

# Class Design Strategy In Python

## OnePlus Mobile Class

```
1  -----
2
3  """
4
5  @Author:Kunal Narkhede
6  @Date:24/12/2023
7  @Goal:To implement class Mobile
8  Capture Real Life Product on Amazon
9  http://surl.li/ooqwc
10 """
11 -----
12
13 import sys
14 class ProductDimension:
15     """
16     this class implement the Dimension of Mobile
17     @__init__(self, length: float, width: float, height: float, weight_in_gm: 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_in_gm(self)
26     getter of attribute weight_in_gm
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_in_gm(self):
```

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

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

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

```

140 def set_weight_in_gm(self,new_weight_in_gm:float):
141     """
142     Sets the weight_in_gm attribute of the calling object to @new_weight_in_gm
143     Before setting, TypeCheck and ValueCheck is performed.
144     """
145     if type(new_weight_in_gm)!=float:
146         raise TypeError("new_weight_in_gm must be an float")
147     if new_weight_in_gm <= 0.0:
148         raise ValueError("new_weight_in_gm must be positive")
149     self.weight_in_gm=new_weight_in_gm
150
151
152 class Mobile:
153
154     def __init__(self,
155                 mob_GPU:str,
156                 mob_RAM_in_gb:int,
157                 mob_prod_dimensions:ProductDimension,
158                 mob_battries:str,
159                 mob_wireless_commu_tech:[str],
160                 mob_special_feature:[str],
161                 mob_display_tech:str,
162                 mob_manufacturer:str,
163                 mob_country_of_origin:str,
164                 mob_connectivity_tech:[str],
165                 mob_colour:str,
166                 mob_screen_size_in_inches:float,
167                 mob_connector_type:str,
168                 mob_front_photo_sensor_reso_in_mp:int,
169                 mob_from_factor:str,
170                 mob_battries_capacity_in_MH:int,
171                 mob_rear_camera_reso_in_mp:int,
172                 mob_model_year:int,
173                 mob_CPU_model:str,
174                 mob_included_components:[str],

```

```
175     mob_display_type:str,
176     mob_human_interface_input:str,
177     mob_batteries_desc:str,
178     mob_sim_card_size:str,
179     mob_material_feature:str,
180     mob_shooting_modes:[str],
181     mob_GPS:str,
182     mob_water_resistence_level:str,
183     mob_optical_sensor_reso_in_mp:int,
184     mob_max_display_reso:str,
185     mob_video_capture_reso_in_pixel:int
186 ):
187
188
189 if type(mob_GPU)!=str:
190     raise TypeError("mob_GPU must be in str")
191 if type(mob_RAM_in_gb)!=int:
192     raise TypeError("mob_RAM_in_gb must be in int")
193 if mob_RAM_in_gb<=0:
194     raise ValueError("mob_RAM_in_gb must be positive")
195 if type(mob_prod_dimensions)!=ProductDimension:
196     raise TypeError("mob_prod_dimensions must be in ProductDimension")
197 if type(mob_batttries)!=str:
198     raise TypeError("mob_batttries must be in str")
199 if '__iter__' not in dir(type(mob_wireless_commu_tech)):
200     raise TypeError("mob_wireless_commu_tech must be iterable")
201 for technology in mob_wireless_commu_tech:
202     if type(technology)!=str:
203         raise TypeError("technology must be in str")
204 if '__iter__' not in dir(type(mob_special_feature)):
205     raise TypeError("mob_special_feature must be iterable")
206 for feature in mob_special_feature:
207     if type(feature)!=str:
208         raise TypeError("feature must be str")
209 if type(mob_display_tech)!=str:
```

```
210         raise TypeError("mob_display_tech must be str")
211     if type(mob_manufacturer)!=str:
212         raise TypeError("mob_manufacturer must be in str")
213     if type(mob_country_of_origin)!=str:
214         raise TypeError("mob_country_of_origin must be in str")
215     if '__iter__' not in dir(type(mob_connectivity_tech)):
216         raise TypeError("mob_connectivity_tech must be iterable")
217     for technology in mob_connectivity_tech:
218         if type(technology)!=str:
219             raise TypeError("technology must be in str")
220     if type(mob_colour)!=str:
221         raise TypeError("mob_colour muse be in str")
222     if type(mob_screen_size_in_inches)!=float:
223         raise TypeError("mob_screen_size_in_inches must be in float")
224     if type(mob_connector_type)!=str:
225         raise TypeError("mob_connector_type must be in str")
226     if type(mob_front_photo_sensor_reso_in_mp)!=int:
227         raise TypeError("mob_front_photo_sensor_reso_in_mp must be in int")
228     if mob_front_photo_sensor_reso_in_mp<=0:
229         raise ValueError("mob_front_photo_sensor_reso_in_mp must be positive")
230     if type(mob_from_factor)!=str:
231         raise TypeError("mob_from_factor must be in str")
232     if type(mob_batttries_capacity_in_MH)!=int:
233         raise TypeError("mob_batttries_capacity_in_MH must be in int")
234     if mob_batttries_capacity_in_MH<=0:
235         raise ValueError("mob_batttries_capacity_in_MH must be positive")
236     if type(mob_rear_camera_reso_in_mp)!=int:
237         raise TypeError("mob_rear_camera_reso_in_mp must be int")
238     if type(mob_model_year)!=int:
239         raise TypeError("mob_model_year must be in int")
240     if mob_model_year<=0:
241         raise ValueError("mob_model_year must be in positive")
242     if type(mob_CPU_model)!=str:
243         raise TypeError("mob_CPU_model must be in str")
244     if mob_rear_camera_reso_in_mp<=0:
```

```
245         raise ValueError("mob_rear_camera_reso_in_mp must be in positive")
246     if '__iter__' not in dir(type(mob_included_components)):
247         raise TypeError("mob_included_components must be iterable")
248     for component in mob_included_components:
249         if type(component)!=str:
250             raise TypeError("component must be str")
251     if type(mob_display_type)!=str:
252         raise TypeError("mob_display_type must be in str")
253     if type(mob_human_interface_input)!=str:
254         raise TypeError("mob_human_interface_input must be str")
255     if type(mob_batteries_desc)!=str:
256         raise TypeError("mob_batteries_desc must be in str")
257     if type(mob_sim_card_size)!=str:
258         raise TypeError("mob_sim_card_size must be in str")
259     if type(mob_material_feature)!=str:
260         raise TypeError("mob_material_feature must be in str")
261     if '__iter__' not in dir(type(mob_shooting_modes)):
262         raise TypeError("mob_shooting_modes must be iterable")
263     for mode in mob_shooting_modes:
264         if type(mode)!=str:
265             raise TypeError("mode must be in str")
266     if type(mob_water_resistence_level)!=str:
267         raise TypeError("mob_water_resistence_level must be in str")
268     if type(mob_optical_sensor_reso_in_mp)!=int:
269         raise TypeError("mob_optical_sensor_reso_in_mp must be in int")
270     if mob_optical_sensor_reso_in_mp<=0:
271         raise ValueError("mob_optical_sensor_reso_in_mp must be positive")
272     if type(mob_max_display_reso)!=str:
273         raise TypeError("mob_max_display_reso must be in str")
274     if type(mob_video_capture_reso_in_pixel)!=int:
275         raise TypeError("mob_video_capture_reso_in_pixel must be in int")
276     if mob_video_capture_reso_in_pixel<=0:
277         raise ValueError("mob_video_capture_reso_in_pixel must be positive")
278
279     self.mob_GPU=mob_GPU
```



```
280     self.mob_RAM_in_gb=mob_RAM_in_gb
281     self.mob_prod_dimensions=mob_prod_dimensions
282     self.mob_battries=mob_battries
283     self.mob_wireless_commu_tech=mob_wireless_commu_tech
284     self.mob_special_feature=mob_special_feature
285     self.mob_display_tech=mob_display_tech
286     self.mob_manufacturer=mob_manufacturer
287     self.mob_country_of_origin=mob_country_of_origin
288     self.mob_connectivity_tech=mob_connectivity_tech
289     self.mob_colour=mob_colour
290     self.mob_screen_size_in_inches=mob_screen_size_in_inches
291     self.mob_connector_type=mob_connector_type
292     self.mob_front_photo_sensor_reso_in_mp=mob_front_photo_sensor_reso_in_mp
293     self.mob_from_factor=mob_from_factor
294     self.mob_battries_capacity_in_MH=mob_battries_capacity_in_MH
295     self.mob_rear_camera_reso_in_mp=mob_rear_camera_reso_in_mp
296     self.mob_model_year=mob_model_year
297     self.mob_CPU_model=mob_CPU_model
298     self.mob_included_components=mob_included_components
299     self.mob_display_type=mob_display_type
300     self.mob_human_interface_input=mob_human_interface_input
301     self.mob_batteries_desc=mob_batteries_desc
302     self.mob_sim_card_size=mob_sim_card_size
303     self.mob_material_feature=mob_material_feature
304     self.mob_shooting_modes=mob_shooting_modes
305     self.mob_GPS=mob_GPS
306     self.mob_water_resistence_level=mob_water_resistence_level
307     self.mob_optical_sensor_reso_in_mp=mob_optical_sensor_reso_in_mp
308     self.mob_max_display_reso=mob_max_display_reso
309     self.mob_video_capture_reso_in_pixel=mob_video_capture_reso_in_pixel
310
311
312     #getter method
313
314     def get_mob_GPU(self)->str:
```

```
315         """
316         Returns the mob_GPU attribute of the calling object
317         """
318         return self.mob_GPU
319     def get_mob_RAM_in_gb(self)->int:
320         """
321         Returns the mob_RAM_in_gb attribute of the calling object
322         """
323         return self.mob_RAM_in_gb
324
325     def get_mob_prod_dimensions(self)->ProductDimension:
326         """
327         Returns the mob_prod_dimensions attribute of the calling object
328         """
329         return self.mob_prod_dimensions
330     def get_mob_battries(self)->str:
331         """
332         Returns the mob_battries attribute of the calling object
333         """
334         return self.mob_battries
335
336     def get_mob_wireless_commu_tech(self)->[str]:
337         """
338         Returns the mob_wireless_commu_tech attribute of the calling object
339         """
340         return self.mob_wireless_commu_tech
341
342     def get_mob_special_feature(self)->[str]:
343         """
344         Returns the mob_special_feature attribute of the calling object
345         """
346         return self.mob_special_feature
347
348     def get_mob_display_tech(self)->str:
349         """
```

```
350         Returns the mob_RAM_in_gb attribute of the calling object
351     """
352     return self.mob_display_tech
353
354     def get_mob_manufacturer(self)->str:
355         """
356         Returns the mob_manufacturer attribute of the calling object
357         """
358         return self.mob_manufacturer
359
360     def get_mob_country_of_origin(self)->str:
361         """
362         Returns the mob_country_of_origin attribute of the calling object
363         """
364         return self.mob_country_of_origin
365
366     def get_mob_connectivity_tech(self)->[str]:
367         """
368         Returns the mob_connectivity_tech attribute of the calling object
369         """
370         return self.mob_connectivity_tech
371
372
373     def get_mob_colour(self)->str:
374         """
375         Returns the mob_colour attribute of the calling object
376         """
377         return self.mob_colour
378
379     def get_mob_screen_size_in_inches(self)->float:
380         """
381         Returns the mob_screen_size_in_inches attribute of the calling object
382         """
383         return self.mob_screen_size_in_inches
384
```

```
385 def get_mob_connector_type(self)->str:
386     """
387     Returns the mob_connector_type attribute of the calling object
388     """
389     return self.mob_connector_type
390
391 def get_mob_front_photo_sensor_reso_in_mp(self)->int:
392     """
393     Returns the mob_front_photo_sensor_reso_in_mp attribute of the calling object
394     """
395     return self.mob_front_photo_sensor_reso_in_mp
396
397 def get_mob_from_factor(self)->str:
398     """
399     Returns the mob_from_factor attribute of the calling object
400     """
401     return self.mob_from_factor
402
403 def get_mob_batties_capacity_in_MH(self)->int:
404     """
405     Returns the mob_batties_capacity_in_MH attribute of the calling object
406     """
407     return self.mob_batties_capacity_in_MH
408
409 def get_mob_rear_camera_reso_in_mp(self)->int:
410     """
411     Returns the mob_rear_camera_reso_in_mp attribute of the calling object
412     """
413     return self.mob_rear_camera_reso_in_mp
414 def get_mob_model_year(self)->int:
415     """
416     Returns the mob_model_year attribute of the calling object
417     """
418     return self.mob_model_year
419
```

```
420 def get_mob_CPU_model(self)->int:
421     """
422     Returns the mob_CPU_model attribute of the calling object
423     """
424     return self.mob_CPU_model
425
426 def get_mob_included_components(self)->[str]:
427     """
428     Returns the mob_included_components attribute of the calling object
429     """
430     return self.mob_included_components
431
432 def get_mob_display_type(self)->str:
433     """
434     Returns the mob_display_type attribute of the calling object
435     """
436     return self.mob_display_type
437
438 def get_mob_human_interface_input(self)->str:
439     """
440     Returns the mob_human_interface_input attribute of the calling object
441     """
442     return self.mob_human_interface_input
443
444 def get_mob_batteries_desc(self)->str:
445     """
446     Returns the mob_batteries_desc attribute of the calling object
447     """
448     return self.mob_batteries_desc
449
450 def get_mob_sim_card_size(self)->str:
451     """
452     Returns the mob_sim_card_size attribute of the calling object
453     """
454     return self.mob_sim_card_size
```

```
455
456 def get_mob_material_feature(self)->str:
457     """
458     Returns the mob_material_feature attribute of the calling object
459     """
460     return self.mob_material_feature
461
462 def get_mob_shooting_modes(self)->[str]:
463     """
464     Returns the mob_shooting_modes attribute of the calling object
465     """
466     return self.mob_shooting_modes
467 def get_mob_GPS(self)->[str]:
468     """
469     Returns the mob_GPS attribute of the calling object
470     """
471     return self.mob_GPS
472 def get_mob_water_resistence_level(self)->str:
473     """
474     Returns the mob_water_resistence_level attribute of the calling object
475     """
476     return self.mob_water_resistence_level
477
478 def get_mob_optical_sensor_reso_in_mp(self)->int:
479     """
480     Returns the mob_optical_sensor_reso_in_mp attribute of the calling object
481     """
482     return self.mob_optical_sensor_reso_in_mp
483
484 def get_mob_max_display_reso(self)->str:
485     """
486     Returns the mob_max_display_reso attribute of the calling object
487     """
488     return self.mob_max_display_reso
489
```

```

490
491 def get_mob_video_capture_reso_in_pixel(self)->int:
492     """
493     Returns the mob_video_capture_reso_in_pixel attribute of the calling object
494     """
495     return self.mob_video_capture_reso_in_pixel
496
497 #Setter Method
498 def set_mob_GPU(self,new_mob_GPU)->None:
499     """
500     Sets the mob_GPU attribute of the calling object to @new_mob_GPU
501     Before setting, TypeCheck is performed.
502     """
503     if type(new_mob_GPU)!=str:
504         raise TypeError("new_mob_GPU must be str")
505     self.mob_GPU=new_mob_GPU
506
507 def set_mob_RAM_in_gb(self,new_mob_RAM_in_gb)->None:
508     """
509     Sets the mob_RAM_in_gb attribute of the calling object to @new_mob_RAM_in_gb
510     Before setting, TypeCheck and ValueCheck is performed.
511     """
512     if type(new_mob_RAM_in_gb)!=int:
513         raise TypeError("new_mob_RAM_in_gb must be int")
514     if new_mob_RAM_in_gb <= 0:
515         raise ValueError("new_mob_RAM_in_gb must be positive")
516     self.mob_RAM_in_gb=new_mob_RAM_in_gb
517
518 def set_mob_prod_dimensions(self,new_mob_prod_dimensions)->None:
519     """
520     Sets the mob_prod_dimensions attribute of the calling object to @new_mob_prod_dimensions
521     Before setting, TypeCheck is performed.
522     """
523     if type(new_mob_prod_dimensions)!=ProductDimension:
524         raise TypeError("mob_prod_dimensions must be ProductDimension")

```

```

525     self.mob_prod_dimensions=new_mob_prod_dimensions
526
527     def set_mob_batttries(self,new_mob_batttries)->None:
528         """
529         Sets the mob_batttries attribute of the calling object to @new_mob_batttries
530         Before setting, TypeCheck is performed.
531         """
532         if type(new_mob_batttries)!=str:
533             raise TypeError("mob_batttries must be str")
534         self.mob_batttries=new_mob_batttries
535
536     def set_mob_wireless_commu_tech(self,new_mob_wireless_commu_tech)->None:
537         """
538         Sets the mob_wireless_commu_tech attribute of the calling object to
539         @new_mob_wireless_commu_tech
540         Before setting, TypeCheck is performed.
541         """
542         if '__iter__' not in dir(type(new_mob_wireless_commu_tech)):
543             raise TypeError("new_mob_wireless_commu_tech must be iterable")
544         for technology in new_mob_wireless_commu_tech:
545             if type(technology)!=str:
546                 raise TypeError("technology must be in str")
547         self.mob_wireless_commu_tech=new_mob_wireless_commu_tech
548
549     def set_mob_special_feature(self,new_mob_special_feature)->None:
550         """
551         Sets the mob_special_feature attribute of the calling object to @new_mob_special_feature
552         Before setting, TypeCheck is performed.
553         """
554         if '__iter__' not in dir(type(new_mob_special_feature)):
555             raise TypeError("new_mob_special_feature must be iterable")
556         for feature in new_mob_special_feature:
557             if type(feature)!=str:
558                 raise TypeError("feature must be in str")
559         self.mob_special_feature=new_mob_special_feature
560

```



```

561 def set_mob_display_tech(self,new_mob_display_tech)->None:
562     """
563     Sets the mob_display_tech attribute of the calling object to @new_mob_display_tech
564     Before setting, TypeCheck is performed.
565     """
566     if type(new_mob_display_tech)!=str:
567         raise TypeError("new_mob_display_tech must be str")
568     self.mob_display_tech=new_mob_display_tech
569 def set_mob_manufacturer(self,new_mob_manufacturer)->None:
570     """
571     Sets the mob_manufacturer attribute of the calling object to @new_mob_manufacturer
572     Before setting, TypeCheck is performed.
573     """
574     if type(new_mob_manufacturer)!=str:
575         raise TypeError("new_mob_manufacturermust be in str")
576     self.mob_manufacturer=new_mob_manufacturer
577
578 def set_mob_country_of_origin(self,new_mob_country_of_origin)->None:
579     """
580     Sets the mob_country_of_origin attribute of the calling object to @new_mob_country_of_origin
581     Before setting, TypeCheck is performed.
582     """
583     if type(new_mob_country_of_origin)!=str:
584         raise TypeError("new_mob_country_of_origin must be in str")
585     self.mob_country_of_origin=new_mob_country_of_origin
586
587 def set_mob_connectivity_tech(self,new_mob_connectivity_tech)->None:
588     """
589     Sets the mob_connectivity_tech attribute of the calling object to @new_mob_connectivity_tech
590     Before setting, TypeCheck is performed.
591     """
592     if '__iter__' not in dir(type(new_mob_connectivity_tech)):
593         raise TypeError("new_mob_connectivity_tech must be in iterable")
594     for technology in new_mob_connectivity_tech:
595         if type(technology)!=str:

```

```

596         raise TypeError("technology must be in str")
597     self.mob_connectivity_tech=new_mob_connectivity_tech
598
599     def set_mob_colour(self,new_mob_colour)->None:
600         """
601             Sets the mob_colour attribute of the calling object to @new_mob_colour
602             Before setting, TypeCheck is performed.
603         """
604         if type(new_mob_colour)!=str:
605             raise TypeError("new_mob_colour must be in str")
606         self.mob_colour=new_mob_colour
607
608     def set_mob_screen_size_in_inches(self,new_mob_screen_size_in_inches)->None:
609         """
610             Sets the mob_screen_size_in_inches attribute of the calling object to
611             @new_mob_screen_size_in_inches
612             Before setting, TypeCheck and ValueCheck is performed.
613         """
614         if type(new_mob_screen_size_in_inches)!=float:
615             raise TypeError("new_mob_screen_size_in_inches must be in float")
616         if new_mob_screen_size_in_inches<=0.0:
617             raise ValueError("new_mob_screen_size_in_inches must be in positive")
618         self.mob_screen_size_in_inches=new_mob_screen_size_in_inches
619
620     def set_mob_connector_type(self,new_mob_connector_type)->None:
621         """
622             Sets the mob_connector_type attribute of the calling object to @new_mob_connector_type
623             Before setting, TypeCheck is performed.
624         """
625         if type(new_mob_connector_type)!=str:
626             raise TypeError("new_mob_connector_type must be in str")
627         self.mob_connector_type=new_mob_connector_type
628
629     def set_mob_front_photo_sensor_reso_in_mp(self,new_mob_front_photo_sensor_reso_in_mp)-
630     >None:
631         """

```

```

632         Sets the mob_front_photo_sensor_reso_in_mp attribute of the calling object to
633 @new_mob_front_photo_sensor_reso_in_mp
634         Before setting, TypeCheck and ValueCheck is performed.
635         """
636         if type(new_mob_front_photo_sensor_reso_in_mp)!=int:
637             raise TypeError("new_mob_front_photo_sensor_reso_in_mp must be in int")
638         if new_mob_front_photo_sensor_reso_in_mp<=0:
639             raise ValueError("new_mob_front_photo_sensor_reso_in_mp must positive")
640         self.mob_front_photo_sensor_reso_in_mp=new_mob_front_photo_sensor_reso_in_mp
641
642     def set_mob_from_factor(self,new_mob_from_factor)->None:
643         """
644         Sets the mob_from_factor attribute of the calling object to @new_mob_from_factor
645         Before setting, TypeCheck is performed.
646         """
647         if type(new_mob_from_factor)!=str:
648             raise TypeError("new_mob_from_factor must be in str")
649         self.mob_from_factor=new_mob_from_factor
650
651     def set_mob_batttries_capacity_in_MH(self,new_mob_batttries_capacity_in_MH)->None:
652         """
653         Sets the mob_batttries_capacity_in_MH attribute of the calling object to
654 @new_mob_batttries_capacity_in_MH
655         Before setting, TypeCheck and ValueCheck is performed.
656         """
657         if type(new_mob_batttries_capacity_in_MH)!=int:
658             raise TypeError("new_mob_batttries_capacity_in_MH must be in int")
659         if new_mob_batttries_capacity_in_MH<=0:
660             raise ValueError("new_mob_batttries_capacity_in_MH must be positive")
661         self.mob_batttries_capacity_in_MH=new_mob_batttries_capacity_in_MH
662
663     def set_mob_rear_camera_reso_in_mp(self,new_mob_rear_camera_reso_in_mp)->None:
664         """
665         Sets the mob_rear_camera_reso_in_mp attribute of the calling object to
666 @new_mob_rear_camera_reso_in_mp
667         Before setting, TypeCheck and ValueCheck is performed.

```

```

668     """
669     if type(new_mob_rear_camera_reso_in_mp)!=int:
670         raise TypeError("new_mob_rear_camera_reso_in_mp must be int")
671     if new_mob_rear_camera_reso_in_mp<=0:
672         raise ValueError("new_mob_rear_camera_reso_in_mp must be positive")
673     self.mob_rear_camera_reso_in_mp=new_mob_rear_camera_reso_in_mp
674 def set_mob_model_year(self,new_mob_model_year)->None:
675     """
676     Sets the mob_model_year attribute of the calling object to @new_mob_model_year
677     Before setting, TypeCheck and ValueCheck is performed.
678     """
679     if type(new_mob_model_year)!=int:
680         raise TypeError("new_mob_model_year must be int")
681     if new_mob_model_year<=0:
682         raise ValueError("new_mob_model_year must be positive")
683     self.mob_model_year=new_mob_model_year
684
685 def set_mob_CPU_model(self,new_mob_CPU_model)->None:
686     """
687     Sets the mob_CPU_model attribute of the calling object to @new_mob_CPU_model
688     Before setting, TypeCheck is performed.
689     """
690     if type(new_mob_CPU_model)!=str:
691         raise TypeError("new_mob_CPU_model must be in str")
692     self.mob_CPU_model=new_mob_CPU_model
693
694 def set_mob_included_components(self,new_mob_included_components)->None:
695     """
696     Sets the mob_included_components attribute of the calling object to
697     @new_mob_included_components
698     Before setting, TypeCheck is performed.
699     """
700     if '__iter__' not in dir(type(new_mob_included_components)):
701         raise TypeError("new_mob_included_components must be iterable")
702     for component in new_mob_included_components:
703         if type(component)!=str:

```

```

704         raise TypeError("new_mob_included_components must be str")
705     self.mob_included_components=new_mob_included_components
706
707     def set_mob_display_type(self,new_mob_display_type)->None:
708         """
709         Sets the mob_display_type attribute of the calling object to @new_mob_display_type
710         Before setting, TypeCheck is performed.
711         """
712         if type(new_mob_display_type)!=str:
713             raise TypeError("new_mob_display_type must be in str")
714         self.mob_display_type=new_mob_display_type
715
716     def set_mob_human_interface_input(self,new_mob_human_interface_input)->None:
717         """
718         Sets the mob_human_interface_input attribute of the calling object to
719         @new_mob_human_interface_input
720         Before setting, TypeCheck is performed.
721         """
722         if type(new_mob_human_interface_input)!=str:
723             raise TypeError("new_mob_human_interface_input must be in str")
724         self.mob_human_interface_input=new_mob_human_interface_input
725
726     def set_mob_batteries_desc(self,new_mob_batteries_desc)->None:
727         """
728         Sets the mob_batteries_desc attribute of the calling object to @new_mob_batteries_desc
729         Before setting, TypeCheck is performed.
730         """
731         if type(new_mob_batteries_desc)!=str:
732             raise TypeError("new_mob_batteries_desc must be in str")
733         self.mob_batteries_desc=new_mob_batteries_desc
734
735     def set_mob_sim_card_size(self,new_mob_sim_card_size)->None:
736         """
737         Sets the mob_sim_card_size attribute of the calling object to @new_mob_sim_card_size
738         Before setting, TypeCheck is performed.
739         """

```

```

740     if type(new_mob_sim_card_size)!=str:
741         raise TypeError("new_mob_sim_card_size must be in str")
742     self.mob_sim_card_size=new_mob_sim_card_size
743
744     def set_mob_material_feature(self,new_mob_material_feature)->None:
745         """
746         Sets the mob_material_feature attribute of the calling object to @new_mob_material_feature
747         Before setting, TypeCheck is performed.
748         """
749         if type(new_mob_material_feature)!=str:
750             raise TypeError("new_mob_material_feature must be in str")
751         self.mob_material_feature=new_mob_material_feature
752
753     def set_mob_shooting_modes(self,new_mob_shooting_modes)->None:
754         """
755         Sets the mob_shooting_modes attribute of the calling object to @new_mob_shooting_modes
756         Before setting, TypeCheck is performed.
757         """
758         if '__iter__' not in dir(type(new_mob_shooting_modes)):
759             TypeError("mob_shooting_modes must be iterable")
760         for mode in new_mob_shooting_modes:
761             if type(mode)!=str:
762                 raise TypeError("mode must be in str")
763         self.mob_shooting_modes=new_mob_shooting_modes
764     def set_mob_GPS(self,new_mob_GPS)->None:
765         """
766         Sets the mob_GPS attribute of the calling object to @new_mob_GPS
767         Before setting, TypeCheck is performed.
768         """
769         if type(new_mob_GPS)!=str:
770             raise TypeError("new_mob_GPS must be in str")
771         self.mob_GPS=new_mob_GPS
772
773     def set_mob_water_resistence_level(self,new_mob_water_resistence_level)->None:
774         """

```

```

775         Sets the mob_water_resistence_level attribute of the calling object to
776 @new_mob_water_resistence_level
777         Before setting, TypeCheck is performed.
778         """
779         if type(new_mob_water_resistence_level)!=str:
780             raise TypeError("new_mob_water_resistence_level must be in str")
781         self.mob_water_resistence_level=new_mob_water_resistence_level
782
783     def set_mob_optical_sensor_reso_in_mp(self,new_mob_optical_sensor_reso_in_mp)->None:
784         """
785         Sets the mob_optical_sensor_reso_in_mp attribute of the calling object to
786 @new_mob_optical_sensor_reso_in_mp
787         Before setting, TypeCheck and ValueCheck is performed.
788         """
789         if type(new_mob_optical_sensor_reso_in_mp)!=int:
790             raise TypeError("new_mob_optical_sensor_reso_in_mp must be in int")
791         if new_mob_optical_sensor_reso_in_mp<=0:
792             raise ValueError("new_mob_optical_sensor_reso_in_mp must be in positive")
793         self.mob_optical_sensor_reso_in_mp=new_mob_optical_sensor_reso_in_mp
794
795     def set_mob_max_display_reso(self,new_mob_max_display_reso)->None:
796         """
797         Sets the mob_max_display_reso attribute of the calling object to
798 @new_mob_max_display_reso
799         Before setting, TypeCheck is performed.
800         """
801         if type(new_mob_max_display_reso)!=str:
802             raise TypeError("new_mob_max_display_reso must be in str")
803         self.mob_max_display_reso=new_mob_max_display_reso
804
805     def set_mob_video_capture_reso_in_pixel(self,new_mob_video_capture_reso_in_pixel)->None:
806         """
807         Sets the mob_video_capture_reso_in_pixel attribute of the calling object to
808 @new_mob_video_capture_reso_in_pixel
809         Before setting, TypeCheck and ValueCheck is performed.
810         """

```

```
811     if type(new_mob_video_capture_reso_in_pixel)!=int:
812         raise TypeError("new_mob_video_capture_reso_in_pixel must be in int")
813     if new_mob_video_capture_reso_in_pixel<=0:
814         raise ValueError("new_mob_video_capture_reso_in_pixel must be positive")
815     self.mob_video_capture_reso_in_pixel=new_mob_video_capture_reso_in_pixel
816 def show_details(self)->None:
817     """
818     This function display all the characterstics of Mobile class
819     """
820     print("GPU:{}".format(self.mob_GPU))
821     print("RAM:{}".format(self.mob_RAM_in_gb))
822     print("Product Dimensions:{}".format(self.mob_prod_dimensions.__dict__))
823     print("Batteries:{}".format(self.mob_battries))
824     print("Wireless Communication technologies:{}".format(self.mob_wireless_commu_tech))
825     print("Spacial Features:{}".format(self.mob_special_feature))
826     print("Display technology:{}".format(self.mob_display_tech))
827     print("Manufacturer:{}".format(self.mob_manufacturer))
828     print("Country of origin:{}".format(self.mob_country_of_origin))
829     print("Colour:{}".format(self.mob_colour))
830     print("Screen Size:{}".format(self.mob_screen_size_in_inches))
831     print("Connector Type:{}".format(self.mob_connector_type))
832     print("From Photo Sensor Resolution:{}".format(self.mob_front_photo_sensor_reso_in_mp))
833     print("From Factor:{}".format(self.mob_from_factor))
834     print("Battery Capacity:{}".format(self.mob_battries_capacity_in_MH))
835     print("Rear Camera Reso:{}".format(self.mob_rear_camera_reso_in_mp))
836     print("Model Year:{}".format(self.mob_model_year))
837     print("CPU Model:{}".format(self.mob_CPU_model))
838     print("Included Componentes:{}".format(self.mob_included_components))
839     print("Display Type:{}".format(self.mob_display_type))
840     print("Human Interface Input:{}".format(self.mob_human_interface_input))
841     print("Battery Description:{}".format(self.mob_batteries_desc))
842     print("SIM card size:{}".format(self.mob_sim_card_size))
843     print("Material Feature:{}".format(self.mob_material_feature))
844     print("Shooting Modes:{}".format(self.mob_shooting_modes))
845     print("GPS:{}".format(self.mob_GPS))
```



```
846     print("Water Resistance Level:{}".format(self.mob_water_resistence_level))
847     print("Optical Sensor Resolution:{}".format(self.mob_optical_sensor_reso_in_mp))
848     print("Display Resolution Maximum:{}".format(self.mob_max_display_reso))
849     print("Video Capture Resolution:{}".format(self.mob_video_capture_reso_in_pixel))
850
851 def main():
852     mob_obj=Mobile(
853         "Qualcomm",
854         8,
855         ProductDimension(7.6,0.8,16.6,195.0),
856         "1 Lithium Polymer batteries required(included)",
857         "Cellular",
858         ["Rear Camera,Camera"],
859         "AMOLED",
860         "Wireless",
861         "Oppo Mobile India Private Limited",
862         "India",
863         "Pastel Lime",
864         6.72,
865         "USB Type C",
866         2,
867         "SmartPhone",
868         5000,
869         2,
870         2023,
871         "Snapdragon",
872         ["SIM Tray Ejector","Adapter","Phone Case","USB Cable"],
873         "LCD",
874         "Keyboard",
875         "Lithium-Ion",
876         "Nano",
877         "plastic",
878         ["Macro","Portrait"],
879         "GLONASS",
880         "Water Resistant",
```

```
881         2,  
882         "1080*2400 Pixels",  
883         1080  
884     )  
885     print("MOBILE PRODUCT DETAILS:")  
886     mob_obj.show_details()  
887     #we can also get the attribute using getter method and  
888     #set the specific attribute using setter method  
889     sys.exit(0)  
890 main()
```