Task1. Implement Stock Exchange project with 2 players.

There are two players on Stock Market at the moment: RedSocks and Blossomers. They would like to sell or buy stock. Market model is simplified. There are rules that should be applied on the market:

1. Player can't buy it's own stock shares. Player can buy only other player's shares.
2. Each stock has a unique name.
3. Stock offers from different players are matched by name and number of shares.
4. If Player buys stock and there is other player's selling offer that matches with buying request, the deal is made. If there are multiple selling offers which satisfy buy request, any request is used.
5. If Player sells stock and there are buying requests which match with selling request, the deal is made. If there are multiple buying offers which satisfy sell request, any request is used.

Example.

'Player 1' sells 10 shares of 'MST'. – deal is not made since there is no matching buy requests.

'Player 1' buys 10 shares of 'MST'. – deal is not made even though there is a matching sell request, but player 1 can't by it's own shares

'Player 2' sells 10 shares of 'MST' – deal is made since there is a matching buy request from player1

'Player 2' sells 10 shares of 'MST' – deal is not made since previous matching request from player 1 was already sold.

```
public class RedSocks
{
    public bool SellOffer(string stockName, int numberOfShares)
    {
        throw new NotImplementedException();
    }

    public bool BuyOffer(string stockName, int numberOfShares)
    {
        throw new NotImplementedException();
    }
}
```

SellOffer should return true if a matching buy request was found and stock was sold, and false otherwise.

BuyOffer should return true if a matching sell request was found and stock was bought, and false otherwise.

The same interface is used for second player:

```
public class Blossomers
```

```
{
    public bool SellOffer(string stockName, int numberOfShares)
    {
        throw new NotImplementedException();
    }

    public bool BuyOffer(string stockName, int numberOfShares)
    {
        throw new NotImplementedException();
    }
}
```

If SellOffer and BuyOffer are called multiple with the same parameters, separate offer requests should be created.

SellOff('MSC', 12)

SellOff('MSC', 12) Create two separate sell offers.

Task2. Implement Stock Exchange project with 3 players using Mediator.

We would like to add a new player RossStones to our StockExchange.

```
public class RossStones
{
    public bool SellOffer(string stockName, int numberOfShares)
    {
        throw new NotImplementedException();
    }

    public bool BuyOffer(string stockName, int numberOfShares)
    {
        throw new NotImplementedException();
    }
}
```

Code should not be copy-pasted among players and players should not be tightly-coupled. For example, RossStones should not know anything about players: RedSocks and Blossomers. Mediator pattern should be used. Assign a unique id to each player instance.

Task3. Add ability to notify players when it's buy or sell request is matched.

When Stock Exchange Mediator matches one player's buy request with other player's sell request, these two players should be notified about the deal. Add two properties to IPlayer interface:

```
public interface IPlayer
{
    int SoldShares { get; }

    int BoughtShares { get; }
}
```

SoldShares – is number of shares that were sold by this player

`BoughtShares`– is number of shares that were bought by this player

For example:

1. Blossomers.BuyOffer(“RTC”, 2) – after this operation nothing happens because there is not match for this buy request.
   Blossomers.BoughtShares is 0

2. RedSocks.SellOffer(“RTC”, 2) – In this case, this sell request should be matched with Blossomer’s buy request and a deal should be made. After that, RedSocks should be notified that it’s shares have been sold and Blossomers should be notifies that it’s buy request was fulfilled.
   After this call, we should receive:
   Blossomers.BoughtShares is 2
   RedSocks.SoldShares is 2

Use Observer pattern.

Task4. Implement Task3 using C# events.