To implement the **Publish-Subscribe (Observer)** design pattern for the online auction site described in Problem 2.31 of Marsic (2012), we must identify which entities publish updates (publishers) and which receive those updates (subscribers). The pattern is particularly relevant for the **bidding** and **auction closing** processes.

Publisher Interface Implementation

The **ItemInfo** class should implement the **Publisher** interface.

Justification:

- Each ItemInfo instance represents an item being auctioned. As bids are placed, interested buyers (those who placed previous bids) should be notified of updates such as:
    - A new higher bid
    - The auction closing
    - A winner being selected
- Since ItemInfo maintains the BidsList, it is in the best position to notify all relevant subscribers about state changes like new bids or auction status.
- Therefore, ItemInfo should include:
    - A method to register subscribers (buyers)
    - A method to remove subscribers
    - A method to notify all subscribers of updates

Subscriber Interface Implementation

}

The **BuyerInfo** class should implement the **Subscriber** interface.

Justification:

- Buyers are interested in updates about items they bid on. When an auction receives a new bid or is closed, buyers need to be notified to decide whether to increase their bid or acknowledge the outcome.
- By implementing the Subscriber interface, BuyerInfo objects can register interest in specific ItemInfo instances.
- For example:
    - If Buyer A bids on Item X, Buyer A registers as a subscriber to Item X.
    - When another buyer places a higher bid or the auction closes, Buyer A is notified.

Additional Design Enhancements

To implement this pattern, you can introduce the following:

Interfaces

```
interface Publisher {
   void registerSubscriber(Subscriber s);
   void removeSubscriber(Subscriber s);
   void notifySubscribers(String message);
}

interface Subscriber {
   void update(String message);
}
```

Modifications

- ItemInfo implements Publisher
- BuyerInfo implements Subscriber
- Add a List<Subscriber> in ItemInfo to track registered buyers.

Example Flow

1. Buyer places a bid on an item.
2. The ItemInfo instance registers this buyer as a subscriber.
3. When another bid is placed, or the auction is closed, ItemInfo calls notifySubscribers().
4. All registered BuyerInfo instances receive an update() call with relevant information.

Conclusion

Applying the Publish-Subscribe pattern enhances decoupling between item state changes and buyer reactions, promoting flexibility and scalability. As Marsic notes, "object-oriented design promotes high cohesion within objects and low coupling between objects" (Marsic, 2012, p. 166), and this pattern embodies that principle by separating the auction logic from the buyer notification system.

Reference:

Marsic, I. (2012, September 10). *Software Engineering*. Rutgers: The State University of New Jersey, p. 166. https://my.uopeople.edu/pluginfile.php/57436/mod_book/chapter/46513/CS4403MarsicTextbook.pdf