By: Eng. Elias Al-Showaiter

[Dart Programming Language] - Lecture [11]

[Exercises part 2]

1. Question:

You are tasked with creating a **Product Management System** for an e-commerce platform. You will design a **Product** data model with various attributes and methods to simulate real-world challenges in managing inventory, sales, and discounts. Follow the steps below to complete the task:

Step 1: Define the Product class

Design a `Product` class that contains the following attributes:

- `id` (unique identifier, integer)
- `name` (product name, string)
- `description` (detailed description of the product, string)
- `price` (price of the product, double)
- `stock` (number of items in stock, integer)
- `category` (the category the product belongs to, string)
- `rating` (average customer rating from 1 to 5, double)
- `reviews` (a list of strings representing customer reviews)
- `discount` (discount percentage, double)
- `soldUnits` (number of units sold, integer)

You will also implement the following methods:

1. `applyDiscount()` – Apply the product's discount to its price and return the discounted price.
2. `sellProduct(int quantity)` – Reduce the stock by the given quantity, increase the `soldUnits` by the same amount, and ensure that there is enough stock before completing the sale.
3. `restock(int quantity)` – Add the specified quantity to the product's stock.
4. `addReview(String review, double rating)` – Add a customer review and update the average rating of the product.
5. `isOutOfStock()` – Return `true` if the stock is zero, otherwise return `false`.

Step 2: Create multiple product instances

After defining the class, create a list of at least 5 different `Product` instances with varying attributes, such as different categories, prices, stock levels, and discounts.

Step 3: Simulate operations on the product list

Perform the following operations on the list of products:

1. **Filtering**: Filter all products in the category "Electronics" and display their names and prices.
2. **Sorting**: Sort the products by price in ascending order and display the sorted list.
3. **Transformation**: Create a list of discounted prices for all products using their `applyDiscount()` method.
4. **Restocking**: Restock 50 units of a product with low stock (i.e., stock less than 10).
5. **Selling**: Sell 5 units of the most expensive product and print the updated stock and sold units.
6. **Reviews**: Add a review to the product with the highest rating and display its updated rating and reviews.

Step 4: Calculate Total Revenue

After simulating the sales, write a method that calculates the total revenue generated by the platform by summing up the `soldUnits` multiplied by the price of each product (before any discount).

Step 5: Generate a Report

Generate a report that displays:

1. The product with the highest rating.
2. The product with the lowest stock.
3. The total revenue generated by all products sold.
4. The average price of all products in the "Furniture" category.

Bonus (Optional): Handling Discontinued Products

1. Implement a way to mark products as "discontinued". Discontinued products should not be restocked or sold.
2. Modify your `sellProduct()` method to raise an error if the product is discontinued.

First we create the class:

```dart
class Product {
  int id;
  String name;
  String description;
  double price;
  int stock;
  String category;
  double rating;
  List<String> reviews;
  double discount;
  int soldUnits;
  bool discontinued;

  Product({
    required this.id,
    required this.name,
    required this.description,
    required this.price,
    required this.stock,
    required this.category,
    this.rating = 0.0,
    this.reviews = const [],
    this.discount = 0.0,
    this.soldUnits = 0,
    this.discontinued = false,
  });

  double applyDiscount() {
    return price * (1 - discount / 100);
  }

  void sellProduct(int quantity) {
    if (discontinued) {
      throw Exception('Product is discontinued and cannot be sold.');
    }
    if (quantity > stock) {
      throw Exception('Not enough stock to complete the sale.');
    }
    stock -= quantity;
    soldUnits += quantity;
  }
```

```
 void restock(int quantity) {
    if (discontinued) {
      throw Exception('Cannot restock a discontinued product.');
    }
    stock += quantity;
  }

  void addReview(String review, double newRating) {
    reviews.add(review);
    rating = ((rating * (reviews.length - 1)) + newRating) /
reviews.length;
  }

  bool isOutOfStock() {
    return stock == 0;
  }

  @override
  String toString() {
    return 'Product($name, Price: \$${price.toStringAsFixed(2)}, Stock:
$stock, Rating: $rating)';
  }
}
```

Now we write the main function.

```
void main() {
  List<Product> products = [
    Product(
        id: 1,
        name: 'Laptop',
        description: 'High-performance laptop',
        price: 999.99,
        stock: 5,
        category: 'Electronics',
        discount: 10.0,
        reviews: ["Good"]),
    Product(
        id: 2,
        name: 'Smartphone',
        description: 'Latest model smartphone',
        price: 699.99,
        stock: 8,
        category: 'Electronics',
        rating: 4.5,
        reviews: ["Good"]),
    Product(
        id: 3,
        name: 'Sofa',
        description: 'Comfortable leather sofa',
        price: 299.99,
        stock: 2,
        category: 'Furniture',
        discount: 15.0,
        reviews: ["Good"]),
    Product(
        id: 4,
        name: 'Office Chair',
        description: 'Comfortable office chair',
        price: 149.99,
        stock: 12,
        category: 'Furniture',
        reviews: ["Good"]),
    Product(
        id: 5,
        name: 'Headphones',
        description: 'High Quality  headphones',
        price: 199.99,
        stock: 20,
        category: 'Electronics',
        rating: 4.8,
        reviews: ["Good"]),
];

// Step 3: Simulate operations
// 1. Filtering: Electronics
List<Product> electronics =
    products.where((p) => p.category == 'Electronics').toList();
print('Filtered Electronics Products:');
electronics.forEach((p) => print('${p.name} - \$${p.category}'));

// 2. Sorting: Sort by price
products.sort((a, b) => a.price.compareTo(b.price));
print('\nProducts Sorted by Price:');
products.forEach((p) => print('${p.name} - \$${p.price}'));

// 3. Transformation: Apply discount
List<double> discountedPrices =
      products.map((p) => p.applyDiscount()).toList();
  print('\nDiscounted Prices:');
  discountedPrices.forEach((p) => print('\$${p.toStringAsFixed(2)}'));

  // 4. Restocking
  Product lowStockProduct = products.firstWhere((p) => p.stock < 10);
  lowStockProduct.restock(50);
  print(
      '\nRestocked ${lowStockProduct.name}. New stock:
${lowStockProduct.stock}');

  // 5. Selling: Most expensive product
  Product mostExpensiveProduct = products.last;
  mostExpensiveProduct.sellProduct(5);
  print(
      '\nSold 5 units of ${mostExpensiveProduct.name}. Remaining stock:
${mostExpensiveProduct.stock}, Sold units:
${mostExpensiveProduct.soldUnits}');

  // 6. Reviews: Add a review to the highest-rated product
  Product highestRatedProduct = products.reduce((a, b) {
    if (a.rating > b.rating) {
      return a;
    } else {
      return b;
    }
  });
  highestRatedProduct.addReview('Amazing product!', 5.0);
  print(
      '\nAdded review to ${highestRatedProduct.name}. New rating:
${highestRatedProduct.rating}, Reviews: ${highestRatedProduct.reviews}');

  // Step 4: Calculate total revenue
  double totalRevenue =
      products.fold(0, (sum, p) => sum + p.soldUnits * p.price);
  print('\nTotal Revenue: \$${totalRevenue.toStringAsFixed(2)}');

  // Step 5: Generate a report
  Product lowestStockProduct =
      products.reduce((a, b) => a.stock < b.stock ? a : b);
  List<Product> furnitureProducts =
      products.where((p) => p.category == 'Furniture').toList();
  double averageFurniturePrice =
      furnitureProducts.fold(0.0, (sum, p) => sum + p.price) /
          furnitureProducts.length;
  print('\n--- Report ---');
  print(
      'Product with highest rating: ${highestRatedProduct.name}
(${highestRatedProduct.rating})');
  print(
      'Product with lowest stock: ${lowestStockProduct.name} (Stock:
${lowestStockProduct.stock})');
  print('Total revenue: \$${totalRevenue.toStringAsFixed(2)}');
  print(
      'Average price of furniture products:
\$${averageFurniturePrice.toStringAsFixed(2)}');
}
```