Assignment: Vending Machines and Method Inheritance

Welcome to this Mini Project.

Awesome. Now you work for a software development company that develops software for vending machines in a school district and in a local hospital.

Your task is to:

- Customize the behavior of the **VendingMachine** class with method inheritance and method overriding.
- To do this, you will define two subclasses: **HospitalVendingMachine** and **SchoolVendingMachine**. They will inherit from **VendingMachine** (this class is already defined in the Python file that you will download for this project).

Requirements for both types of vending machines:

- The subclasses must have a custom greeting message for the user. This message should be displayed before the list of available products.
  - The code must override the **sales_menu** method in these two subclasses.
  - They should print their custom message (shown below) before calling the **sales_menu** method of the **VendingMachine** superclass.
- Custom messages:
  - For **SchoolVendingMachine**:

    Welcome to our School Vending Machine
    We hope you have a great day full of learning!

  - For **HospitalVendingMachine**:

    Welcome to our Hospital Vending Machine
    We hope you are feeling better today!

- The subclass should have a class attribute **snack_prices**. This attribute should replace the value defined in the superclass. You can customize the prices (values) to your liking but please keep the structure of the original dictionary.
- Override the **find_snack_price** method in both subclasses to make them use the class attribute that corresponds to the subclass.

Requirements for SchoolVendingMachine:

- Define a class attribute called **student_debt** (a dictionary with students' names as the keys and their corresponding debt as values).
- Define a method **print_student_debt** that takes the name of the student as argument and prints the value of the debt.

Requirements for HospitalVendingMachine:

- Define a method called **print_days_until_maintenance** that prints a string with the number of days remaining until the next scheduled maintenance (this is an instance attribute of the superclass).

Tips:

- I would suggest taking a few minutes to read the code and familiarize yourself with the code. This is intended to help you practice working with existing code and reading code written by other developers.
- The code in the downloadable Python file includes descriptive comments that you may find helpful.

Code:

This is the code in the downloadable Python file for this assignment:

```
class VendingMachine:

    total_revenue = 0 # Total revenue of all vending machines in the system

    snack_prices = {"candy": 2.00, "soda": 1.50, "chips": 3.00, "cookies": 3.50}

    def __init__(self, inventory, serial, days_until_maintenance):
        self.inventory = inventory
        self.revenue = 0
        self.serial = serial
        self.days_until_maintenance = days_until_maintenance


    def sales_menu(self):

        while True:

            greetings = "\nWelcome! I have:\n"
            request = "\nPlease enter the number of the item: "

            print(greetings)

            i = 1
            for snack in self.inventory:
                print("(" + str(i) + ") " + snack.capitalize())
                i += 1

            cust_input = int(input(request))

            while cust_input <= 0 or cust_input > len(self.inventory):
                print("Please enter a number from 1 to", len(self.inventory))
                cust_input = int(input(request))

            self.process_sale(list(self.inventory.keys())[cust_input -
1].lower())
            answer = int(input("\nWould you like to buy another snack?\nEnter 1
for YES and 0 for NO: "))

            if not answer:
                break


    def process_sale(self, option): # option must be in lowercase

        print("\nYou selected: %s" % option.capitalize())

        if self.inventory[option] > 0:

            print("Great! I currently have %d %s in my inventory\n" %
(self.inventory[option], option))
```

```
            num_items = int(input("How many %s would you like to buy?\n" %
option))

            while num_items <= 0:
                print("Please enter a positive integer")
                num_items = int(input("\nHow many %s would you like to buy?\n" %
option))

            if num_items <= self.inventory[option]:
                self.remove_from_inventory(option, num_items)

                total = self.update_revenue(option, num_items)

                print("That would be: $ " + str(total))

                print("\nThank you for your purchase!")
                print("Now I have %d %s and my revenue is $%d" %
(self.inventory[option], option, self.revenue))

            else:
                print("I don't have so many %s. Sorry! :(" % option)

        else:
            print("I don't have any more %s. Sorry! :(" % option)


    def remove_from_inventory(self, option, num_items):
        self.inventory[option] -= num_items

    def update_revenue(self, option, num_items):
        # Find price of the snack
        price = self.find_snack_price(option)

        # Update Instance and class
        self.revenue += num_items * price
        VendingMachine.total_revenue += num_items * price

        return num_items * price

    def find_snack_price(self, snack):
        return VendingMachine.snack_prices[snack]

    def display_revenue(self):
        print("The total revenue of this vending machine is:", self.revenue)


class HospitalVendingMachine(VendingMachine):

    # Complete the class


class SchoolVendingMachine(VendingMachine):

    # Complete the class
```

```
floor_machine = VendingMachine({"candy": 36, "soda": 15, "chips": 40,
"cookies": 120}, "011423424", 24)
floor_machine.sales_menu()

hospital_machine = HospitalVendingMachine({"candy": 32, "soda": 50, "chips":
45, "cookies": 80}, "03223424", 15)
# hospital_machine.sales_menu()

school_machine = SchoolVendingMachine({"candy": 36, "soda": 15, "chips": 40,
"cookies": 120}, "0534424", 2)
# school_machine.sales_menu()
```

Note:

The instances are already defined to help you test your code. You can comment and uncomment the lines **<instance>.salesMenu()** to run different versions of this method.

Solution:

You can find a sample solution in the "Instructor example" tab.