Class Design Strategy In Python

OnePlus Mobile Class

```
-----------------------------------------------------------------------------------------------------------------------
"""
    @Author:Kunal Narkhede
    @Date:24/12/2023
    @Goal:To implement class Mobile
    Capture Real Life Product on Amazon
    http://surl.li/ooqwc
 """
-----------------------------------------------------------------------------------------------------------------------

import sys
class ProductDimension:
    """
        this class implement the Dimension of Mobile
        @__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):
        """
            Sets the weight_in_gm attribute of the calling object to @new_weight_in_gm
            Before setting, TypeCheck and ValueCheck is performed.
        """
        if type(new_weight_in_gm)!=float:
            raise TypeError("new_weight_in_gm must be an float")
        if new_weight_in_gm <= 0.0:
            raise ValueError("new_weight_in_gm must be positive")
        self.weight_in_gm=new_weight_in_gm


class Mobile:

    def __init__(self,
                mob_GPU:str,
                mob_RAM_in_gb:int,
                mob_prod_dimensions:ProductDimension,
                mob_battries:str,
                mob_wireless_commu_tech:[str],
                mob_special_feature:[str],
                mob_display_tech:str,
                mob_manufacturer:str,
                mob_country_of_origin:str,
                mob_connectivity_tech:[str],
                mob_colour:str,
                mob_screen_size_in_inches:float,
                mob_connector_type:str,
                mob_front_photo_sensor_reso_in_mp:int,
                mob_from_factor:str,
                mob_battries_capacity_in_MH:int,
                mob_rear_camera_reso_in_mp:int,
                mob_model_year:int,
                mob_CPU_model:str,
                mob_included_components:[str],
            mob_display_type:str,
            mob_human_interface_input:str,
            mob_batteries_desc:str,
            mob_sim_card_size:str,
            mob_material_feature:str,
            mob_shooting_modes:[str],
            mob_GPS:str,
            mob_water_resistence_level:str,
            mob_optical_sensor_reso_in_mp:int,
            mob_max_display_reso:str,
            mob_video_capture_reso_in_pixel:int
            ):


        if type(mob_GPU)!=str:
            raise TypeError("mob_GPU must be in str")
        if type(mob_RAM_in_gb)!=int:
            raise TypeError("mob_RAM_in_gb must be in int")
        if mob_RAM_in_gb<=0:
            raise ValueError("mob_RAM_in_gb must be positive")
        if type(mob_prod_dimensions)!=ProductDimension:
            raise TypeError("mob_prod_dimensions must be in ProductDimension")
        if type(mob_battries)!=str:
            raise TypeError("mob_battries must be in str")
        if '__iter__' not in dir(type(mob_wireless_commu_tech)):
            raise TypeError("mob_wireless_commu_tech must be iterable")
        for technology in mob_wireless_commu_tech:
            if type(technology)!=str:
                raise TypeError("technology must be in str")
        if '__iter__' not in dir(type(mob_special_feature)):
            raise TypeError("mob_special_feature must be iterable")
        for feature in mob_special_feature:
            if type(feature)!=str:
                raise TypeError("feature must be str")
        if type(mob_display_tech)!=str:
            raise TypeError("mob_display_tech must be str")
        if type(mob_manufacturer)!=str:
            raise TypeError("mob_manufacturer must be in str")
        if type(mob_country_of_origin)!=str:
            raise TypeError("mob_country_of_origin must be in str")
        if '__iter__' not in dir(type(mob_connectivity_tech)):
            raise TypeError("mob_connectivity_tech must be iterable")
        for technology in mob_connectivity_tech:
            if type(technology)!=str:
                raise TypeError("technology must be in str")
        if type(mob_colour)!=str:
            raise TypeError("mob_colour muse be in str")
        if type(mob_screen_size_in_inches)!=float:
            raise TypeError("mob_screen_size_in_inches must be in float")
        if type(mob_connector_type)!=str:
            raise TypeError("mob_connector_type must be in str")
        if type(mob_front_photo_sensor_reso_in_mp)!=int:
            raise TypeError("mob_front_photo_sensor_reso_in_mp must be in int")
        if mob_front_photo_sensor_reso_in_mp<=0:
            raise ValueError("mob_front_photo_sensor_reso_in_mp must be positive")
        if type(mob_from_factor)!=str:
            raise TypeError("mob_from_factor must be in str")
        if type(mob_battries_capacity_in_MH)!=int:
            raise TypeError("mob_battries_capacity_in_MH must be in int")
        if mob_battries_capacity_in_MH<=0:
            raise ValueError("mob_battries_capacity_in_MH must be positive")
        if type(mob_rear_camera_reso_in_mp)!=int:
            raise TypeError("mob_rear_camera_reso_in_mp must be int")
        if type(mob_model_year)!=int:
            raise TypeError("mob_model_year must be in int")
        if mob_model_year<=0:
            raise ValueError("mob_model_year must be in positive")
        if type(mob_CPU_model)!=str:
            raise TypeError("mob_CPU_model must be in str")
        if mob_rear_camera_reso_in_mp<=0:
            raise ValueError("mob_rear_camera_reso_in_mp must be in positive")
        if '__iter__' not in dir(type(mob_included_components)):
            raise TypeError("mob_included_components must be iterable")
        for componant in mob_included_components:
            if type(componant)!=str:
                raise TypeError("componant must be str")
        if type(mob_display_type)!=str:
            raise TypeError("mob_display_type must be in str")
        if type(mob_human_interface_input)!=str:
            raise TypeError("mob_human_interface_input must be str")
        if type(mob_batteries_desc)!=str:
            raise TypeError("mob_batteries_desc must be in str")
        if type(mob_sim_card_size)!=str:
            raise TypeError("mob_sim_card_size must be in str")
        if type(mob_material_feature)!=str:
            raise TypeError("mob_material_feature must be in str")
        if '__iter__' not in dir(type(mob_shooting_modes)):
            raise TypeError("mob_shooting_modes must be iterable")
        for mode in mob_shooting_modes:
            if type(mode)!=str:
                raise TypeError("mode must be in str")
        if type(mob_water_resistence_level)!=str:
            raise TypeError("mob_water_resistence_level must be in str")
        if type(mob_optical_sensor_reso_in_mp)!=int:
            raise TypeError("mob_optical_sensor_reso_in_mp must be in int")
        if mob_optical_sensor_reso_in_mp<=0:
            raise ValueError("mob_optical_sensor_reso_in_mp must be positive")
        if type(mob_max_display_reso)!=str:
            raise TypeError("mob_max_display_reso must be in str")
        if type(mob_video_capture_reso_in_pixel)!=int:
            raise TypeError("mob_video_capture_reso_in_pixel must be in int")
        if mob_video_capture_reso_in_pixel<=0:
            raise ValueError("mob_video_capture_reso_in_pixel must be positive")

        self.mob_GPU=mob_GPU
        self.mob_RAM_in_gb=mob_RAM_in_gb
        self.mob_prod_dimensions=mob_prod_dimensions
        self.mob_battries=mob_battries
        self.mob_wireless_commu_tech=mob_wireless_commu_tech
        self.mob_special_feature=mob_special_feature
        self.mob_display_tech=mob_display_tech
        self.mob_manufacturer=mob_manufacturer
        self.mob_country_of_origin=mob_country_of_origin
        self.mob_connectivity_tech=mob_connectivity_tech
        self.mob_colour=mob_colour
        self.mob_screen_size_in_inches=mob_screen_size_in_inches
        self.mob_connector_type=mob_connector_type
        self.mob_front_photo_sensor_reso_in_mp=mob_front_photo_sensor_reso_in_mp
        self.mob_from_factor=mob_from_factor
        self.mob_battries_capacity_in_MH=mob_battries_capacity_in_MH
        self.mob_rear_camera_reso_in_mp=mob_rear_camera_reso_in_mp
        self.mob_model_year=mob_model_year
        self.mob_CPU_model=mob_CPU_model
        self.mob_included_components=mob_included_components
        self.mob_display_type=mob_display_type
        self.mob_human_interface_input=mob_human_interface_input
        self.mob_batteries_desc=mob_batteries_desc
        self.mob_sim_card_size=mob_sim_card_size
        self.mob_material_feature=mob_material_feature
        self.mob_shooting_modes=mob_shooting_modes
        self.mob_GPS=mob_GPS
        self.mob_water_resistence_level=mob_water_resistence_level
        self.mob_optical_sensor_reso_in_mp=mob_optical_sensor_reso_in_mp
        self.mob_max_display_reso=mob_max_display_reso
        self.mob_video_capture_reso_in_pixel=mob_video_capture_reso_in_pixel


    #getter method

    def get_mob_GPU(self)->str:
        """
            Returns the mob_GPU attribute of the calling object
        """
        return self.mob_GPU
    def get_mob_RAM_in_gb(self)->int:
        """
            Returns the mob_RAM_in_gb attribute of the calling object
        """
        return self.mob_RAM_in_gb

    def get_mob_prod_dimensions(self)->ProductDimension:
        """
            Returns the mob_prod_dimensions attribute of the calling object
        """
        return self.mob_prod_dimensions
    def get_mob_battries(self)->str:
        """
            Returns the mob_battries attribute of the calling object
        """
        return self.mob_battries

    def get_mob_wireless_commu_tech(self)->[str]:
        """
            Returns the mob_wireless_commu_tech attribute of the calling object
        """
        return self.mob_wireless_commu_tech

    def get_mob_special_feature(self)->[str]:
        """
            Returns the mob_special_feature attribute of the calling object
        """
        return self.mob_special_feature

    def get_mob_display_tech(self)->str:
        """
```

```
            Returns the mob_RAM_in_gb attribute of the calling object
        """
        return self.mob_display_tech

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

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

    def get_mob_connectivity_tech(self)->[str]:
        """
            Returns the mob_connectivity_tech attribute of the calling object
        """
        return self.mob_connectivity_tech


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

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

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

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

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

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

    def get_mob_rear_camera_reso_in_mp(self)->int:
        """
            Returns the mob_rear_camera_reso_in_mp attribute of the calling object
        """
        return self.mob_rear_camera_reso_in_mp
    def get_mob_model_year(self)->int:
        """
            Returns the mob_model_year attribute of the calling object
        """
        return self.mob_model_year

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

    def get_mob_included_components(self)->[str]:
        """
            Returns the mob_included_components attribute of the calling object
        """
        return self.mob_included_components

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

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

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

    def get_mob_sim_card_size(self)->str:
        """
            Returns the mob_sim_card_size attribute of the calling object
        """
        return self.mob_sim_card_size
    def get_mob_material_feature(self)->str:
        """
            Returns the mob_material_feature attribute of the calling object
        """
        return self.mob_material_feature

    def get_mob_shooting_modes(self)->[str]:
        """
            Returns the mob_shooting_modes attribute of the calling object
        """
        return self.mob_shooting_modes
    def get_mob_GPS(self)->[str]:
        """
            Returns the mob_GPS attribute of the calling object
        """
        return self.mob_GPS
    def get_mob_water_resistence_level(self)->str:
        """
            Returns the mob_water_resistence_level attribute of the calling object
        """
        return self.mob_water_resistence_level

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

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

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

    #Setter Method
    def set_mob_GPU(self,new_mob_GPU)->None:
        """
            Sets the mob_GPU attribute of the calling object to @new_mob_GPU
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_GPU)!=str:
            raise TypeError("new_mob_GPU must be str")
        self.mob_GPU=new_mob_GPU

    def set_mob_RAM_in_gb(self,new_mob_RAM_in_gb)->None:
        """
            Sets the mob_RAM_in_gb attribute of the calling object to @new_mob_RAM_in_gb
            Before setting, TypeCheck and ValueCheck is performed.
        """
        if type(new_mob_RAM_in_gb)!=int:
            raise TypeError("new_mob_RAM_in_gb must be int")
        if new_mob_RAM_in_gb <= 0:
            raise ValueError("new_mob_RAM_in_gb must be positive")
        self.mob_RAM_in_gb=new_mob_RAM_in_gb

    def set_mob_prod_dimensions(self,new_mob_prod_dimensions)->None:
        """
            Sets the mob_prod_dimensions attribute of the calling object to @new_mob_prod_dimensions
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_prod_dimensions)!=ProductDimension:
            raise TypeError("mob_prod_dimensions must be ProductDimension")
        self.mob_prod_dimensions=new_mob_prod_dimensions

    def set_mob_battries(self,new_mob_battries)->None:
        """
            Sets the mob_battries attribute of the calling object to @new_mob_battries
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_battries)!=str:
            raise TypeError("mob_battries must be str")
        self.mob_battries=new_mob_battries

    def set_mob_wireless_commu_tech(self,new_mob_wireless_commu_tech)->None:
        """
            Sets the mob_wireless_commu_tech attribute of the calling object to
@new_mob_wireless_commu_tech
            Before setting, TypeCheck is performed.
        """
        if '__iter__' not in dir(type(new_mob_wireless_commu_tech)):
            raise TypeError("new_mob_wireless_commu_tech must be iterable")
        for technology in new_mob_wireless_commu_tech:
            if type(technology)!=str:
                raise TypeError("technology must be in str")
        self.mob_wireless_commu_tech=new_mob_wireless_commu_tech

    def set_mob_special_feature(self,new_mob_special_feature)->None:
        """
            Sets the mob_special_feature attribute of the calling object to @new_mob_special_feature
            Before setting, TypeCheck is performed.
        """
        if '__iter__' not in dir(type(new_mob_special_feature)):
            raise TypeError("new_mob_special_feature must be iterable")
        for feature in new_mob_special_feature:
            if type(feature)!=str:
                raise TypeError("feature must be in str")
        self.mob_special_feature=new_mob_special_feature

    def set_mob_display_tech(self,new_mob_display_tech)->None:
        """
            Sets the mob_display_tech attribute of the calling object to @new_mob_display_tech
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_display_tech)!=str:
            raise TypeError("new_mob_display_tech must be str")
        self.mob_display_tech=new_mob_display_tech
    def set_mob_manufacturer(self,new_mob_manufacturer)->None:
        """
            Sets the mob_manufacturer attribute of the calling object to @new_mob_manufacturer
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_manufacturer)!=str:
            raise TypeError("new_mob_manufacturermust be in str")
        self.mob_manufacturer=new_mob_manufacturer

    def set_mob_country_of_origin(self,new_mob_country_of_origin)->None:
        """
            Sets the mob_country_of_origin attribute of the calling object to @new_mob_country_of_origin
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_country_of_origin)!=str:
            raise TypeError("new_mob_country_of_origin must be in str")
        self.mob_country_of_origin=new_mob_country_of_origin

    def set_mob_connectivity_tech(self,new_mob_connectivity_tech)->None:
        """
            Sets the mob_connectivity_tech attribute of the calling object to @new_mob_connectivity_tech
            Before setting, TypeCheck is performed.
        """
        if '__iter__'not in dir(type(new_mob_connectivity_tech)):
            raise TypeError("new_mob_connectivity_tech must be in iterable")
        for technology in new_mob_connectivity_tech:
            if type(technology)!=str:
```

```
            raise TypeError("technology must be in str")
        self.mob_connectivity_tech=new_mob_connectivity_tech

    def set_mob_colour(self,new_mob_colour)->None:
        """
            Sets the mob_colour attribute of the calling object to @new_mob_colour
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_colour)!=str:
            raise TypeError("new_mob_colour must be in str")
        self.mob_colour=new_mob_colour

    def set_mob_screen_size_in_inches(self,new_mob_screen_size_in_inches)->None:
        """
            Sets the mob_screen_size_in_inches attribute of the calling object to @new_mob_screen_size_in_inches
            Before setting, TypeCheck and ValueCheck is performed.
        """
        if type(new_mob_screen_size_in_inches)!=float:
            raise TypeError("new_mob_screen_size_in_inches must be in float")
        if new_mob_screen_size_in_inches<=0.0:
            raise ValueError("new_mob_screen_size_in_inches must be in positive")
        self.mob_screen_size_in_inches=new_mob_screen_size_in_inches

    def set_mob_connector_type(self,new_mob_connector_type)->None:
        """
            Sets the mob_connector_type attribute of the calling object to @new_mob_connector_type
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_connector_type)!=str:
            raise TypeError("new_mob_connector_type must be in str")
        self.mob_connector_type=new_mob_connector_type

    def set_mob_front_photo_sensor_reso_in_mp(self,new_mob_front_photo_sensor_reso_in_mp)->None:
        """
```

```
            Sets the mob_front_photo_sensor_reso_in_mp attribute of the calling object to
@new_mob_front_photo_sensor_reso_in_mp
            Before setting, TypeCheck and ValueCheck is performed.
        """
        if type(new_mob_front_photo_sensor_reso_in_mp)!=int:
            raise TypeError("new_mob_front_photo_sensor_reso_in_mp must be in int")
        if new_mob_front_photo_sensor_reso_in_mp<=0:
            raise ValueError("new_mob_front_photo_sensor_reso_in_mp must positive")
        self.mob_front_photo_sensor_reso_in_mp=new_mob_front_photo_sensor_reso_in_mp

    def set_mob_from_factor(self,new_mob_from_factor)->None:
        """
            Sets the mob_from_factor attribute of the calling object to @new_mob_from_factor
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_from_factor)!=str:
            raise TypeError("new_mob_from_factor must be in str")
        self.mob_from_factor=new_mob_from_factor

    def set_mob_battries_capacity_in_MH(self,new_mob_battries_capacity_in_MH)->None:
        """
            Sets the mob_battries_capacity_in_MH attribute of the calling object to
@new_mob_battries_capacity_in_MH
            Before setting, TypeCheck and ValueCheck is performed.
        """
        if type(new_mob_battries_capacity_in_MH)!=int:
            raise TypeError("new_mob_battries_capacity_in_MH must be in int")
        if new_mob_battries_capacity_in_MH<=0:
            raise ValueError("new_mob_battries_capacity_in_MH must be positive")
        self.mob_battries_capacity_in_MH=new_mob_battries_capacity_in_MH

    def set_mob_rear_camera_reso_in_mp(self,new_mob_rear_camera_reso_in_mp)->None:
        """
            Sets the mob_rear_camera_reso_in_mp attribute of the calling object to
@new_mob_rear_camera_reso_in_mp
            Before setting, TypeCheck and ValueCheck is performed.
        """
        if type(new_mob_rear_camera_reso_in_mp)!=int:
            raise TypeError("new_mob_rear_camera_reso_in_mp must be int")
        if new_mob_rear_camera_reso_in_mp<=0:
            raise ValueError("new_mob_rear_camera_reso_in_mp must be positive")
        self.mob_rear_camera_reso_in_mp=new_mob_rear_camera_reso_in_mp
    def set_mob_model_year(self,new_mob_model_year)->None:
        """
            Sets the mob_model_year attribute of the calling object to @new_mob_model_year
            Before setting, TypeCheck and ValueCheck is performed.
        """
        if type(new_mob_model_year)!=int:
            raise TypeError("new_mob_model_year must be int")
        if new_mob_model_year<=0:
            raise ValueError("new_mob_model_year must be positive")
        self.mob_model_year=new_mob_model_year

    def set_mob_CPU_model(self,new_mob_CPU_model)->None:
        """
            Sets the mob_CPU_model attribute of the calling object to @new_mob_CPU_model
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_CPU_model)!=str:
            raise TypeError("new_mob_CPU_model must be in str")
        self.mob_CPU_model=new_mob_CPU_model

    def set_mob_included_components(self,new_mob_included_components)->None:
        """
            Sets the mob_included_components attribute of the calling object to
@new_mob_included_components
            Before setting, TypeCheck is performed.
        """
        if '__iter__' not in dir(type(new_mob_included_components)):
            raise TypeError("new_mob_included_components must be iterable")
        for componant in new_mob_included_components:
            if type(componant)!=str:
```

```
            raise TypeError("new_mob_included_components must be str")
        self.mob_included_components=new_mob_included_components

    def set_mob_display_type(self,new_mob_display_type)->None:
        """
            Sets the mob_display_type attribute of the calling object to @new_mob_display_type
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_display_type)!=str:
            raise TypeError("new_mob_display_type must be in str")
        self.mob_display_type=new_mob_display_type

    def set_mob_human_interface_input(self,new_mob_human_interface_input)->None:
        """
            Sets the mob_human_interface_input attribute of the calling object to
@new_mob_human_interface_input
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_human_interface_input)!=str:
            raise TypeError("new_mob_human_interface_input must be in str")
        self.mob_human_interface_input=new_mob_human_interface_input

    def set_mob_batteries_desc(self,new_mob_batteries_desc)->None:
        """
            Sets the mob_batteries_desc attribute of the calling object to @new_mob_batteries_desc
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_batteries_desc)!=str:
            raise TypeError("new_mob_batteries_desc must be in str")
        self.mob_batteries_desc=new_mob_batteries_desc

    def set_mob_sim_card_size(self,new_mob_sim_card_size)->None:
        """
            Sets the mob_sim_card_size attribute of the calling object to @new_mob_sim_card_size
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_sim_card_size)!=str:
            raise TypeError("new_mob_sim_card_size must be in str")
        self.mob_sim_card_size=new_mob_sim_card_size

    def set_mob_material_feature(self,new_mob_material_feature)->None:
        """
            Sets the mob_material_feature attribute of the calling object to @new_mob_material_feature
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_material_feature)!=str:
            raise TypeError("new_mob_material_feature must be in str")
        self.mob_material_feature=new_mob_material_feature

    def set_mob_shooting_modes(self,new_mob_shooting_modes)->None:
        """
            Sets the mob_shooting_modes attribute of the calling object to @new_mob_shooting_modes
            Before setting, TypeCheck is performed.
        """
        if '__iter__' not in dir(type(new_mob_shooting_modes)):
            TypeError("mob_shooting_modes must be iterable")
        for mode in new_mob_shooting_modes:
            if type(mode)!=str:
                raise TypeError("mode must be in str")
        self.mob_shooting_modes=new_mob_shooting_modes
    def set_mob_GPS(self,new_mob_GPS)->None:
        """
            Sets the mob_GPS attribute of the calling object to @new_mob_GPS
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_GPS)!=str:
            raise TypeError("new_mob_GPS must be in str")
        self.mob_GPS=new_mob_GPS

    def set_mob_water_resistence_level(self,new_mob_water_resistence_level)->None:
        """
```

```
            Sets the mob_water_resistence_level attribute of the calling object to
@new_mob_water_resistence_level
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_water_resistence_level)!=str:
            raise TypeError("new_mob_water_resistence_level must be in str")
        self.mob_water_resistence_level=new_mob_water_resistence_level

    def set_mob_optical_sensor_reso_in_mp(self,new_mob_optical_sensor_reso_in_mp)->None:
        """
            Sets the mob_optical_sensor_reso_in_mp attribute of the calling object to
@new_mob_optical_sensor_reso_in_mp
            Before setting, TypeCheck and ValueCheck is performed.
        """
        if type(new_mob_optical_sensor_reso_in_mp)!=int:
            raise TypeError("new_mob_optical_sensor_reso_in_mp must be in int")
        if new_mob_optical_sensor_reso_in_mp<=0:
            raise ValueError("new_mob_optical_sensor_reso_in_mp must be in positive")
        self.mob_optical_sensor_reso_in_mp=new_mob_optical_sensor_reso_in_mp

    def set_mob_max_display_reso(self,new_mob_max_display_reso)->None:
        """
            Sets the mob_max_display_reso attribute of the calling object to
@new_mob_max_display_reso
            Before setting, TypeCheck is performed.
        """
        if type(new_mob_max_display_reso)!=str:
            raise TypeError("new_mob_max_display_reso must be in str")
        self.mob_max_display_reso=new_mob_max_display_reso

    def set_mob_video_capture_reso_in_pixel(self,new_mob_video_capture_reso_in_pixel)->None:
        """
            Sets the mob_video_capture_reso_in_pixel attribute of the calling object to
@new_mob_video_capture_reso_in_pixel
            Before setting, TypeCheck and ValueCheck is performed.
        """
        if type(new_mob_video_capture_reso_in_pixel)!=int:
            raise TypeError("new_mob_video_capture_reso_in_pixel must be in int")
        if new_mob_video_capture_reso_in_pixel<=0:
            raise ValueError("new_mob_video_capture_reso_in_pixel must be positive")
        self.mob_video_capture_reso_in_pixel=new_mob_video_capture_reso_in_pixel
    def show_details(self)->None:
        """
            This function display all the characterstics of Mobile class
        """
        print("GPU:{}".format(self.mob_GPU))
        print("RAM:{}".format(self.mob_RAM_in_gb))
        print("Product Dimensions:{}".format(self.mob_prod_dimensions.__dict__))
        print("Batteries:{}".format(self.mob_battries))
        print("Wireless Communication technologies:{}".format(self.mob_wireless_commu_tech))
        print("Spacial Features:{}".format(self.mob_special_feature))
        print("Display technology:{}".format(self.mob_display_tech))
        print("Manufacturer:{}".format(self.mob_manufacturer))
        print("Country of origin:{}".format(self.mob_country_of_origin))
        print("Colour:{}".format(self.mob_colour))
        print("Screen Size:{}".format(self.mob_screen_size_in_inches))
        print("Connector Type:{}".format(self.mob_connector_type))
        print("From Photo Sensor Resolution:{}".format(self.mob_front_photo_sensor_reso_in_mp))
        print("From Factor:{}".format(self.mob_from_factor))
        print("Battery Capacity:{}".format(self.mob_battries_capacity_in_MH))
        print("Rear Camera Reso:{}".format(self.mob_rear_camera_reso_in_mp))
        print("Model Year:{}".format(self.mob_model_year))
        print("CPU Model:{}".format(self.mob_CPU_model))
        print("Included Componentes:{}".format(self.mob_included_components))
        print("Display Type:{}".format(self.mob_display_type))
        print("Human Interface Input:{}".format(self.mob_human_interface_input))
        print("Battery Description:{}".format(self.mob_batteries_desc))
        print("SIM card size:{}".format(self.mob_sim_card_size))
        print("Material Feature:{}".format(self.mob_material_feature))
        print("Shooting Modes:{}".format(self.mob_shooting_modes))
        print("GPS:{}".format(self.mob_GPS))
        print("Water Resistance Level:{}".format(self.mob_water_resistence_level))
        print("Optical Sensor Resolution:{}".format(self.mob_optical_sensor_reso_in_mp))
        print("Display Resolution Maximum:{}".format(self.mob_max_display_reso))
        print("Video Capture Resolution:{}".format(self.mob_video_capture_reso_in_pixel))

def main():
    mob_obj=Mobile(
            "Qualcomm",
            8,
            ProductDimension(7.6,0.8,16.6,195.0),
            "1 Lithium Polymer batteries required(included)",
            "Cellular",
            ["Rear Camera,Camera"],
            "AMOLED",
            "Wireless",
            "Oppo Mobile India Private Limited",
            "India",
            "Pastel Lime",
            6.72,
            "USB Type C",
            2,
            "SmartPhone",
            5000,
            2,
            2023,
            "Snapdragon",
            ["SIM Tray Ejector","Adapter","Phone Case","USB Cable"],
            "LCD",
            "Keyboard",
            "Lithium-Ion",
            "Nano",
            "plastic",
            ["Macro","Portrait"],
            "GLONASS",
            "Water Resistant",
                2,
                "1080*2400 Pixels",
                1080
    )
    print("MOBILE PRODUCT DETAILS:")
    mob_obj.show_details()
    #we can also get the attribute using getter method and
    #set the specific attribute using setter method
    sys.exit(0)
main()
```