->None:
386         """
387         Sets the er_net_quantity_in_piece attribute of the calling object to
388         @new_er_net_quantity_in_piece
389         Before setting, TypeCheck and ValueCheck is performed.and this
390         function returns nothing
391         """
392         if type(new_er_net_quantity_in_piece)!=int:
393             raise TypeError("new_er_net_quantity_in_piece must be in int")
394         if new_er_net_quantity_in_piece<=0:
395             raise ValueError("new_er_net_quantity_in_piece must be positive")
396         self.er_net_quantity_in_piece=new_er_net_quantity_in_piece
397     def set_er_colour(self,new_er_colour)->None:
398         """
399         Sets the er_colour attribute of the calling object to @new_er_colour
400         Before setting, TypeCheck is performed and this function returns nothing
401         """
402         if type(new_er_colour)!=str:
403             raise TypeError("new_er_colour must be in str")
404         self.er_colour=new_er_colour
405
406     def set_er_battery_backup_in_hrs(self,new_er_battery_backup_in_hrs)->None:
407         """
408         Sets the er_battery_backup_in_hrs attribute of the calling object to
409         @new_er_battery_backup_in_hrs
410         Before setting, TypeCheck and ValueCheck is performed.and this
411         function returns nothing
412         """
413         if type(new_er_battery_backup_in_hrs)!=int:
414             raise TypeError("new_er_battery_backup_in_hrs must be in str")
415         if new_er_battery_backup_in_hrs<=0:
416             raise ValueError("new_er_battery_backup_in_hrs must be positive")
417         self.er_battery_backup_in_hrs=new_er_battery_backup_in_hrs
418
419     def set_er_control_method(self,new_er_control_method)->None:
420         """

```

```
421         Sets the er_control_method attribute of the calling object to @new_er_control_method
422         Before setting, TypeCheck is performed and this function returns nothing
423         """
424         if type(new_er_control_method)!=str:
425             raise TypeError("new_er_control_method must be in str")
426         self.er_control_method=new_er_control_method
427
428     def set_er_nr_of_items(self,new_er_nr_of_items)->None:
429         """
430         Sets the er_nr_of_items attribute of the calling object to @new_er_nr_of_items
431         Before setting, TypeCheck and ValueCheck is performed.and this
432         function returns nothing
433         """
434         if type(new_er_nr_of_items)!=int:
435             raise TypeError("new_er_nr_of_items must be int")
436         if new_er_nr_of_items<=0:
437             raise ValueError("er_nr_of_items must be positive")
438         self.er_nr_of_items=new_er_nr_of_items
439
440     def set_er_manufacturer(self,new_er_manufacturer)->None:
441         """
442         Sets the er_manufacturer attribute of the calling object to @new_er_manufacturer
443         Before setting, TypeCheck is performed and this function returns nothing
444         """
445         if(new_er_manufacturer)!=str:
446             TypeError("new_er_manufacturer must be in str")
447         self.er_manufacturer=new_er_manufacturer
448
449     def set_er_model(self,new_er_model)->None:
450         """
451         Sets the er_model attribute of the calling object to @new_er_model
452         Before setting, TypeCheck is performed and this function returns nothing
453         """
454         if type(new_er_model)!=str:
455             raise TypeError("new_er_model must be in str")
```

```

456     self.er_model=new_er_model
457
458     def set_er_prod_dimensions(self,new_er_prod_dimensions)->None:
459         """
460             Sets the er_prod_dimensions attribute of the calling object to @new_er_prod_dimensions
461             Before setting, TypeCheck is performed and this function returns nothing
462         """
463         if type(new_er_prod_dimensions)!=ProductDimension:
464             raise TypeError("new_er_prod_dimensions must be in ProductDimension")
465         self.er_prod_dimensions=new_er_prod_dimensions
466
467     def set_er_batteries(self,new_er_batteries)->None:
468         """
469             Sets the er_batteries attribute of the calling object to @new_er_batteries
470             Before setting, TypeCheck is performed and this function returns nothing
471         """
472         if type(new_er_batteries)!=str:
473             raise TypeError("new_er_batteries must be in str")
474         self.er_batteries=new_er_batteries
475
476     def set_er_model_nr(self,new_er_model_nr)->None:
477         """
478             Sets the er_model_nr attribute of the calling object to @new_er_model_nr
479             Before setting, TypeCheck is performed and this function returns nothing
480         """
481         if type(new_er_model_nr)!=str:
482             raise TypeError("new_er_model_nr must be in str")
483         self.er_model_nr=new_er_model_nr
484
485     def set_er_special_feature(self,new_er_special_feature)->None:
486         """
487             Sets the er_special_feature attribute of the calling object to @new_er_special_feature
488             Before setting, TypeCheck is performed and this function returns nothing
489         """
490         if type(new_er_special_feature)!=str:

```

```

491         raise TypeError("new_er_special_feature must be in str")
492     self.er_special_feature=new_er_special_feature
493
494     def set_er_mounting_hardware(self,new_er_mounting_hardware)->None:
495         """
496         Sets the er_mounting_hardware attribute of the calling object to @new_er_mounting_hardware
497         Before setting, TypeCheck is performed and this function returns nothing
498         """
499
500         if '__iter__' not in dir(type(new_er_mounting_hardware)):
501             raise TypeError("new_er_mounting_hardware must be iterable")
502         for hardware in new_er_mounting_hardware:
503             if type(hardware)!=str:
504                 raise TypeError("hardware must be in str")
505         self.er_mounting_hardware=new_er_mounting_hardware
506     def set_er_batteries_included(self,new_er_batteries_included)->None:
507         """
508         Sets the er_batteries_included attribute of the calling object to @new_er_batteries_included
509         Before setting, TypeCheck is performed and this function returns nothing
510         """
511         if type(new_er_batteries_included)!=bool:
512             raise TypeError("new_er_batteries_included must be in bool")
513         self.er_batteries_included=new_er_batteries_included
514
515     def set_er_batteries_required(self,new_er_batteries_required)->None:
516         """
517         Sets the er_batteries_required attribute of the calling object to @new_er_batteries_required
518         Before setting, TypeCheck is performed and this function returns nothing
519         """
520         if type(new_er_batteries_required)!=bool:
521             raise TypeError("new_er_batteries_required must be in bool")
522         self.er_batteries_required=new_er_batteries_required
523
524     def set_er_batteries_cell_composition(self,new_er_batteries_cell_composition)->None:
525         """

```

```

526         Sets the er_batteries_cell_composition attribute of the calling object to
527 @new_er_batteries_cell_composition
528         Before setting, TypeCheck is performed and this function returns nothing
529         """
530         if type(new_er_batteries_cell_composition)!=str:
531             raise TypeError("new_er_batteries_cell_composition must be in str")
532         self.er_batteries_cell_composition=new_er_batteries_cell_composition
533
534     def set_er_from_factor(self,new_er_from_factor)->None:
535         """
536         Sets the er_from_factor attribute of the calling object to @new_er_from_factor
537         Before setting, TypeCheck is performed and this function returns nothing
538         """
539         if type(new_er_from_factor)!=str:
540             raise TypeError("new_er_from_factor must be in str")
541         self.er_from_factor=new_er_from_factor
542
543     def set_er_rechargeable_battery(self,new_er_rechargeable_battery)->None:
544         """
545         Sets the er_rechargeable_battery attribute of the calling object to
546 @new_er_rechargeable_battery
547         Before setting, TypeCheck is performed and this function returns nothing
548         """
549         if type(new_er_rechargeable_battery)!=bool:
550             raise TypeError("new_er_rechargeable_battery must be in str")
551         self.er_rechargeable_battery=new_er_rechargeable_battery
552
553     def set_er_country_origin(self,new_er_country_origin)->None:
554         """
555         Sets the er_country_origin attribute of the calling object to @new_er_country_origin
556         Before setting, TypeCheck is performed and this function returns nothing
557         """
558         if type(new_er_country_origin)!=str:
559             raise TypeError("new_er_country_origin must be in str")
560         self.er_country_origin=new_er_country_origin
561     def show_earphone_details(self):

```



```
562     print("Brand:{}".format(self.er_brand))
563     print("Cable Feature: {}".format(self.er_cable_feature))
564     print("Net Quantity: {}".format(self.er_net_quantity_in_piece))
565     print("Colour: {}".format(self.er_colour))
566     print("Battery Backup in hours: {}".format(self.er_battery_backup_in_hrs))
567     print("Control Method: {}".format(self.er_control_method))
568     print("Number of items: {}".format(self.er_nr_of_items))
569     print("Manufacturer: {}".format(self.er_manufacturer))
570     print("Model: {}".format(self.er_model))
571     print("Product Dimensions: {}".format(self.er_prod_dimensions.__dict__))
572     print("Batteries: {}".format(self.er_batteries))
573     print("Item model number: {}".format(self.er_model_nr))
574     print("Special Features: {}".format(self.er_special_feature))
575     print("Mounting Hardware:{}".format(self.er_mounting_hardware))
576     print("Batteries Included: {}".format(self.er_batteries_included))
577     print("Batteries Required: {}".format(self.er_batteries_required))
578     print("Batteries Cell Composition:{}".format(self.er_batteries_cell_composition))
579     print("From Factor: {}".format(self.er_from_factor))
580     print("Includes Rechargeable Battery:{}".format(self.er_rechargeable_battery))
581     print("Country of Origin: {}".format(self.er_country_origin))
582
583
584 def main():
585     er_obj=Earphone(
586         "Boat",
587         "Without Cable",
588         1,
589         "Black",
590         60,
591         "Voice",
592         4,
593         "Boat",
594         "Rockerz 255 Pro+",
595         ProductDimension(45.0,1.0,1.0,45.0),
596         "1 Lithium Ion batteries required(included)",
```

```
597         "Rockerz 255 Pro+",
598         "Built-In Voice Assistant",
599         "Rockerz 255 Pro +, Charging Cable, Warranty Card, User Manual",
600         True,
601         True,
602         "Lithium Ion",
603         "In ear",
604         True,
605         "India")
606     print("EARPHONE PRODUCT DETAILS:")
607     er_obj.show_earphone_details()
608     #we can also get the attribute using getter method and
609     #set the specific attribute using setter method
610
611     sys.exit(0)
612 main()
613
614 """
615 Output:-
616 EARPHONE PRODUCT DETAILS:
617 Brand:Boat
618 Cable Feature: Without Cable
619 Net Quantity: 1
620 Colour: Black
621 Battery Backup in hours: 60
622 Control Method: Voice
623 Number of items: 4
624 Manufacturer: Boat
625 Model: Rockerz 255 Pro+
626 Product Dimensions: {'length': 45.0, 'width': 1.0, 'height': 1.0, 'weight': 45.0}
627 Batteries: 1 Lithium Ion batteries required(included)
628 Item model number: Rockerz 255 Pro+
629 Special Features: Built-In Voice Assistant
630 Mounting Hardware:Rockerz 255 Pro +, Charging Cable, Warranty Card, User Manual
631 Batteries Included: True
632 Batteries Required: True
633 Batteries Cell Composition:Lithium Ion
```

634 From Factor: In ear
635 Includes Rechargeble Battery:True
636 Country of Origin: India
637 ""
638