In our project, we introduced the **Priced** interface to separate pricing behaviour from the base product logic. This design makes pricing consistent across all products and decorators, allowing any class that implements **Priced** to define its own pricing logic while maintaining a common interface. This helps ensure flexibility and clarity in how price calculations are performed.

One example of preserving the Open/Closed Principle (OCP) is in the use of the **ProductFactory**. The factory can create new combinations of products and add-ons without modifying the existing factory logic or product classes. Each decorator (e.g., **ExtraShot, OatMilk, Syrup, SizeLarge**) extends functionality by wrapping around an existing Product, rather than altering any of the original code. The system is therefore open for extension but closed for modification.

If I wanted to add a new add-on next week, such as **WhippedCream**, I would simply:

- Create a new class **WhippedCream** that extends **ProductDecorator** and implements Priced.
- Define how it modifies the product's name() and price().
- Add a new token (e.g., "**WC**") in the **ProductFactory** switch statement to instantiate it.

This approach preserves the Open/Closed Principle because the new functionality is added through extension, not by changing the internal logic of existing classes.

If I experimented with new tokens in the factory, the system remained compliant with OCP because these additions simply extended the set of recognized products, without modifying any existing product or decorator behaviour.