線上拍賣系統 — 新架構表

> 說明：此文件為「新版系統架構表」，以課程專題與實作需求為出發點，包含元件分層、類別摘要、流程序列、資料模型與擴充建議。使用繁體中文撰寫，適合放入報告或附在程式碼繳交資料夾。

一、系統概觀

本系統為一個簡化但完整的線上拍賣平台（Online Auction System），設計原則為：清晰模組化、物件導向、易於測試與擴充。主要支援：手動出價、AutoBid（自動競標）、拍賣延長（soft-close）、觀察者通知（由 Auctioneer 統一對外宣告）、出價歷史記錄與結果回傳（BidResult）。

目標使用情境：課堂專題展示、簡易模擬真實拍賣行為、能清楚說明核心演算法與設計決策。

二、模組分層（高階）

1. **Presentation（展示層）**
2. Main（命令列示範或測試入口）
3. Auctioneer（統一公開宣告）
4. **Application（應用層）**
5. AuctionController（可選，封裝外部呼叫的 API）
6. NotificationService（若需分離 Auctioneer 行為）
7. **Domain（領域核心）**
8. Listing / StandardAuction
9. User / Seller / Buyer
10. Bid / AutoBid / BidResult
11. Observer（介面）
12. **Infrastructure（基礎設施）**
13. InMemoryStore（暫存資料）
14. TimeProvider（方便測試的時間依賴注入）
15. **Integration / Extension**
16. Persistence（資料庫）
17. Messaging（Email, WebSocket）
18. Scheduler（定期任務、結標觸發）

三、主要類別清單（摘要）

- **User (abstract)**
- 屬性：id:String, name:String
- 方法：getId(), getName()
- **Seller extends User**
- 說明：上架者
- **Buyer extends User implements Observer**
- 說明：出價者，可接收私人通知（可選）
- 方法：update(String message)
- **Auctioneer implements Observer**
- 說明：唯一公開敘事者，負責印出拍賣公示訊息
- 方法：update(String message)
- **Listing (abstract)**
- 屬性：title, seller, endTime
- 方法：isExpired()
- **StandardAuction extends Listing**
- 屬性：highestBid, bidHistory(List<Bid>), watchers(List<Observer>), autoBids(List<AutoBid>), minIncrement
- 方法：placeBid(Buyer, double):BidResult, addBidInternal(Bid), registerAutoBid(AutoBid), addWatcher(Observer), announce(String)
- **Bid**
- 屬性：buyer, amount, time
- **AutoBid**
- 屬性：bidder, maxAmount, (auction)
- 方法：compete(Bid incoming):boolean, setAuction(StandardAuction)
- **BidResult**
- 屬性：isHighest, extended, currentHighestAmount, currentHighestBuyerName, rank

- **Observer (interface)**
- 方法：update(String message)
- **TimeProvider (介面，可選)**
- 方法：now():LocalDateTime

四、數據模型（Entity）

```
User { id, name }
Seller extends User
Buyer extends User
Listing { id, title, sellerId, endTime }
Auction (StandardAuction) { listingId, minIncrement, highestBidId }
Bid { id, buyerId, amount, time }
AutoBid { id, buyerId, maxAmount }
```

備註：在進階實作可將 above entities 對應到 RDBMS table 或 NoSQL document。

五、關鍵流程序列（PlaceBid 與 AutoBid 競爭）

5.1 placeBid(Buyer, amount) — 高階步驟

1. 請求進入（由 Presentation 或 Controller 呼叫）
2. 驗證拍賣未結束 → 否則 `IllegalStateException("Auction has ended")`
3. 驗證出價者非賣家本人 → 否則 `IllegalArgumentException("Seller cannot bid on own listing")`
4. 計算最低可出價：若無最高出價，minAllowed = minIncrement；否則 = highest + minIncrement
5. 若 amount < minAllowed → `IllegalArgumentException("Bid must be at least " + minAllowed)`
6. 建立 Manual Bid (Bid) → 呼叫 `addBidInternal(bid)`
7. 更新 highestBid、加入 bidHistory
8. 通知先前最高出價者被超越（拍賣官 announce）
9. 拍賣官公布最新最高出價
10. 若距結束 < 5 分鐘 → 延長結束時間 5 分鐘，並 announce 延長
11. 執行 AutoBid 回合：對每一個註冊的 AutoBid 執行 `compete(currentHighest)`，若 auto 成功出價（回傳 true）→ 重新從頭檢查 AutoBid 列表，直到無人能超越
12. 計算並回傳 BidResult（isHighest、extended、currentHighestAmount、buyerName、rank）

5.2 AutoBid.compete(incomingBid) — 規格

- 若 incoming.amount < maxAmount：
- proposed = min(incoming.amount + minIncrement, maxAmount)
- 若 proposed > currentHighest.amount → 產生新的 Manual Bid（autoPlaced），呼叫 `auction.addBidInternal(autoPlaced)`，回傳 `true`

- 否則不出價，回傳 `false`
- 若 incoming.amount >= maxAmount：
- 不出價，auction.announce("買家「X」的自動上限 Y 元已被 A 以 Z 元超越"), 回傳 `false`

六、觀察者（Notification）策略

- **公開通知（public announcements）**：只由 `Auctioneer`（拍賣官）發表，所有觀眾（UI、log）皆可看到。避免每個 Buyer 重複印出相同訊息。
- **私人通知（private notifications）**：可選，僅通知被超越的前最高出價者（存入該 Buyer 的訊息清單或以 callback 發送）。

建議：在作業版本中採用「拍賣官公布」為主，私人通知可實作為額外功能。

七、API / 方法簽章（供整合測試或報告使用）

- `BidResult StandardAuction.placeBid(Buyer buyer, double amount)`
- `void StandardAuction.addWatcher(Observer o)`
- `void StandardAuction.registerAutoBid(AutoBid auto)`
- `void AutoBid.setAuction(StandardAuction auction)`
- `boolean AutoBid.compete(Bid incoming)`
- `void StandardAuction.announce(String message)`

八、非功能需求（NFR）

- **可測試性**：提供 `TimeProvider`（或在測試中使用可控 LocalDateTime）以模擬結束時間與延長邏輯。
- **可讀性**：所有公開訊息使用 Auctioneer 統一陳述；程式碼以註解說明流程
- **可擴充性**：AutoBid、ReserveAuction、DutchAuction 等型別可作為 Listing 的子類別擴充
- **效能**：基礎版以記憶體結構儲存，面對大量使用者需改為 DB + queue。

九、部署建議（教學示範版）

- 開發/示範：單一 JVM、命令列或簡單 Web 前端（Servlet / Spring Boot）
- 生產 / 進階：
- 使用資料庫儲存 Bid / Listing / User
- 使用消息中介（RabbitMQ / Kafka）或 WebSocket 處理即時通知
- 使用分散式鎖或樂觀鎖確保競標時的資料一致性

十、擴充與改進點（可列於報告）

1. 加入 `ReservePrice`（底價）與 `BuyNow`（直購）邏輯
2. 將通知改為「拍賣官 + 私訊」雙軌模式
3. 針對大量出價者導入優先隊列或 heap 來優化名次查詢

4. 將 AutoBid 改為可設定代理出價策略（漸進式、百分比等）

> 如果你要我，我可以： - 把此架構轉成 UML 類別圖與序列圖（PNG） - 將此文件匯出為 Word / PDF（可直接繳交） - 根據此架構自動生成 Java 類別骨架（每個類別一個 .java）

請告訴我接下來你要我做哪一項：UML、Word / PDF、或直接產生程式骨架？