Class Design Strategy In Python

American Tourister Class

```
---------------------------------------------------------------------------------------------------------------------
"""
   @Author:Kunal Narkhede
   @Date:28/12/2023
   @Goal:To implement class Bag
   Capture Real Life Product on Amazon
   link:-http://surl.li/orkre
 """
---------------------------------------------------------------------------------------------------------------------
import sys
class ProductDimension:
    """
        This class implement the Dimension of Bag
        @__init__(self, length: float, width: float, height: float, weight_in_gm: float):
        Constructor
        @get_length(self)
            getter of attribute length
        @get_width(self)
            getter of attribute width
        @get_height(self)
            getter of attribute height
        @get_weight_in_gm(self)
            getter of attribute weight_in_gm
        ------------------------------------
        @set_length(self):
            setter of attribute length
        @set_width(self):
            setter of attribute width
        @set_height(self):
            setter of attribute height
        @set_weight_in_gm(self):
            setter of attribute weight_in_gm
```

```
    """
    def __init__(
            self,
            length:float,
            width:float,
            height:float,
            weight_in_gm:float
      ):

        """
            Constructor of ProductDimension class:
            @__init__(self, length: float, width: float, height: float, weight_in_gm: float):

            @self:newly created class object of ProductDimension
            @length:Client specified value for attribute length
            @width:Client specified value for attribute width
            @height:Client specified value for attribute height
            @weight_in_gm:Client specified value for attribute weight_in_gm

        """
        if type(length)!=float:
            raise TypeError("Bad type:length")
        if type(width)!=float:
            raise TypeError("Bad type:width")
        if type(height)!=float:
            raise TypeError("Bad type:height")
        if type(weight_in_gm)!=float:
            raise TypeError("Bad type:weight_in_gm")
        if length<=0.0:
            raise ValueError("Length must be positive")
        if width<=0.0:
            raise ValueError("Width must be positive")
        if height<=0.0:
            raise ValueError("Height must be positive")
        if weight_in_gm<=0.0:
```

```
            raise ValueError("weight_in_gm must be positive")

        self.length=length
        self.width=width
        self.height=height
        self.weight_in_gm=weight_in_gm

    #getter method

    def get_length(self) -> float:
        """
            Returns the length attribute of the calling object
        """
        return self.length

    def get_width(self) -> float:
        """
            Returns the width attribute of the calling object
        """
        return self.width

    def get_height(self) -> float:
        """
            Returns the height attribute of the calling object
        """
        return self.height

    def get_weight_in_gm(self) -> float:
        """
            Returns the weight_in_gm attribute of the calling object
        """
        return self.weight_in_gm

    #setter method

```

```
    def set_length(self,new_length:float):
        """
            Sets the length attribute of the calling object to @new_length
            Before setting, TypeCheck and ValueCheck is performed.
        """
        if type(new_length)!=float:
            raise TypeError("new_length must be an float")
        if new_length <= 0.0:
            raise TypeError("new_length must be positive")
        self.length=new_length

    def set_width(self,new_width:float):
        """
            Sets the width attribute of the calling object to @new_width
            Before setting, TypeCheck and ValueCheck is performed.
        """
        if type(new_width)!=float:
            raise TypeError("new_width must be an float")
        if new_width <= 0.0:
            raise ValueError("new_width must be positive")
        self.width=new_width

    def set_height(self,new_height:float):
        """
            Sets the height attribute of the calling object to @new_height
            Before setting, TypeCheck and ValueCheck is performed.
        """
        if type(new_height)!=float:
            raise TypeError("new_height must be an float")
        if new_height <= 0.0 :
            raise ValueError("new_height must be positive")
        self.height=new_height

    def set_weight_in_gm(self,new_weight_in_gm:float):
        """
```

```
140             Sets the weight_in_gm attribute of the calling object to @new_weight_in_gm
141             Before setting, TypeCheck and ValueCheck is performed.
142         """
143         if type(new_weight_in_gm)!=float:
144             raise TypeError("new_weight_in_gm must be an float")
145         if new_weight_in_gm <= 0.0:
146             raise ValueError("new_weight_in_gm must be positive")
147         self.weight_in_gm=new_weight_in_gm
148 
149 class Bag:
150     def __init__(self,
151                  bag_colour:str,
152                  bag_model_number:str,
153                  bag_prod_dimensions:ProductDimension,
154                  bag_primary_material:str,
155                  bag_capacity:str,
156                  bag_volume_capacity_in_lt:int,
157                  bag_warranty_desc:str,
158                  bag_shape:str,
159                  bag_manufacturer:str,
160                  bag_ASIN:str,
161                  bag_country_of_origin:str,
162                  bag_cust_reviews:float,
163                  bag_best_seller_rank:str,
164                  bag_date_first_available:str,
165                  bag_generic_name:str
166                  ):
167 
168             if type(bag_colour)!=str:
169                 raise TypeError("bag_colour must be in str")
170             if type(bag_model_number)!=str:
171                 raise TypeError("bag_model_number must be in str")
172             if type(bag_prod_dimensions)!=ProductDimension:
173                 raise TypeError("bag_prod_dimensions must be in str")
174             if type(bag_primary_material)!=str:
175                 raise TypeError("bag_primary_material must be in str")
```

```
        if type(bag_capacity)!=str:
            raise TypeError("bag_capacity must be in str")
        if type(bag_volume_capacity_in_lt)!=int:
            raise TypeError("bag_volume_capacity_in_lt must be in str")
        if bag_volume_capacity_in_lt<=0:
            raise ValueError("bag_volume_capacity_in_lt must be positive")
        if type(bag_warranty_desc)!=str:
            raise TypeError("bag_warranty_desc must be in str")
        if type(bag_shape)!=str:
            raise TypeError("bag_shape must be in str")
        if type(bag_manufacturer)!=str:
            raise TypeError("bag_manufacturer must be in str")
        if type(bag_ASIN)!=str:
            raise TypeError("bag_ASIN must be in str")
        if type(bag_country_of_origin)!=str:
            raise TypeError("bag_country_of_origin must be in str")
        if type(bag_cust_reviews)!=float:
            raise TypeError("bag_cust_reviews must be in float")
        if bag_cust_reviews<=0.0:
            raise ValueError("bag_cust_reviews must be in positve")
        if type(bag_best_seller_rank)!=str:
            raise TypeError("bag_best_seller_rank must be in str")
        if type(bag_date_first_available)!=str:
            raise TypeError("bag_date_first_available must be in str")
        if type(bag_generic_name)!=str:
            raise TypeError("bag_generic_name must be inn str")


        self.bag_colour=bag_colour
        self.bag_model_number=bag_model_number
        self.bag_prod_dimensions=bag_prod_dimensions
        self.bag_primary_material=bag_primary_material
        self.bag_capacity=bag_capacity
        self.bag_volume_capacity_in_lt=bag_volume_capacity_in_lt
        self.bag_warranty_desc=bag_warranty_desc
```

```
211             self.bag_shape=bag_shape
212             self.bag_manufacturer=bag_manufacturer
213             self.bag_ASIN=bag_ASIN
214             self.bag_country_of_origin=bag_country_of_origin
215             self.bag_cust_reviews=bag_cust_reviews
216             self.bag_best_seller_rank=bag_best_seller_rank
217             self.bag_date_first_available=bag_date_first_available
218             self.bag_generic_name=bag_generic_name
219
220     #Getter
221
222     def get_bag_colour(self)->str:
223         """
224             Returns the bag_colour attribute of the calling object
225         """
226         return self.bag_colour
227
228     def get_bag_model_number(self)->str:
229         """
230             Returns the bag_model_number attribute of the calling object
231         """
232         return self.bag_model_number
233
234     def get_bag_prod_dimensions(self)->ProductDimension:
235         """
236             Returns the bag_prod_dimensions attribute of the calling object
237         """
238         return self.bag_prod_dimensions
239
240     def get_bag_primary_material(self)->str:
241         """
242             Returns the bag_primary_material attribute of the calling object
243         """
244         return self.bag_primary_material
245
```

```
    def get_bag_capacity(self)->str:
        """
            Returns the bag_capacity attribute of the calling object
        """
        return self.bag_capacity

    def get_bag_volume_capacity_in_lt(self)->int:
        """
            Returns the bag_volume_capacity_in_lt attribute of the calling object
        """
        return self.bag_volume_capacity_in_lt

    def get_bag_warranty_desc(self)->str:
        """
            Returns the bag_warranty_desc attribute of the calling object
        """
        return self.bag_warranty_desc

    def get_bag_shape(self)->str:
        """
            Returns the bag_shape attribute of the calling object
        """
        return self.bag_shape

    def get_bag_manufacturer(self)->str:
        """
            Returns the bag_manufacturer attribute of the calling object
        """
        return self.bag_manufacturer

    def get_bag_ASIN(self)->str:
        """
            Returns the bag_ASIN attribute of the calling object
        """
        return self.bag_ASIN
```

```
281
282     def get_bag_country_of_origin(self)->str:
283         """
284             Returns the bag_country_of_origin attribute of the calling object
285         """
286         return self.bag_country_of_origin
287
288     def get_bag_cust_reviews(self)->float:
289         """
290             Returns the bag_cust_reviews attribute of the calling object
291         """
292         return self.bag_cust_reviews
293
294     def get_bag_best_seller_rank(self)->str:
295         """
296             Returns the bag_best_seller_rank attribute of the calling object
297         """
298         return self.bag_best_seller_rank
299
300     def get_bag_date_first_available(self)->str:
301         """
302             Returns the bag_date_first_available attribute of the calling object
303         """
304         return self.bag_date_first_available
305
306     def get_bag_generic_name(self)->str:
307         """
308             Returns the bag_generic_name attribute of the calling object
309         """
310         return self.bag_generic_name
311
312
313     #Setter Method
314
315     def set_bag_colour(self,new_bag_colour)->None:
```

```
316         """
317             Sets the bag_colour attribute of the calling object to @new_bag_colour
318             Before setting, TypeCheck is performed and this function returns nothing
319         """
320         if type(new_bag_colour)!=str:
321             raise TypeError("new_bag_colour must be in str")
322         self.bag_colour=new_bag_colour
323 
324     def set_bag_model_number(self,new_bag_model_number)->str:
325         """
326             Sets the bag_model_number attribute of the calling object to @new_bag_model_number
327             Before setting, TypeCheck is performed and this function returns nothing
328         """
329         if type(new_bag_model_number)!=str:
330             raise TypeError("new_bag_model_number must be in str")
331         self.bag_model_number=new_bag_model_number
332 
333     def set_bag_prod_dimensions(self,new_bag_prod_dimensions)->None:
334         """
335             Sets the bag_prod_dimensions attribute of the calling object to @new_bag_prod_dimensions
336             Before setting, TypeCheck is performed and this function returns nothing
337         """
338         if type(new_bag_prod_dimensions)!=ProductDimension:
339             raise TypeError("new_bag_prod_dimensions must be in ProductDimension")
340         self.bag_prod_dimensions=new_bag_prod_dimensions
341 
342     def set_bag_primary_material(self,new_bag_primary_material)->None:
343         """
344             Sets the bag_primary_material attribute of the calling object to @new_bag_primary_material
345             Before setting, TypeCheck is performed and this function returns nothing
346         """
347         if type(new_bag_primary_material)!=str:
348             raise TypeError("new_bag_primary_material must be in str")
349         self.bag_primary_material=new_bag_primary_material
350 
```

```
    def set_bag_capacity(self,new_bag_capacity):
        """
            Sets the bag_capacity attribute of the calling object to @new_bag_capacity
            Before setting, TypeCheck is performed and this function returns nothing
        """
        if type(new_bag_capacity)!=str:
            raise TypeError("new_bag_capacity must be in str")
        self.bag_capacity=new_bag_capacity

    def set_bag_volume_capacity_in_lt(self,new_bag_volume_capacity_in_lt)->None:
        """
            Sets the bag_volume_capacity_in_lt attribute of the calling object to
@new_bag_volume_capacity_in_lt
            Before setting, TypeCheck and ValueCheck is performed.and this
            function returns nothing
        """
        if type(new_bag_volume_capacity_in_lt)!=int:
            raise TypeError("new_bag_volume_capacity_in_lt must be in int")
        if new_bag_volume_capacity_in_lt<=0:
            raise  ValueError("new_bag_volume_capacity_in_lt must be positive")
        self.bag_volume_capacity_in_lt=new_bag_volume_capacity_in_lt

    def set_bag_warranty_desc(self,new_bag_warranty_desc)->str:
        """
            Sets the bag_warranty_desc attribute of the calling object to @new_bag_warranty_desc
            Before setting, TypeCheck is performed and this function returns nothing
        """
        if type(new_bag_warranty_desc)!=str:
            raise TypeError("new_bag_warranty_desc must be in str")
        self.bag_warranty_desc=new_bag_warranty_desc

    def set_bag_shape(self,new_bag_shape)->None:
        """
            Sets the bag_shape attribute of the calling object to @new_bag_shape
            Before setting, TypeCheck is performed and this function returns nothing
        """
```

```
        if type(new_bag_shape)!=str:
            raise TypeError("new_bag_shape must be in str")
        self.bag_shape=new_bag_shape

    def set_bag_manufacturer(self,new_bag_manufacturer)->None:
        """
            Sets the bag_manufacturer attribute of the calling object to @new_bag_manufacturer
            Before setting, TypeCheck is performed and this function returns nothing
        """
        if type(new_bag_manufacturer)!=str:
            raise TypeError("new_bag_manufacturer must be in str")
        self.bag_manufacturer=new_bag_manufacturer

    def set_bag_ASIN(self,new_bag_ASIN)->None:
        """
            Sets the bag_ASIN attribute of the calling object to @new_bag_ASIN
            Before setting, TypeCheck is performed and this function returns nothing
        """
        if type(new_bag_ASIN)!=str:
            raise TypeError("new_bag_ASIN must be in str")
        self.bag_ASIN=new_bag_ASIN

    def set_bag_country_of_origin(self,new_bag_country_of_origin)->None:
        """
            Sets the bag_country_of_origin attribute of the calling object to @new_bag_country_of_origin
            Before setting, TypeCheck is performed and this function returns nothing
        """
        if type(new_bag_country_of_origin)!=str:
            raise TypeError("new_bag_country_of_origin must be in str")
        self.bag_country_of_origin=new_bag_country_of_origin

    def set_bag_cust_reviews(self,new_bag_cust_reviews)->None:
        """
            Sets the bag_cust_reviews attribute of the calling object to @new_bag_cust_reviews
            Before setting, TypeCheck and ValueCheck is performed.and this
```

```
            function returns nothing
        """
        if type(new_bag_cust_reviews)!=float:
            raise TypeError("new_bag_cust_reviews must be in float")
        if new_bag_cust_reviews<=0.0:
            raise ValueError("new_bag_cust_reviews must be positive")
        self.bag_cust_reviews=new_bag_cust_reviews

    def set_bag_best_seller_rank(self,new_bag_best_seller_rank)->None:
        """
            Sets the bag_best_seller_rank attribute of the calling object to @new_bag_best_seller_rank
            Before setting, TypeCheck is performed and this function returns nothing
        """
        if type(new_bag_best_seller_rank)!=str:
            raise TypeError("new_bag_best_seller_rank must be in str")
        self.bag_best_seller_rank=new_bag_best_seller_rank

    def set_bag_date_first_available(self,new_bag_date_first_available)->None:
        """
            Sets the bag_date_first_available attribute of the calling object to
@new_bag_date_first_available
            Before setting, TypeCheck is performed and this function returns nothing
        """
        if type(new_bag_date_first_available)!=str:
            raise TypeError("new_bag_date_first_available must be in str")
        self.bag_date_first_available=new_bag_date_first_available

    def set_bag_generic_name(self,new_bag_generic_name)->None:
        """
            Sets the bag_generic_name attribute of the calling object to @new_bag_generic_name
            Before setting, TypeCheck is performed and this function returns nothing
        """
        if type(new_bag_generic_name)!=str:
            raise TypeError("new_bag_generic_name must be in str")
        self.bag_generic_name=new_bag_generic_name

```

```
458     def show_bag_details(self)->None:
459         print("Bag Colour:{}".format(self.bag_colour))
460         print("Bag Model Number:{}".format(self.bag_model_number))
461         print("Bag Dimensions:{}".format(self.bag_prod_dimensions.__dict__))
462         print("Bag Primary Material:{}".format(self.bag_primary_material))
463         print("Bag Capacity:{}".format(self.bag_capacity))
464         print("Bag Volume Capacity in Liter:{}".format(self.bag_volume_capacity_in_lt))
465         print("Bag Warrenty Description:{}".format(self.bag_warranty_desc))
466         print("Bag Shape:{}".format(self.bag_shape))
467         print("Bag Manufacturer:{}".format(self.bag_manufacturer))
468         print("Bag ASIN:{}".format(self.bag_ASIN))
469         print("Bag Country of Origin:{}".format(self.bag_country_of_origin))
470         print("Bag Customer Reviews:{}".format(self.bag_cust_reviews))
471         print("Bag Best Seller Rank:{}".format(self.bag_best_seller_rank))
472         print("Bag Date Of First Available:{}".format(self.bag_date_first_available))
473         print("Bag Generic Name:{}".format(self.bag_generic_name))
474 
475 
476 def main():
477 
478     bag_obj=Bag(
479             "Blue",
480             "AMT FIZZ SCH BAG 02 - BLUE",
481             ProductDimension(22.0,31.5,49.5,400.0),
482             "Polyester",
483             "L",
484             32,
485             "1 year",
486             "Rectangular",
487             "Samsonite",
488             "B07CG8BFPP",
489             "India",
490             4.0,
491             "21 in Bags",
492             "18-April-2018",
493             "Backpack"
```

```
        )
    print("BAG PRODUCT DETAILS:")
    bag_obj.show_bag_details()
    #we can also get the attribute using getter method and
    #set the specific attribute using setter method
    sys.exit(0)
main()

"""
Output:-

BAG PRODUCT DETAILS:
Bag Colour:Blue
Bag Model Number:AMT FIZZ SCH BAG 02 - BLUE
Bag Dimensions:{'length': 22.0, 'width': 31.5, 'height': 49.5, 'weight_in_gm': 400.0}
Bag Primary Material:Polyester
Bag Capacity:L
Bag Volume Capacity in Liter:32
Bag Warrenty Description:1 year
Bag Shape:Rectangular
Bag Manufacturer:Samsonite
Bag ASIN:B07CG8BFPP
Bag Country of Origin:India
Bag Customer Reviews:4.0
Bag Best Seller Rank:21 in Bags
Bag Date Of First Available:18-April-2018
Bag Generic Name:Backpack
"""
```