DVM

최종 발표

TEAM9 김동민, 송현우

목차

1. 개발환경 (CI/CD, UT)
2. UT 및 System Test 시나리오 및 결과
3. 시스템 동작 Demo
4. OOD(2040)에서 변경/수정된 부분
5. 구현 시 예상보다 어려웠던 점
6. 구현 시 예상보다 쉬웠던 점
7. 객체지향개발방법의 장단점 + 개인적 소감

https://github.com/yunuo46/Distributed-vending-machine

CI/CD

Back-End Server

Front-end Server

Unit Testing

Team10 Server
for System Testing

1.개발환경 (CI/CD, UT)

UT 개요

총 68개의 Test

빌드 시 자동으로 테스트 하도록 설정

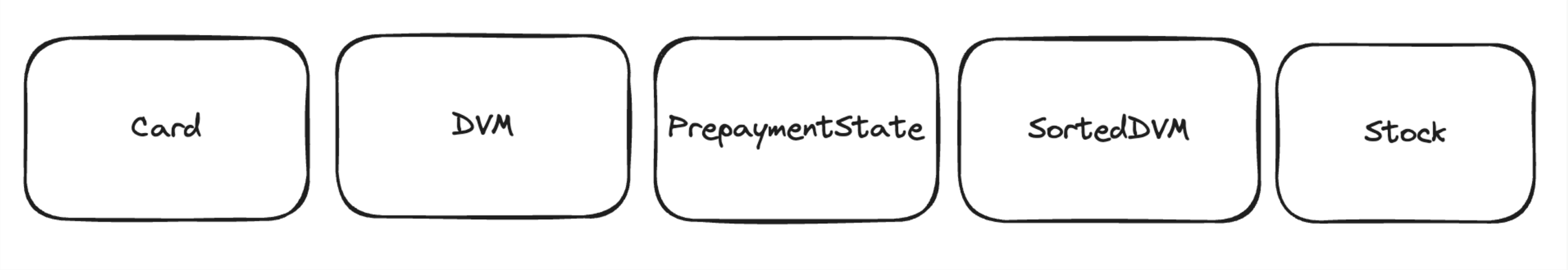
책임이 단순한 도메인부터 개발하며 UT 작성

메서드 단위로 성공 로직, 실패 로직을 테스트

2. UT 및 System Test 시나리오 및 결과

UT 1
data 로직 관련

H2 In-Memory 데이터 베이스 생성하여
자료 검색 및 접근 속도 향상



2. UT 및 System Test 시나리오 및 결과

UT 2

통신 로직 관련

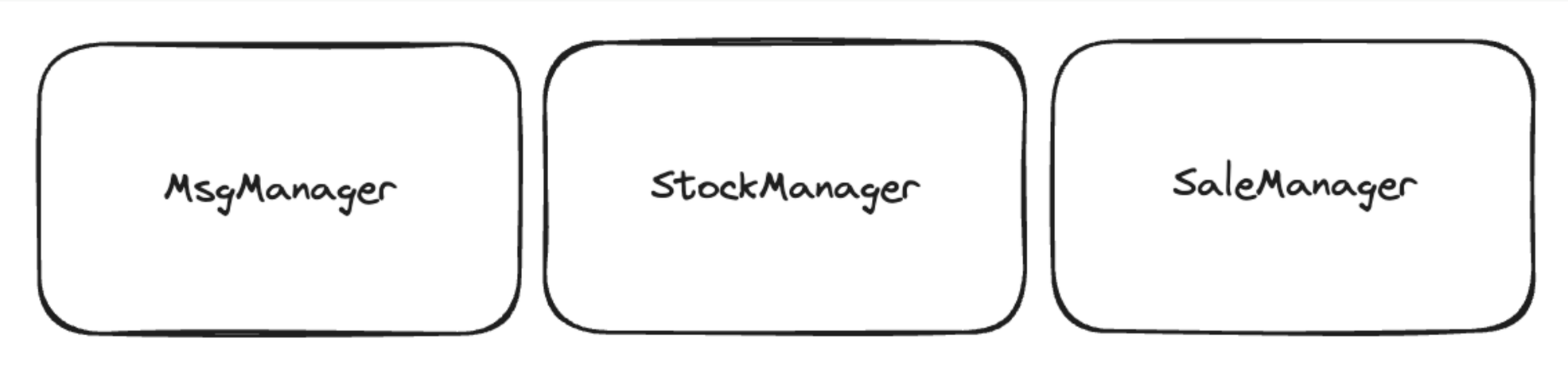
-> request, response Test

-> ConnectionSocket 테스트 -> 변명 테스트 시간에 jsonObject 어쩌구 스켈레톤 코드 참조했는데 String과 타입이 안맞아서 못함

1조님들 죄송해요

2. UT 및 System Test 시나리오 및 결과

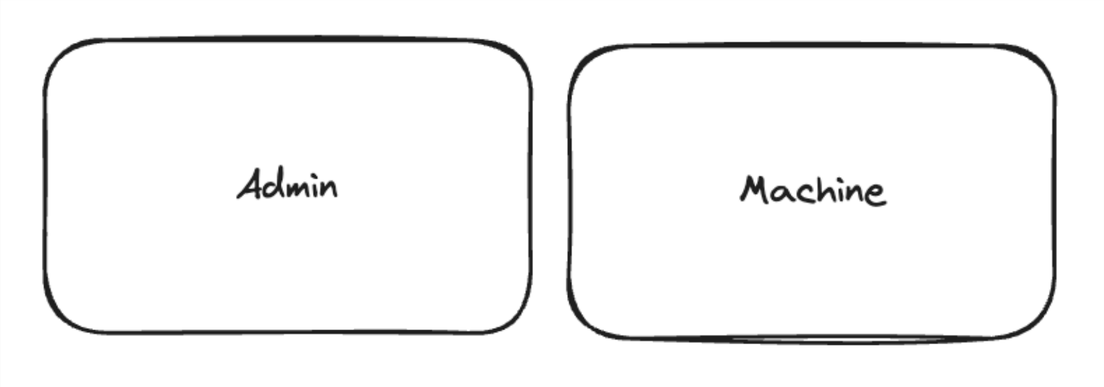
UT 3
Service 로직



-> private메서드는 reflection을 이용

2. UT 및 System Test 시나리오 및 결과

UT 4
Controller 로직



-> 환경변수 설정에서 좀 애를 먹음

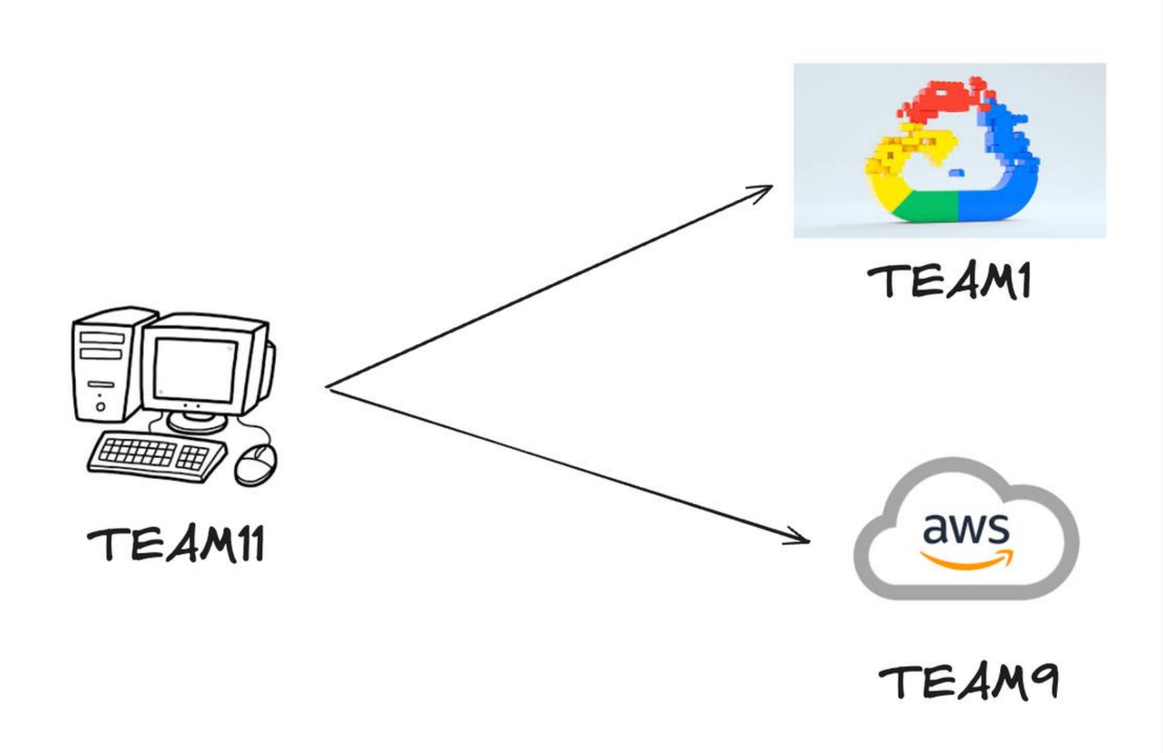
-> junitpioneer API 이용

-> JAVA 16부터 reflection 접근이 엄격하게 설정되어 있어 JVM 설정에서 권한을 부여해서 해결

2. UT 및 System Test 시나리오 및 결과

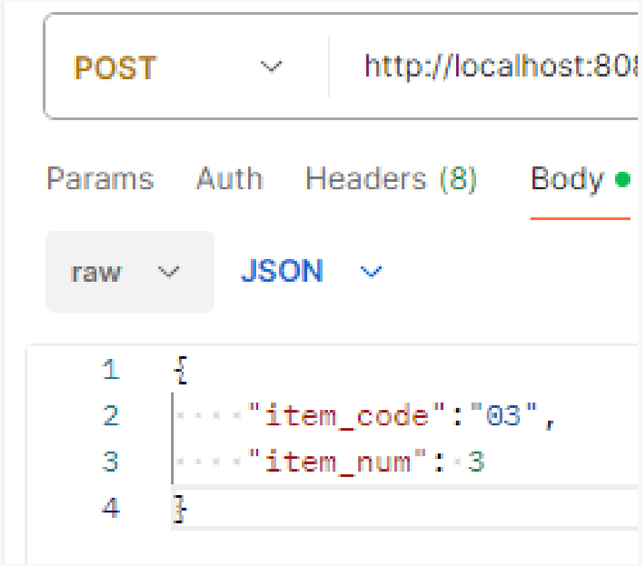
System Test : Sorting Test

	iddvm	id	ip	port
	1	Team9	15.164.4.234	8888
	2	Team1	34.22.98.249	8888
▶*	NULL	NULL	NULL	NULL



2. UT 및 System Test 시나리오 및 결과

System Test :




```
Database connected!
HTTP Server started on port 8080
Waiting for connection on port 8888...
Machine Info: Team11{0, 14}
select item api
stock not enough to payment
stock request to Team9
Connect Socket To : 15.164.4.234:8888
sendMessage: {"msg_type":"req_stock","src_i
received message : {"msg_type":"resp_stock"
Team9 have sufficient stock
stock request to Team1
Connect Socket To : 34.22.98.249:8888
sendMessage: {"msg_type":"req_stock","src_i
received message : {"msg_type":"resp_stock"
Team1 have sufficient stock
dvm_id: Team9 dist: 7.615773
dvm_id: Team1 dist: 9.055386
closest DVM: Team9
```

```
Database connected!
HTTP Server started on port 8080
Waiting for connection on port 8888...
Machine Info: Team11{0, 13}
select item api
stock not enough to payment
stock request to Team9
Connect Socket To : 15.164.4.234:8888
sendMessage: {"msg_type":"req_stock","src
received message : {"msg_type":"resp_stock
Team9 have sufficient stock
stock request to Team1
Connect Socket To : 34.22.98.249:8888
sendMessage: {"msg_type":"req_stock","src
received message : {"msg_type":"resp_stock
Team1 have sufficient stock
dvm_id: Team9 dist: 8.062258
dvm_id: Team1 dist: 8.062258
closest DVM: Team1
```

2. UT 및 System Test 시나리오 및 결과

1.admin: 아이템 코드 09 -> 실패

2.admin: 아이템 코드 04, 3개

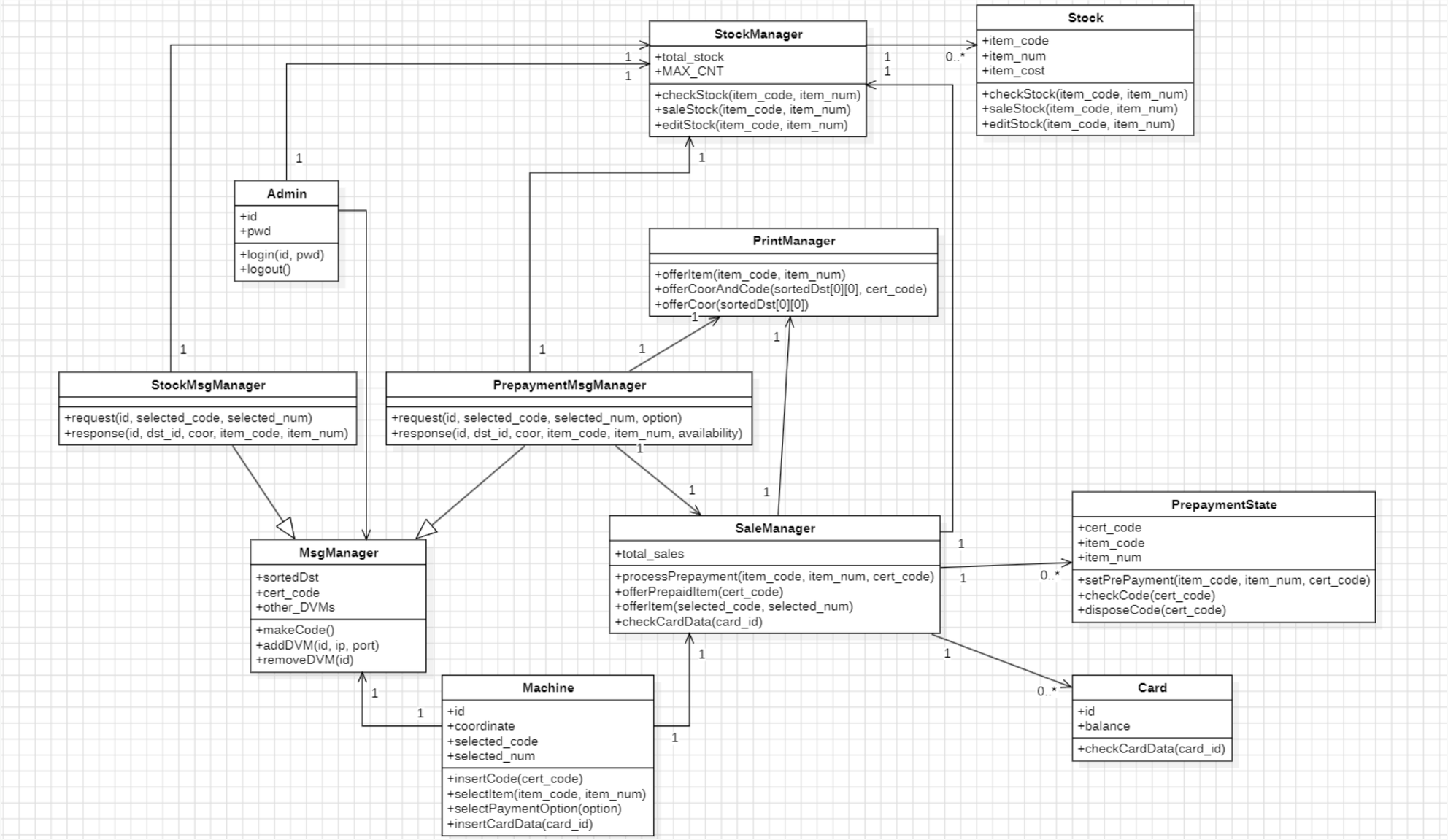
3. select : 04, 2개 구매 -> 성공

4.04 2개 구매 -> 실패 -> 선결제 성공

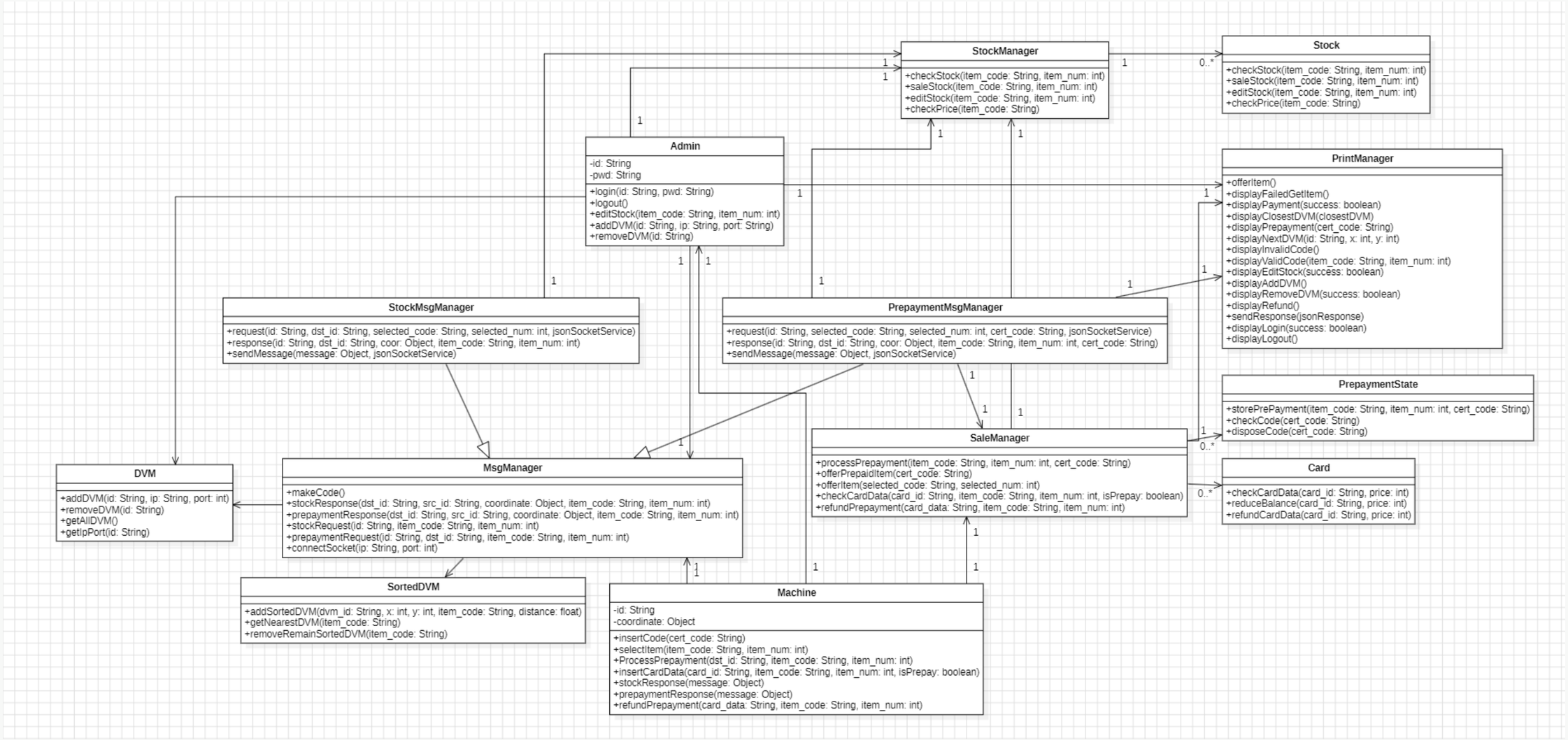
5.04 9개 구매 -> 실패 -> 선결제 실패

6. 젠킨스 로그

3. 시스템 동작 **Demo**



4. OOD(2040)에서 변경/수정된 부분



4. OOD(2040)에서 변경/수정된 부분

- 설계가 생각보다 너무 많이/자주 바뀌었다
- java/junit이 처음이었다
  - 상속, 다형성 등을 어떻게 활용해야할 지 어려웠다
  - 쓰레드를 어떻게 활용해야할 지 어렵다. 특히 메모리 공유하는 부분
- 프레임워크없이 순수 자바 http 통신 및 소켓 통신 구현
  - 자료가 생각보다 많이 없다
  - 유효성 검사, api 라우팅이 매우 복잡
  - Thread를 생성하여 8080 포트로 http 통신 서버
  - 8888 포트로 소켓 서버
- getter, setter 안쓰고 개발하기 -> dto를 제외하고는 성공!
- 동시성 문제

5. 구현 시 예상보다 어려웠던 점

- GUI안쓰고 웹 페이지로 구현해서 간단하게 해결했다
- 설계 기반의 빠른 개발
  - 결합도가 낮은 도메인부터 개발하니까 나중에는 퍼즐맞추는 느낌으로 빠르게 개발할 수 있었다
  - use case 시나리오대로 순차적으로 개발할 수 있었다
- jenkins 생각보다 쉽고 간편하다
  - 스왑메모리 설정으로 prettier에서도 빌드가 가능했다
- 언어 숙련도 문제는 생성형 ai의 도움을 받을 수 있어서 생각보다 격차가 크게 느껴지지 않았다

6. 구현 시 예상보다 쉬웠던 점

장점

1. 높은 생산성
2. 높은 유지보수성
3. 높은 재사용성
4. 테스트 용이

단점

1. 설계에 시간 오래걸림
2. high risk를 숙련된 설계자/개발자가 아니면 파악하기 어려울 수도

개인적 소감

iteration을 한 번만해서 (많이) 아쉬웠다

언어 숙련도가 낮은만큼 프로토타이핑을 좀 더 신경써야했을 것 같다

다음에는 다양한 디자인 패턴도 공부해서 설계에 적용해보고 싶다

재밌었다!!

7. 객체지향개발방법의 장단점 + 개인적 소감

경청해주셔서 감사합니다

TEAM9 김동민, 송현우