Design Pattern

Code Smell:

1. In userconstraints.java file I found a code smell called large class. In this class, it contains too much information and takes too much responsibility. So it looks too much messy. So I refactored this class by using the extract class.
2. In cartservice.java file in addproductToCart and updateProductToCart I found a code smell called Long parameter list. In long parameter list it takes too much parameter and that’s why it creates a code smell. So I refactored by using the introduce parameter object.
3. In loginSrv.java file I found codesmell called dispensable code smell specially comments. In dispensable code smell it doesn’t create any real value rather it makes code block too much messy. So I refactored it using the Extract variable method.

Principles/pattern:

1. In UpdateProductSrv.java file I used state pattern for better implementations,Also here add new functionality is very tough. State pattern is such kind of patterns that’s change it’s behavior when it’s internal state changes. I used here state pattern to implements LoginAsAdmin and errorlogin. I create an interface class UpdateProductSrv.And then implements those two class. Now it is clean and also if someone wants to add any functionality then it is easily possible .

Implements a new feature:

1. I want a centralized observation systems in this above system. Where the admin can see which product is best selling, in which reagion which product is more suitable, warehousee informations etc etc through as a mediator. For this feature I want to use mediator design patterns in my new mediator.java file.