# Name: Memento Pattern

## Intent:

This applies at the user account information page.

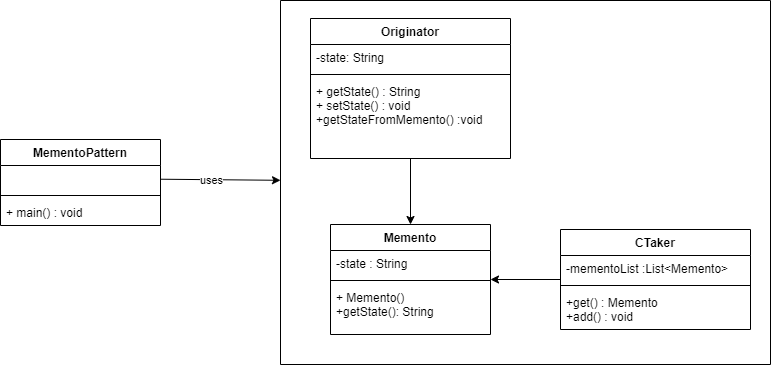
## Problem:

If the user change the info in the main database all the user accounts info will be changed.

## Solution:

Make a class to Memento contains the state of the object being restored. Originators create and store state in Memento objects, and Caretaker objects are responsible for restoring the state of objects in Memento. We have created Memento, Originator and Ctaker classes.

## Structure:



## Participants:

MementoPattern, Originator, Memento, CTaker and User.

## Consequences:

The effect on the system that the info came back to its defaulted that will increase the performance with lowering the cost of database.

# Name: Prototype Pattern

## Intent:

When new bid about to be created.

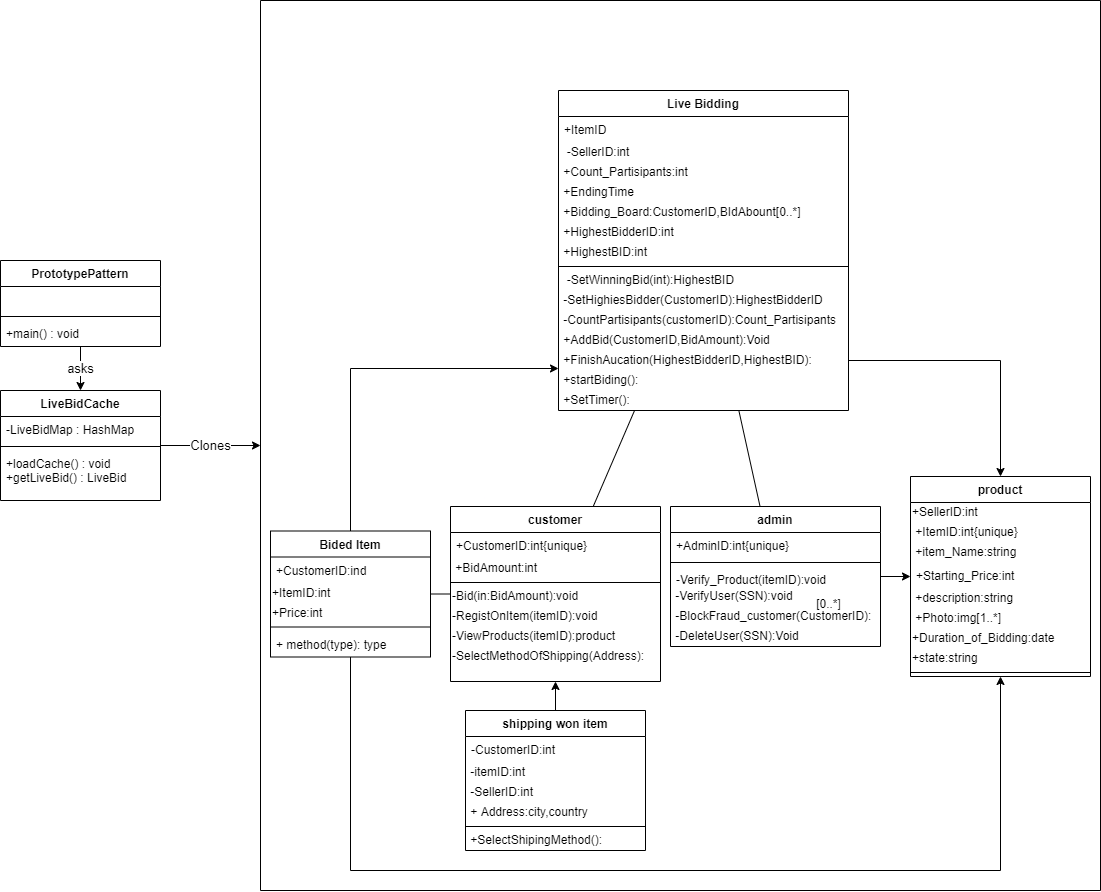
## Problem:

Every time we create new bid will create duplicate object while keeping performance on mind.

## Solution:

This pattern involves the implementation of a prototype interface that calls the creation of a clone of the current object. You can reduce the number of database calls by caching objects, returning a clone on the next request, and updating the database as needed.

## Structure:



## Participants:

Live Bidding, Bided Item, customer, admin and product.

## Consequences:

Every time the seller starts new bid it will take a clone from the cached one rather than create a new one from the scratch.

# Name: Read-only repositories Pattern

## Intent:

When Admin or user check the notification.

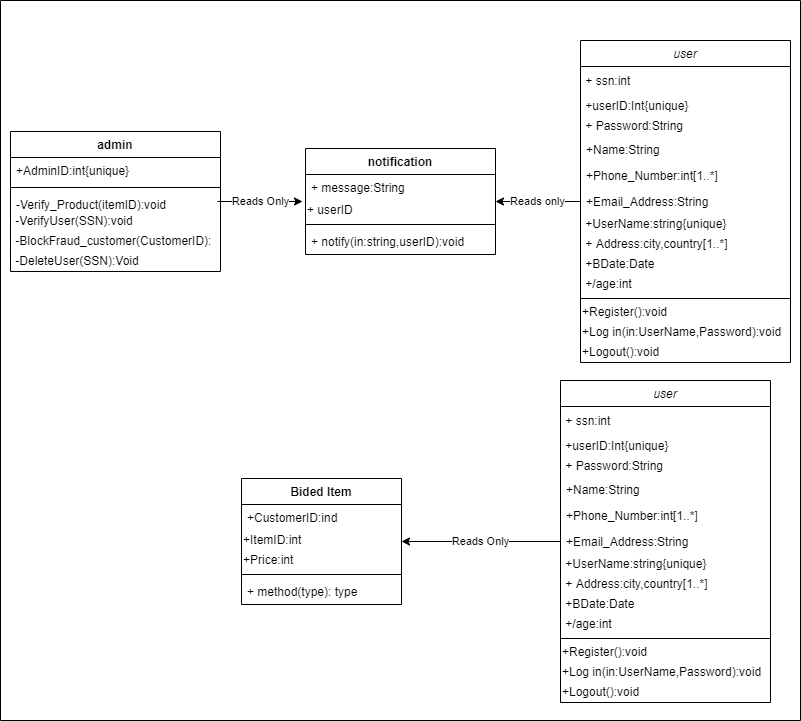
## Problem:

Class will be create every time each user access it.

## Solution:

A repository pattern can be used to encapsulate data storage specific code in designated components. The part of application that needs the data will only work with the repositories. Will create a repository for each combination of item you store and your database technology.

## Structure:



## Participants:

Admin, user, notification and bided item.

## Consequences:

Will create a repository for each combination of item you store and your database technology.

# Name: Factory Pattern

## Intent:

While registering new user.

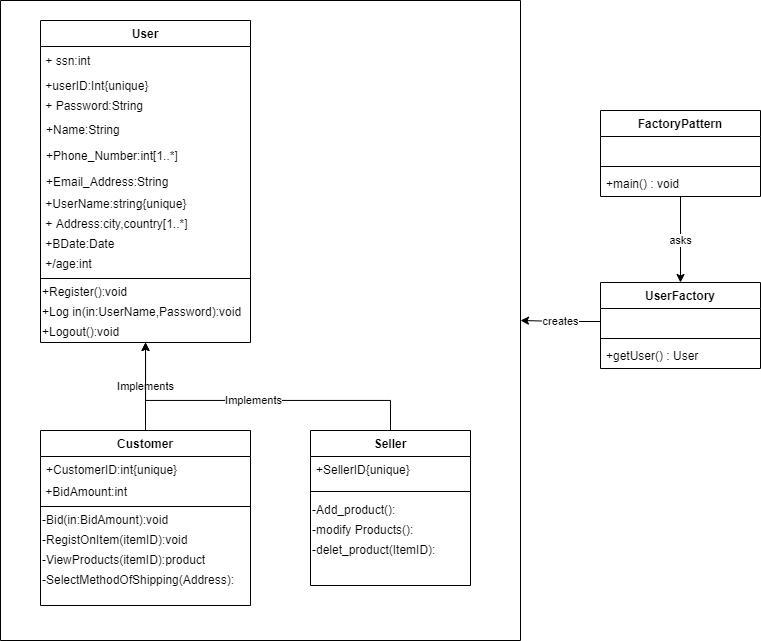
## Problem:

We create object without exposing the creation logic to the client.

## Solution:

We're going to create a User interface and concrete classes implementing the Userinterface. A factory class UserFactory is defined as a next step.

## Structure:



## Participants:

User, customer and seller.

## Consequences:

FactoryPattern class will use UserFactory to get a User object. It will pass information (Customer / Seller) to UserFactory to get the type of object it needs.