Build a program that simulates an online store's inventory system. This exercise will practice inheritance, abstract and virtual methods, and method overriding.

System Overview:

The system will manage different types of products in an online store, such as electronics, clothing, and groceries, with specific attributes and behaviors for each category.

Class Hierarchy:

* **Product (Abstract Class)**:
  + Properties: ID (string), Name (string), Price (decimal), Description (string)
  + Abstract Method: **CalculateShippingCost()** (returns a **decimal**, cost based on product type and weight)
  + Virtual Method: **PrintProductDetails()** (prints basic details about the product)
* **Electronics (Derived from Product)**:
  + Additional Properties: WarrantyPeriod (int, in months), Weight (decimal, in kg)
  + Overrides **CalculateShippingCost()**
  + Overrides **PrintProductDetails()**
* **Clothing (Derived from Product)**:
  + Additional Properties: Size (string), Material (string), Weight (decimal, in kg)
  + Overrides **CalculateShippingCost()**
  + Overrides **PrintProductDetails()**
* **Groceries (Derived from Product)**:
  + Additional Properties: ExpiryDate (DateTime), Weight (decimal, in kg)
  + Overrides **CalculateShippingCost()**
  + Overrides **PrintProductDetails()**
* **Smartphone (Derived from Electronics)**:
  + Additional Properties: ScreenSize (decimal, in inches), BatteryLife (int, in hours)
  + Overrides **PrintProductDetails()**
* **TShirt (Derived from Clothing)**:
  + Additional Property: Color (string)
  + Overrides **PrintProductDetails()**

Exercise Tasks:

1. **Implement the Abstract and Base Classes**:
   * Define the **Product** class with abstract and virtual methods as described.
   * Implement derived classes **Electronics**, **Clothing**, and **Groceries**, providing implementations for abstract methods.
2. **Add Further Specialization**:
   * Implement the **Smartphone** class derived from **Electronics** and the **TShirt** class derived from **Clothing**. These should add more specific properties and override the **PrintProductDetails()** method to include all relevant information.
3. **Shipping Cost Calculation**:
   * Implement logic in **CalculateShippingCost()** for each derived class, considering factors like weight and product type. For example, electronics might have a higher base shipping cost due to insurance, while groceries might have different rates based on perishability.
4. **Detailed Product Display**:
   * Override the **PrintProductDetails()** method in each class to display all properties of the product, including those inherited from parent classes.
5. **Main Program**:
   * In the Main method, create instances of each product type, set their properties, and demonstrate the calculation of shipping costs and printing of product details.

Additional Challenge (Optional):

* Add a feature to calculate discounts for products based on categories or specific attributes (e.g., clearance items in clothing).
* Implement input validation for all settable properties across the class hierarchy to ensure data integrity (e.g., positive values for weight, future dates for expiry dates).