Assignment on POJO class

You are required to create a Java POJO class for a Product management system. The class should represent the attributes and behavior of a Product object.

Requirements

Here are the requirements for the Product POJO class:

- The class should have private fields to represent the following attributes of a Product: name, description, SKU (Stock Keeping Unit), price, and quantity.
- The class should have public getters and setters for each of the fields.
- The class should have a constructor that takes in values for all of the fields and initializes them.
- The class should override the **toString** method to provide a human-readable string representation of the Product object.

Bonus

Here are some bonus requirements that you can implement if you want to:

- Add a method to calculate the total cost of the products based on their price and quantity.
- Add a method to compare two products based on their price.
- Add a method to validate the SKU format.

Conclusion

The Product POJO class provides a simple solution for representing and managing product data. By encapsulating product attributes into private fields and providing public getters and setters, the class ensures that data is accessed and modified safely and consistently.

The constructor provides an easy way to initialize a Product object with all of its attributes, and the **toString** method allows the object to be easily printed and understood by humans.

The bonus requirements extend the functionality of the Product class and provide additional features, such as calculating the total cost of products or comparing two products based on their price.

Overall, the Product POJO class is a flexible, efficient, and scalable solution for managing product data and can be used as a building block for more complex product management systems.

Below is more detailed explanation of this problem

You are required to create a Java POJO (Plain Old Java Object) class for a simple Product management system. The class should represent the attributes and behavior of a Product object.

Requirements

Here are the requirements for the Product POJO class:

Fields

The **Product** class should contain private fields to represent the following attributes of a Product:

- **name** - the name of the product
- **description** - a brief description of the product
- **SKU** - the SKU (Stock Keeping Unit) code of the product
- **price** - the price of the product
- **quantity** - the quantity of the product in stock

Getters and setters
The Product class should have public getters and setters for each of the fields to allow safe access to and modification of the data.
Constructor
The Product class should have a constructor that takes in values for all of the fields and initializes them.
toString method
The Product class should override the **toString** method to provide a human-readable string representation of the Product object.
Bonus
Here are some bonus requirements that you can implement if you want to:

- Add a method to calculate the total cost of the products based on their price and quantity.
- Add a method to compare two products based on their price.
- Add a method to validate the SKU format.

Here's a sample code template for the **Product** POJO class without business logic:

```
public class Product {
    private String name;
    private String description;
    private String SKU;
    private double price;
    private int quantity;

    public Product(String name, String description, String SKU, double price, int quantity) {
        // TODO: Implement constructor
    }

    public String getName() {
        // TODO: Implement getter
        return null;
    }

    public void setName(String name) {
        // TODO: Implement setter
    }

    public String getDescription() {
        // TODO: Implement getter
        return null;
    }

    public void setDescription(String description) {
        // TODO: Implement setter
    }
```

```
    public String getSKU() {
        // TODO: Implement getter
        return null;
    }

    public void setSKU(String SKU) {
        // TODO: Implement setter
    }

    public double getPrice() {
        // TODO: Implement getter
        return 0.0;
    }

    public void setPrice(double price) {
        // TODO: Implement setter
    }

    public int getQuantity() {
        // TODO: Implement getter
        return 0;
    }

    public void setQuantity(int quantity) {
        // TODO: Implement setter
    }

    @Override
    public String toString() {
        // TODO: Implement toString method
        return null;
    }
}
```

Here's an example **Main** method that creates two **Product** objects and displays their details:

```
public static void main(String[] args) {
    // Create two Product objects
    Product product1 = new Product("Apple", "Fresh red apples", "1234", 0.99, 100);
    Product product2 = new Product("Banana", "Yellow bananas", "5678", 0.49, 200);

    // Display the details of the two products
    System.out.println(product1);
    System.out.println(product2);
}
```

And here's the expected output:

Name: Apple
Description: Fresh red apples
SKU: 1234
Price: $0.99
Quantity: 100

Name: Banana
Description: Yellow bananas
SKU: 5678
Price: $0.49
Quantity: 200