**Mini E-Commerce System using Python OOP**

**Context:**

You are hired as a junior software developer to work on a mini e-commerce system for a startup. Your task is to design a simplified system using Python's Object-Oriented Programming (OOP) principles. This project will involve building a basic structure of an e-commerce application that handles products, applies discounts based on product categories, and demonstrates secure access to product data.

**Objectives:**

1. Understand and implement **inheritance** by creating a hierarchy of product categories.
2. Demonstrate **polymorphism** by applying different discount rates for different product types.
3. Apply **encapsulation** to protect and manage product data securely.

**Requirements:**

You need to develop a Python program with the following features:

**1. Product Class (Base Class)**

Create a base class named Product with the following characteristics:

* **Attributes (Private):**
  + product\_id (integer): A unique identifier for each product.
  + name (string): The name of the product.
  + price (float): The original price of the product.
* **Methods:**
  + Constructor (\_\_init\_\_): Initialize the product attributes with provided values.
  + Getter Methods: get\_product\_id(), get\_name(), get\_price() to retrieve the values of the attributes.
  + Setter Method: set\_price(price) to update the price of the product.
  + apply\_discount(): A method that returns the price without any discount (default behavior).

**2. Inheritance - Subclasses for Product Categories**

You need to extend the Product class to create specialized product categories:

* **Electronics (Subclass)**:
  + Additional Attribute:
    - warranty\_period (integer): The warranty period in years.
  + Override the apply\_discount() method to provide a 10% discount for electronics products.
* **Clothing (Subclass)**:
  + Additional Attribute:
    - material (string): The material of the clothing (e.g., "Cotton", "Wool").
  + Override the apply\_discount() method to provide a 20% discount for clothing products.

**3. Polymorphism - Applying Discounts**

Implement **polymorphism** by overriding the apply\_discount() method in the Electronics and Clothing subclasses:

* In the base Product class, apply\_discount() returns the original price.
* In Electronics, apply\_discount() should return the price after applying a 10% discount.
* In Clothing, apply\_discount() should return the price after applying a 20% discount.

### ****4. Encapsulation - Secure Access to Product Data****

Ensure that the product details (product\_id, name, price) are private and cannot be accessed directly. Use getter and setter methods to access or modify the data.

**5. Additional Features (Optional)**

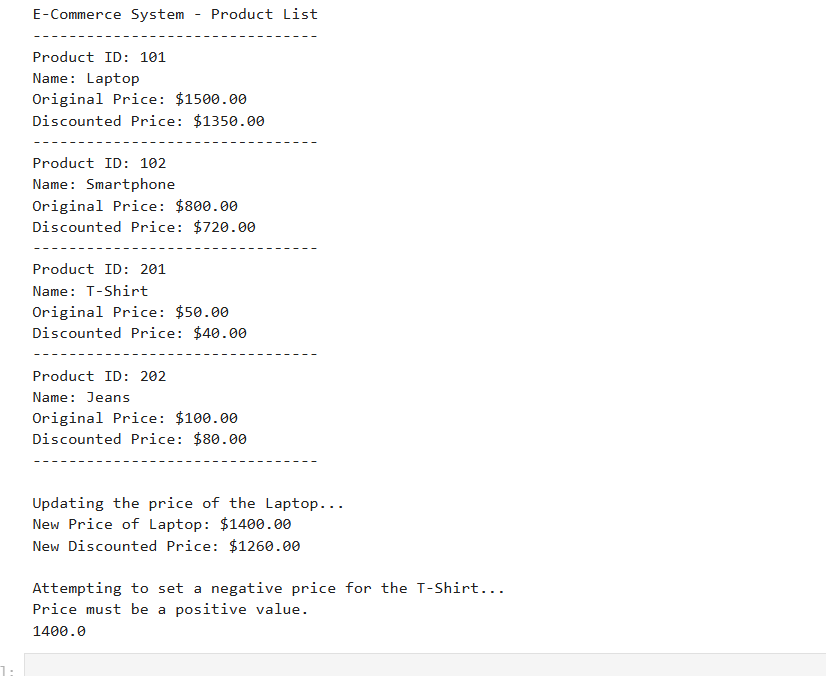
Implement the following features:

* Add a method to update the price of a product using the setter method.
* Implement an \_\_str\_\_() method for the Product class to provide a formatted string representation of the product details.

**6. Test the System**

* Create a few products of each type (Electronics and Clothing).
* Print the details of each product, including the discounted price.
* Demonstrate the usage of inheritance, polymorphism, and encapsulation.

**Expected Output:**

****