**Design and analysis of cycle price engine**

**Use case diagram**

**user**

z

Sends request

**PricingEngineController**

Process request

**PricingService**

fetch values

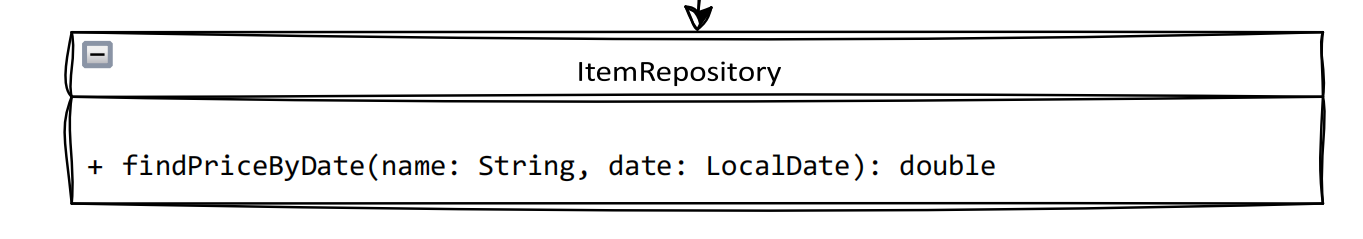
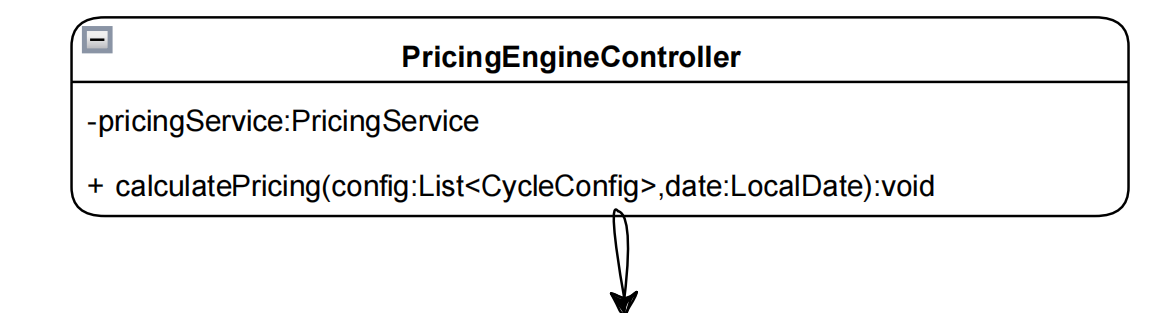
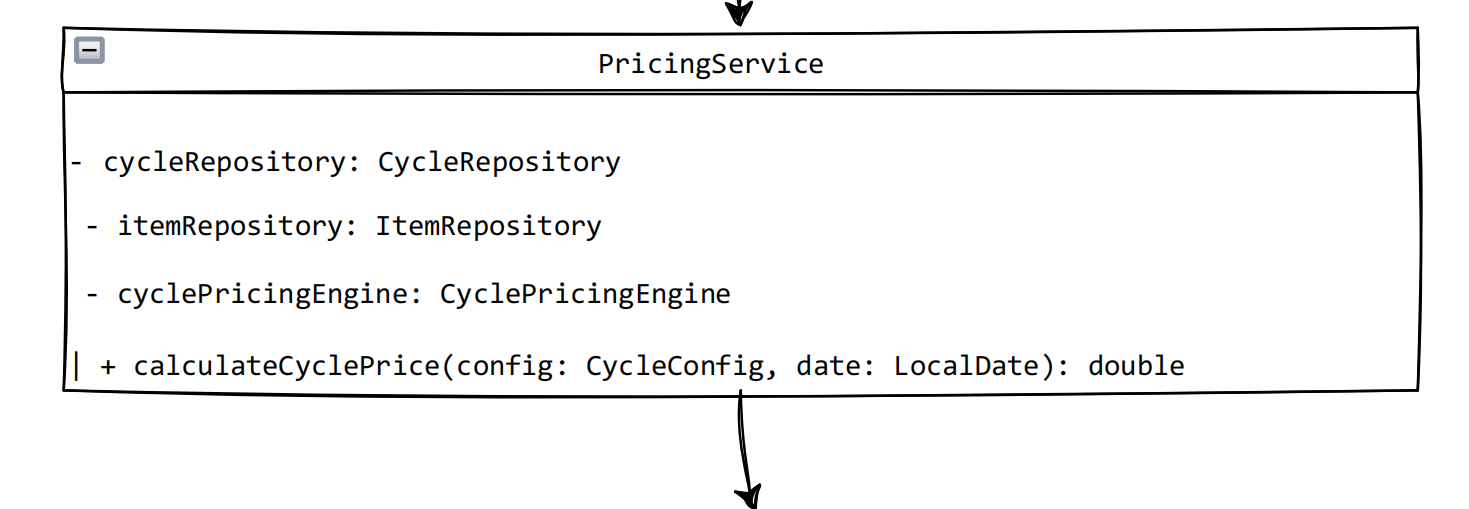
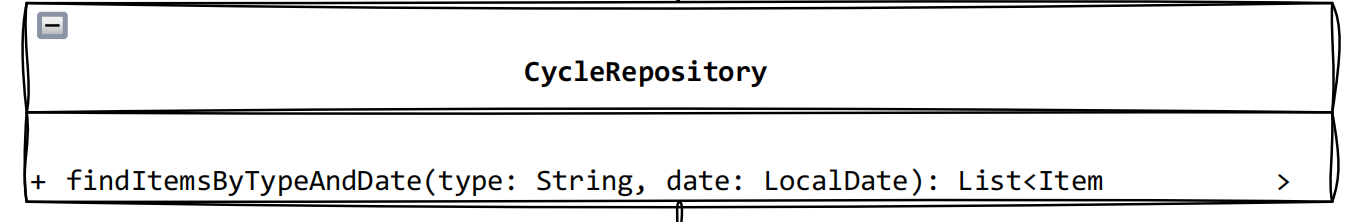
**CycleRepository**

fetch values

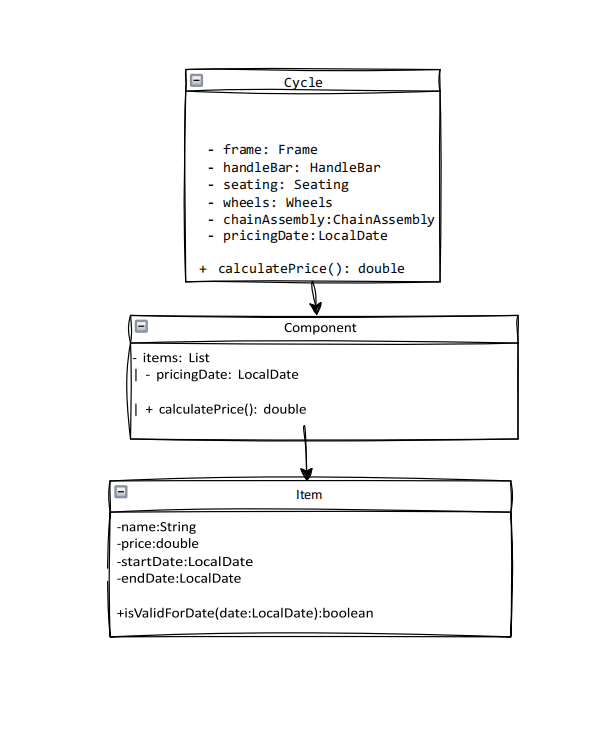
**ItemRepository**

calculate the price

**CyclePricingEngine**



**Class diagram**



**Class diagram(Entity Classes)**

**Sequence Diagram**

**User PricingService Cycle Component Item PriceHistory**

**| | | | | |**

**| ------------- calculatePrice() ------> | | | |**

**| | --- getPrice() -----> | | | |**

**| | | --- getPrice() ----> | | |**

**| | | | --- getPrice() ----> | |**

**| | | | | --- getPrice() ----> |**

**| | | | | <-- return price |**

**| | | | <-- return price |**

**| | | <-- return price |**

**| | <-- return total price |**

**| <-- display total price |**

**Activity Diagram (Multithreading Process for 1000 Cycles)**

**Start**

**|**

**▼**

**Read input (JSON/CLI)**

**|**

**▼**

**Add 1000 CyclePricingTask objects to BlockingQueue**

**|**

**▼**

**Create Thread Pool (10 Threads)**

**|**

**▼**

**[Each Thread]**

**|**

**▼**

**Retrieve task from BlockingQueue**

**|**

**▼**

**Calculate cycle price**

**|**

**▼**

**Store/Display result**

**|**

**▼**

**Repeat until queue is empty**

**|**

**▼**

**Terminate threads**

**|**

**▼**

**End**