

2024-1학기 객체지향개발방법론(3227-001)

OOPT Stage 2050, 2060 - Construct, Testing

Distributed Vending Machine

Team [T4] 202011282 김희준 202011286 남경식 202011377 지상준

목차

Stage 2050. Construct

1. 개발환경 (CI/CD, UT)
2. UT 및 System Test 시나리오 및 결과
3. 시스템 동작 Demo 화면 또는 영상
4. OOD (2040) 에서 변경/수정된 부분 정리
5. 구현 시 예상보다 어려웠던 점
6. 구현 시 예상보다 쉬웠던 점
7. 객체지향개발방법론의 장단점 + 개인적인 소감들

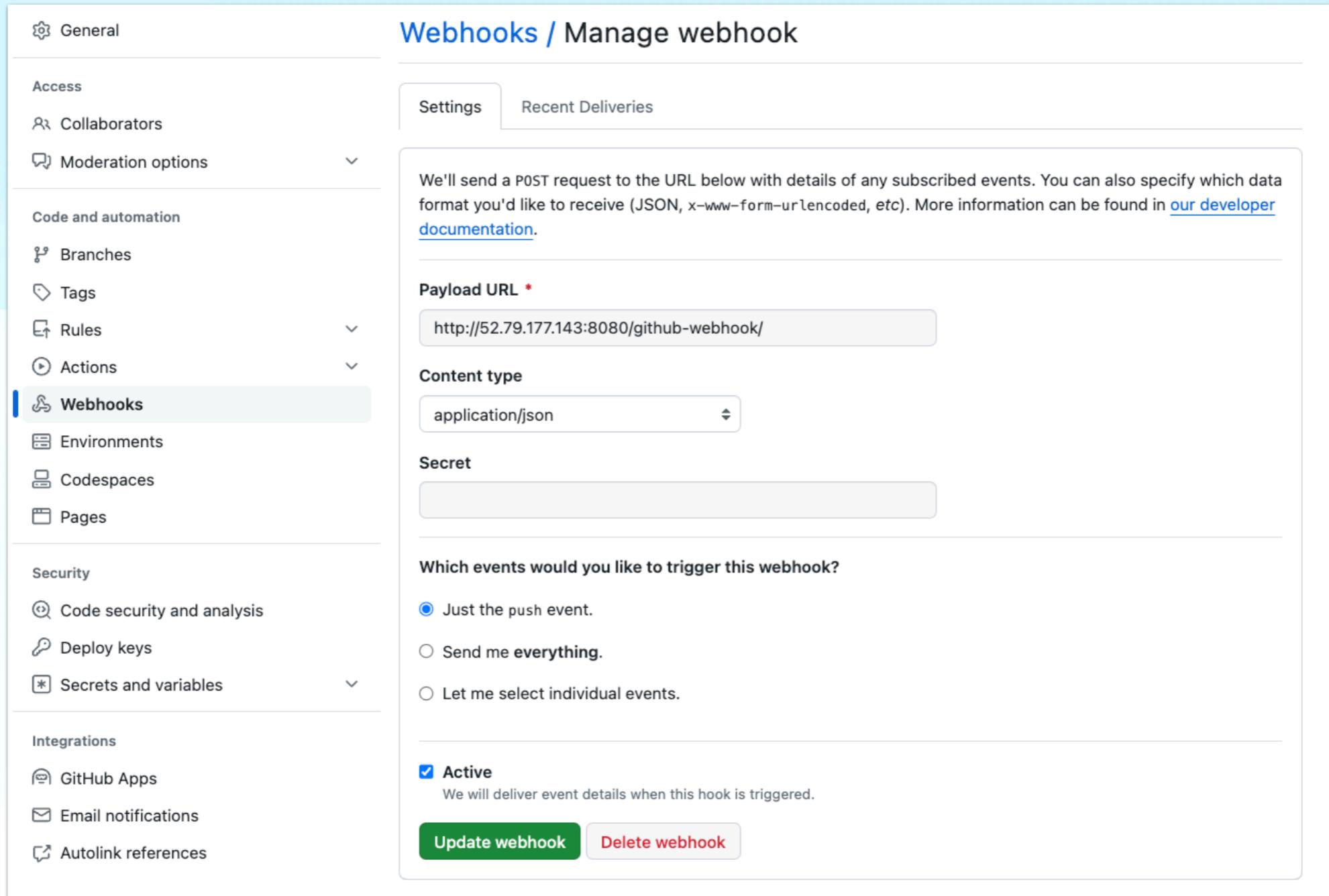
1. 개발환경 (CI/CD, UT)

Jenkins Github 연동, Unit Test 자동화

- AWS EC2 이용해서 Jenkins 사용
- Github Webhook을 이용해서 Jenkins가 변화를 감지
- Jenkins에서 정의되어 있는 Test Case를 바탕으로 Gradle 을 통한 test 자동으로 수행

1. 개발환경 (CI/CD, UT)

Jenkins Github 연동, Unit Test 자동화



The screenshot shows the Jenkins 'Webhooks / Manage webhook' configuration page. The left sidebar contains a navigation menu with categories: General, Access, Code and automation, Security, and Integrations. The 'Webhooks' option is selected. The main content area is titled 'Webhooks / Manage webhook' and has two tabs: 'Settings' (active) and 'Recent Deliveries'. The 'Settings' tab contains the following configuration options:

- Access:** Collaborators, Moderation options (dropdown).
- Code and automation:** Branches, Tags, Rules (dropdown), Actions (dropdown), **Webhooks** (selected), Environments, Codespaces, Pages.
- Security:** Code security and analysis, Deploy keys, Secrets and variables (dropdown).
- Integrations:** GitHub Apps, Email notifications, Autolink references.

The 'Settings' tab configuration includes:

- Settings / Recent Deliveries:** Two tabs, with 'Settings' selected.
- Introduction:** We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).
- Payload URL *:**
- Content type:**
- Secret:**
- Which events would you like to trigger this webhook?:**
 - Just the push event.
 - Send me **everything**.
 - Let me select individual events.
- Active:** **Active**
We will deliver event details when this hook is triggered.
- Buttons:**

1. 개발환경 (CI/CD, UT)

Jenkins Github 연동, Unit Test 자동화

Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

✓ <http://52.79.177.143:8080/github-...> (push)

Edit

Delete

1. 개발환경 (CI/CD, UT)

Jenkins Github 연동, Unit Test 자동화

 Build History 추이 ▾

Filter...

- ✘ #3
| 2024. 6. 8. 오전 3:10
- ✔ #2
| 2024. 5. 25. 오후 2:45
- ✔ #1
| 2024. 5. 25. 오후 2:26

[Atom feed \(전체\)](#) [Atom feed \(실패\)](#)

✔ #10 (2024. 6. 8. 오전 4:16:56)

 Started by [GitHub push by Klngscrown](#)

 This run spent:

- 6.4 sec waiting;
- 29 sec build duration;
- 35 sec total from scheduled to completion.

 **git**
Revision: b253241cf9cfa99029c1f039d587c00e6814fc57
Repository: https://github.com/Klngscrown/OOAD_T4_DVM.git

- refs/remotes/origin/main

 [Test Result](#) (실패가 없습니다)

 Changes

1. build.gradle Edit for Success3 ([details](#) / [githubweb](#))

1. 개발환경 (CI/CD, UT)

Jenkins Github 연동, Unit Test 자동화

Test Result

0 실패 (±0)

4 테스트 (-2)

Took 0.11 sec.

 상세 내용 입력

모든 테스트

Package	실행 시간	실패	(비교)	건너 뛰	(비교)	Pass	(비교)	총	(비교)
org.DVM	0.11 sec	0		0		4	+4	4	+4

2. UT 및 System Test 시나리오 및 결과

Unit Test Case - 결제 관련

- 카드로 결제 시 PaymentManger의 pay()가 잘 동작하는지...

```
@Test @Proxy
public void testPay2() {
    Card card = new Card( cardNum: "4444", cardType: "", balance: 0);
    assertFalse(paymentManager.pay(card, totalCost: 400.0f), message: "Payment should be Failed");
    assertTrue(paymentManager.pay(card, totalCost: 100.0f), message: "Payment should be successful");
}
```

2. UT 및 System Test 시나리오 및 결과

Unit Test Case - 재고확인 관련

- 현재 자판기에 재고가 원하는 수량만큼 존재하는지 체크하는 Stock의 checkStock()이 잘 동작하는지...

```
@Test @Proxy
public void testCheckStock() {
    assertTrue(stock.checkStock( item_code: 1, item_num: 5), message: "Stock should be enough");
    assertFalse(stock.checkStock( item_code: 20, item_num: 6), message: "Stock should not be enough");
}
}
```

2. UT 및 System Test 시나리오 및 결과

Unit Test Case - 인증코드 관련

- 현재 자판기에 재고가 원하는 수량만큼 존재하는지 체크하는 VerificationManager의 verifyVCode()가 잘 동작하는지...

```
@Test  △ Proxy
public void testVerifyVCode() {
    assertTrue(verificationManager.verifyVCode("1234567890"), message: "Verification should be successful");
    assertFalse(verificationManager.verifyVCode("1234567891"), message: "Verification should be Failed");
}
```

2. UT 및 System Test 시나리오 및 결과

System Test Case - 현재 DVM에 사용자가 구매를 희망하는 음료 재고가 존재하는 경우

- ✓ 음료 구매 시도 -> 결제UI -> 카드번호 입력 -> 결제완료 -> 음료 수령
- ✓ 음료 구매 시도 -> 결제UI -> 카드번호 입력 -> 잔고부족 및 한도초과로 결제 실패 -> MainUI

2. UT 및 System Test 시나리오 및 결과

System Test Case - 현재 DVM에 사용자가 구매를 희망하는 음료 재고가 없는 경우

- ✓ 음료 구매 시도 -> 가장 가까운 DVM의 위치정보 제공 -> 카드번호 입력
-> 인증코드 발급 및 구매정보 출력
- ✓ 음료 구매 시도 -> 가장 가까운 DVM의 위치정보 제공 -> 뒤로가기 -> MainUI
- ✓ 음료 구매 시도 -> 가장 가까운 DVM의 위치정보 제공 -> 카드번호 입력
-> 잔고부족 및 한도초과로 결제 실패 -> MainUI

2. UT 및 System Test 시나리오 및 결과

System Test Case - 다른 DVM으로부터 선결제 요청을 받고 인증코드를 입력하는 경우

- ✓ 유효한 인증코드 입력 -> 음료 수령
- ✓ 유효하지 않은 인증코드 입력 -> 팝업 메시지 -> MainUI

3. 시스템 동작 Demo 화면 또는 영상

MainUI - 음료 구매 / 인증코드 입력

[T4] Distributed Vending Machine				
콜라(01)	사이다(02)	녹차(03)	홍차(04)	밀크티(05)
탄산수(06)	보리차(07)	캔커피(08)	물(09)	에너지드링크(10)
유자차(11)	식혜(12)	아이스티(13)	딸기주스(14)	오렌지주스(15)
포도주스(16)	이온음료(17)	아메리카노(18)	핫초코(19)	카페라떼(20)

<input type="text"/>	수량(0~99)	구매하기
<input type="text"/>	인증코드	선결제 인증코드

3. 시스템 동작 Demo 화면 또는 영상

PaymentUI - 결제

[T4] Distributed Vending Machine

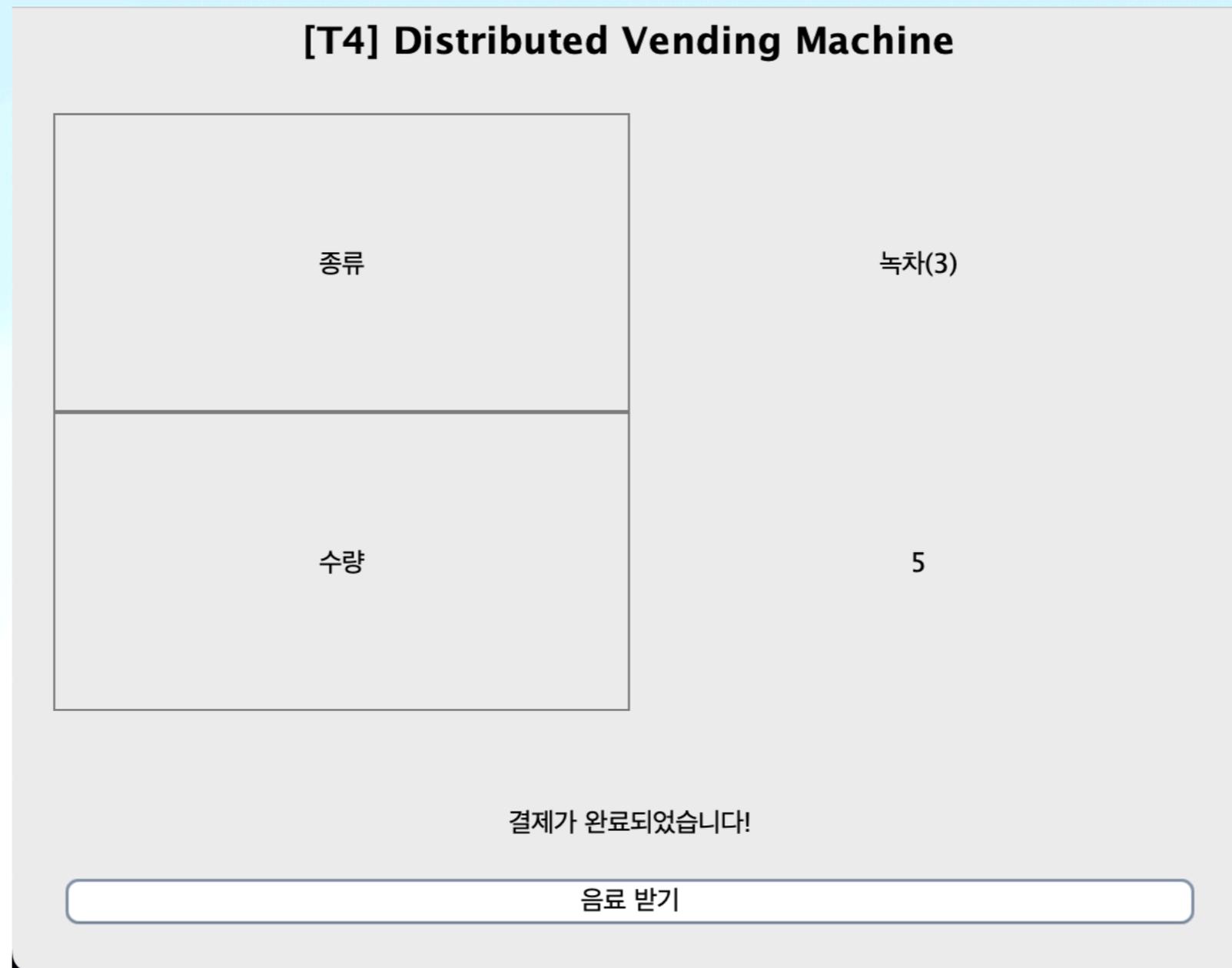
종류	녹차(3)
수량	5

카드 번호를 입력해주세요.

카드번호

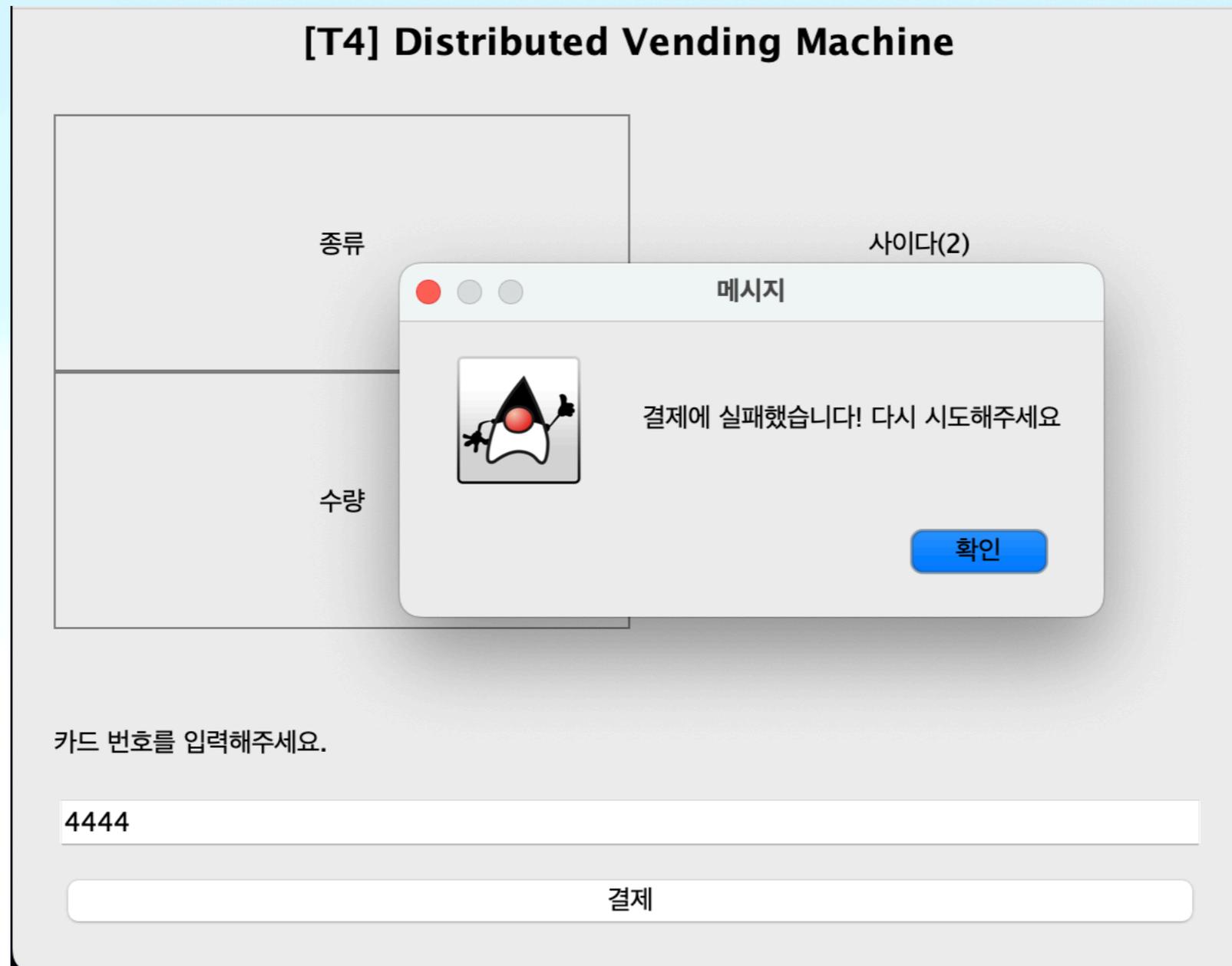
3. 시스템 동작 Demo 화면 또는 영상

카드 결제가 성공



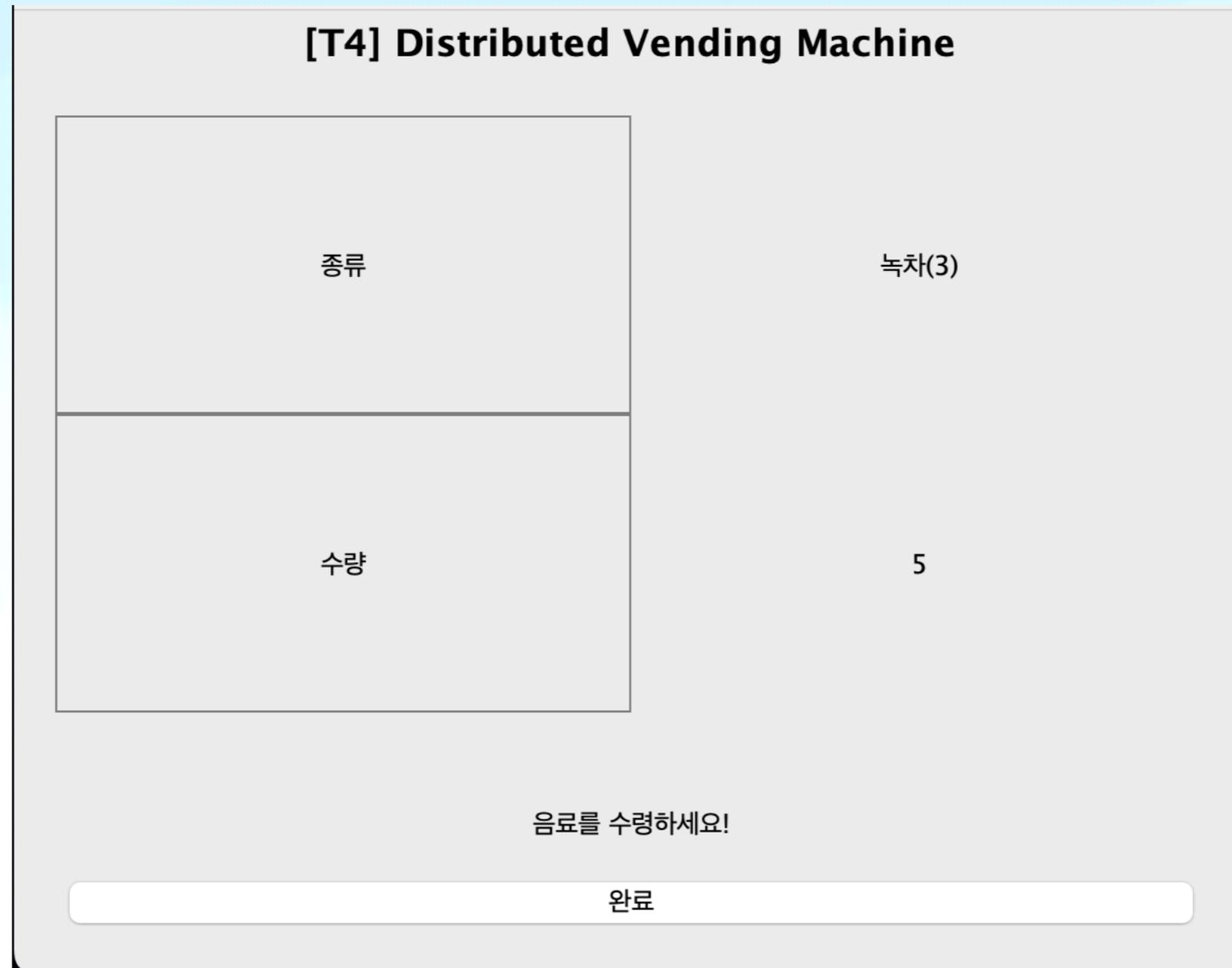
3. 시스템 동작 Demo 화면 또는 영상

카드결제에 실패



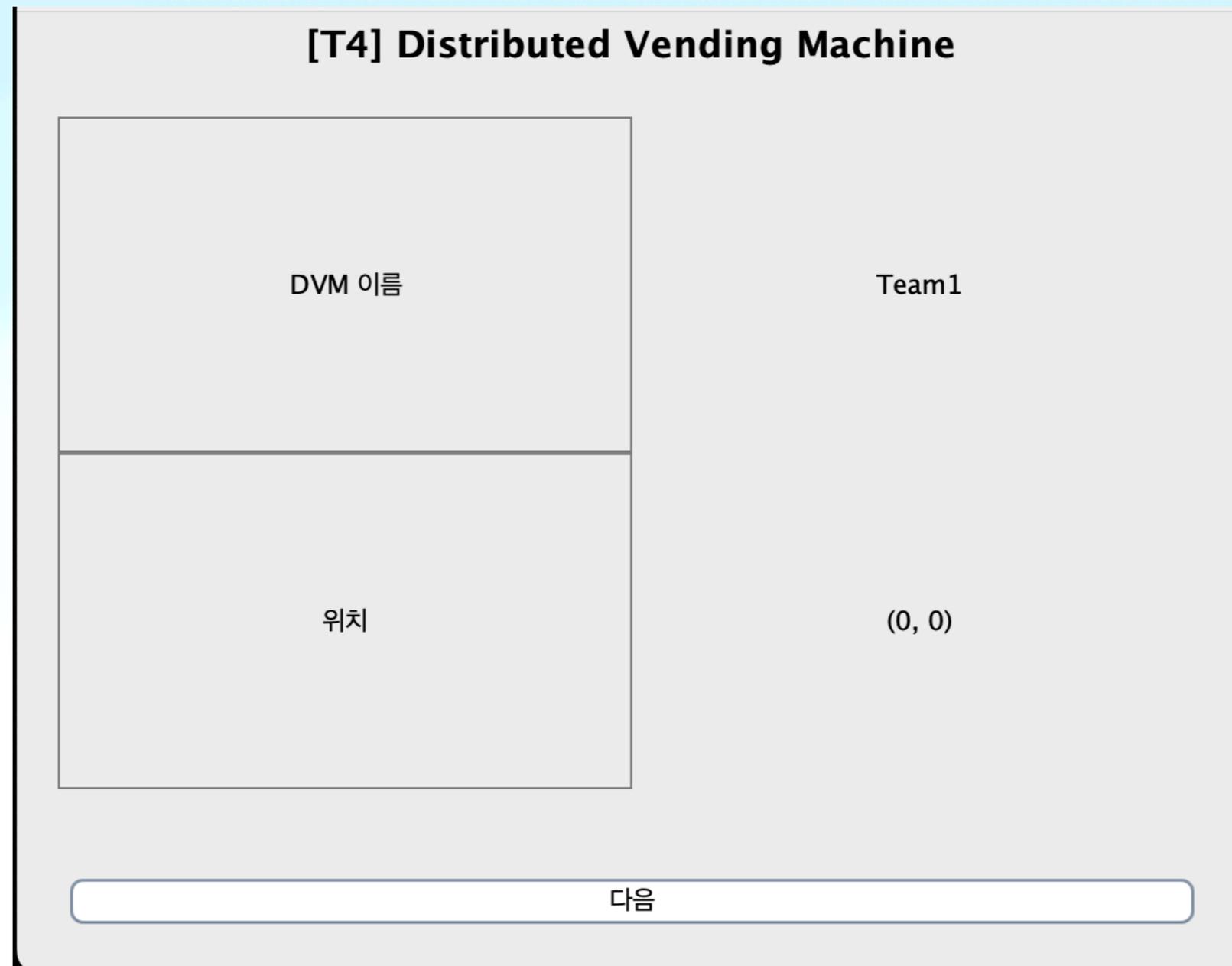
3. 시스템 동작 Demo 화면 또는 영상

DispenseResultUI - 구매한 음료의 정보



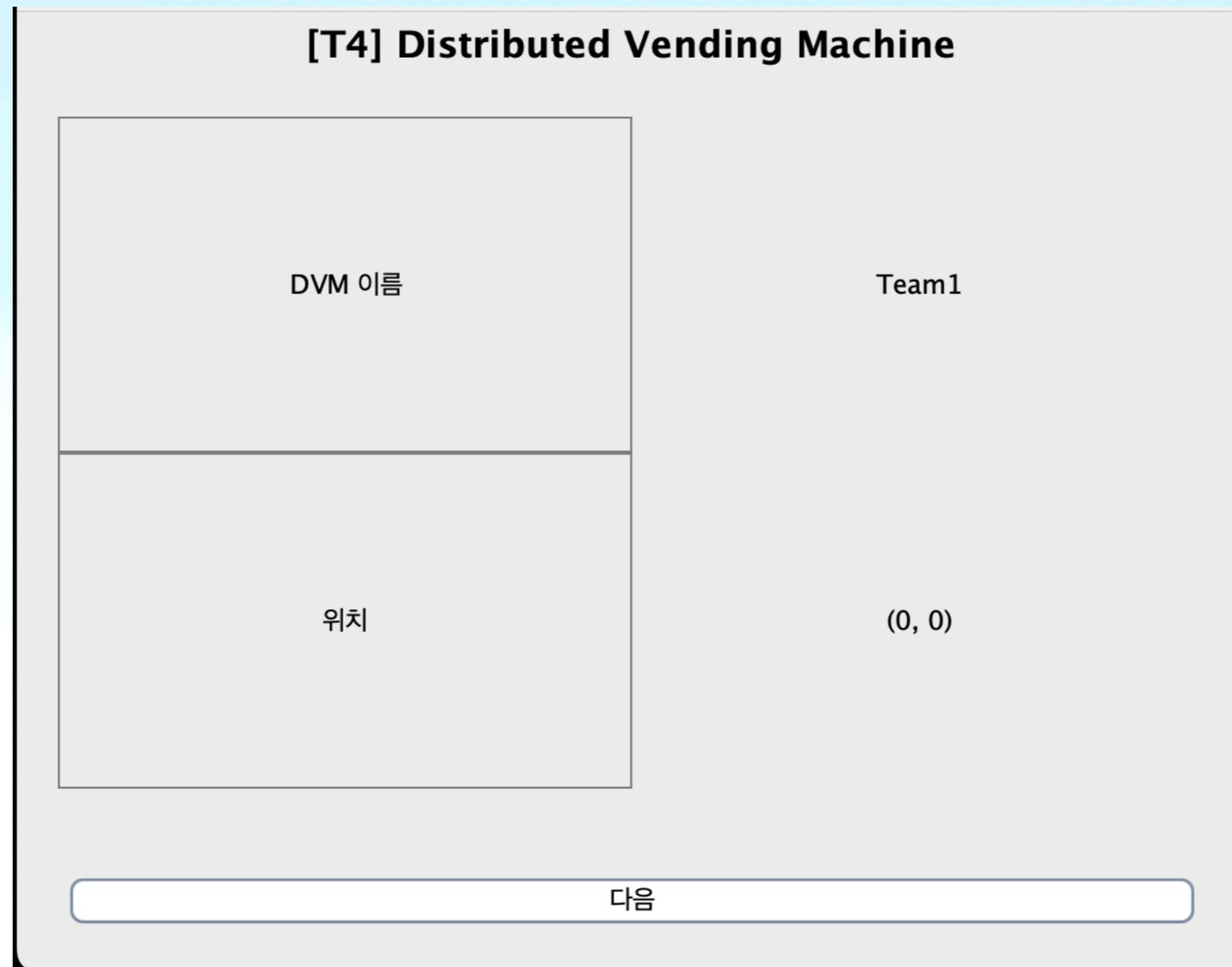
3. 시스템 동작 Demo 화면 또는 영상

LocationinfoUI - 구매하고자 하는 음료의 재고가 없거나 부족한 경우



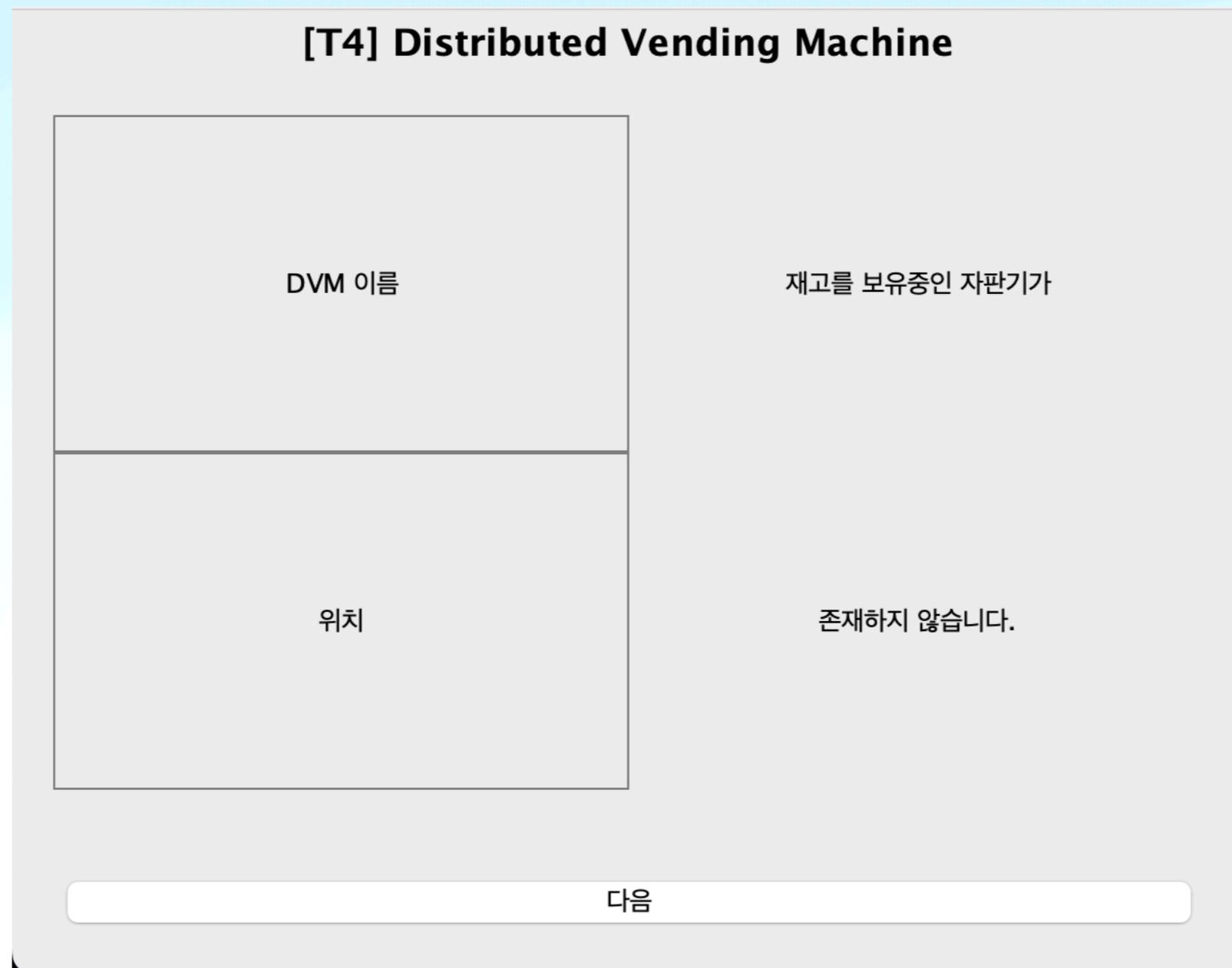
3. 시스템 동작 Demo 화면 또는 영상

LocationinfoUI - 구매하고자 하는 음료의 재고가 없거나 부족한 경우



3. 시스템 동작 Demo 화면 또는 영상

LocationinfoUI - 다른 모든 DVM들 또한 구매하고자 하는 음료의 재고가 없거나 부족한 경우



3. 시스템 동작 Demo 화면 또는 영상

PrepaymentUI - 선결제 희망여부 / 결제

[T4] Distributed Vending Machine

종류	홍차(4)
수량	20
위치	(0, 0)

카드 번호를 입력해주세요.

3. 시스템 동작 Demo 화면 또는 영상

선결제 성공

[T4] Distributed Vending Machine

종류	홍차(4)
수량	20
위치	(0, 0)

결제가 완료되었습니다!

인증 코드 받기

3. 시스템 동작 Demo 화면 또는 영상

인증코드 안내

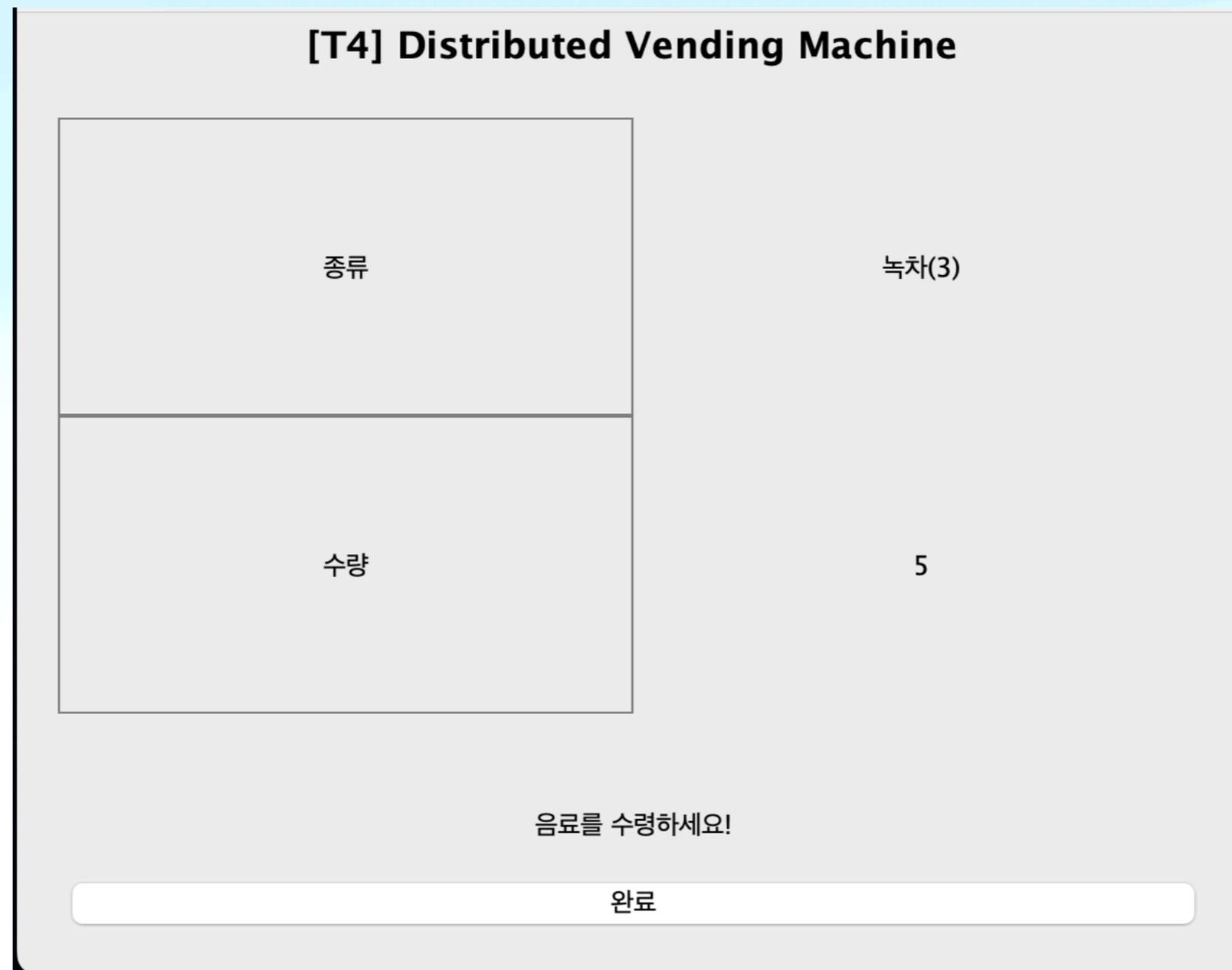
[T4] Distributed Vending Machine

DVM 이름	Team1
위치	(0, 0)
인증코드	a3dkeiE13l

돌아가기

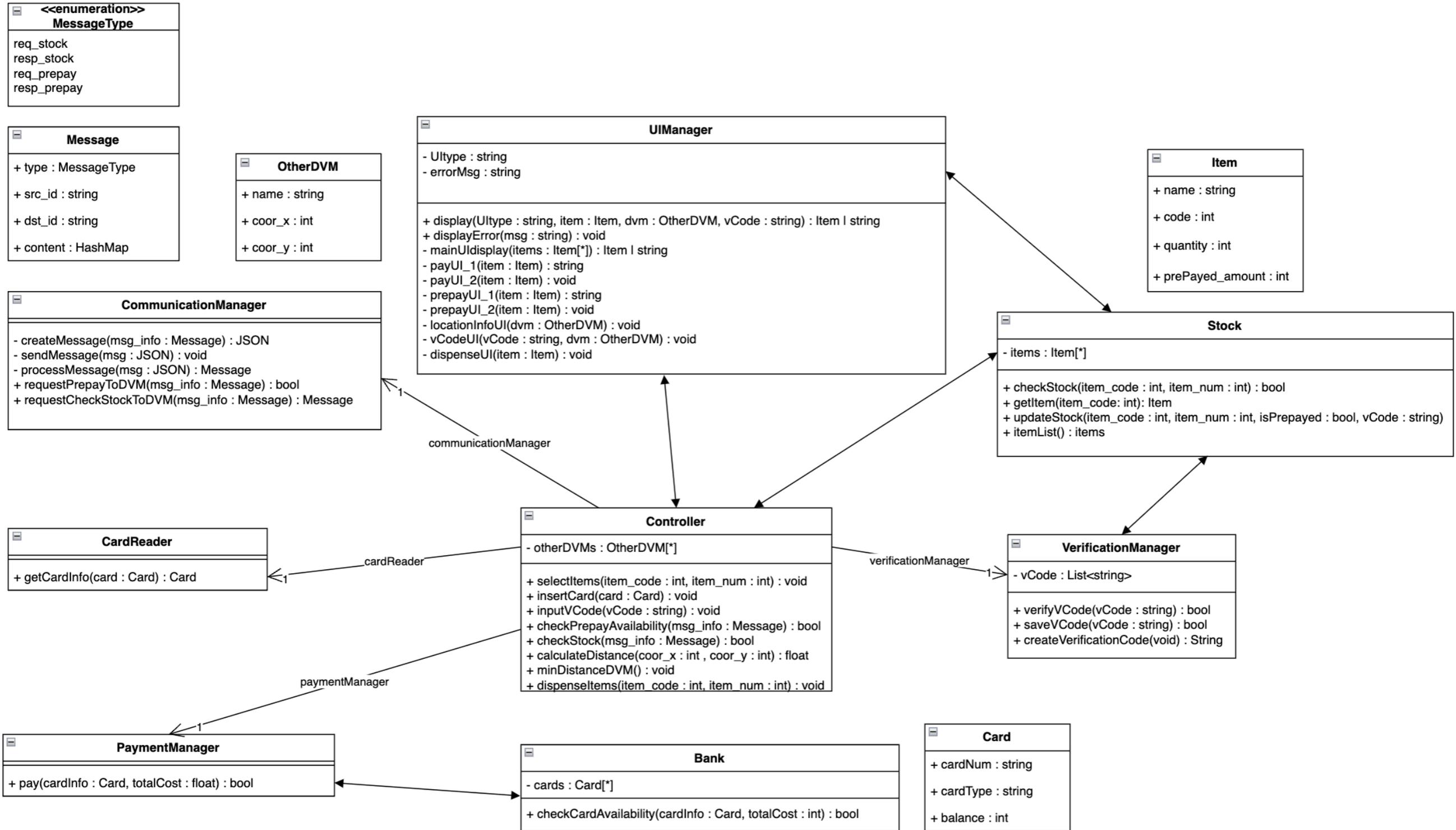
3. 시스템 동작 Demo 화면 또는 영상

DispenseResultUI - 올바른 인증코드 입력



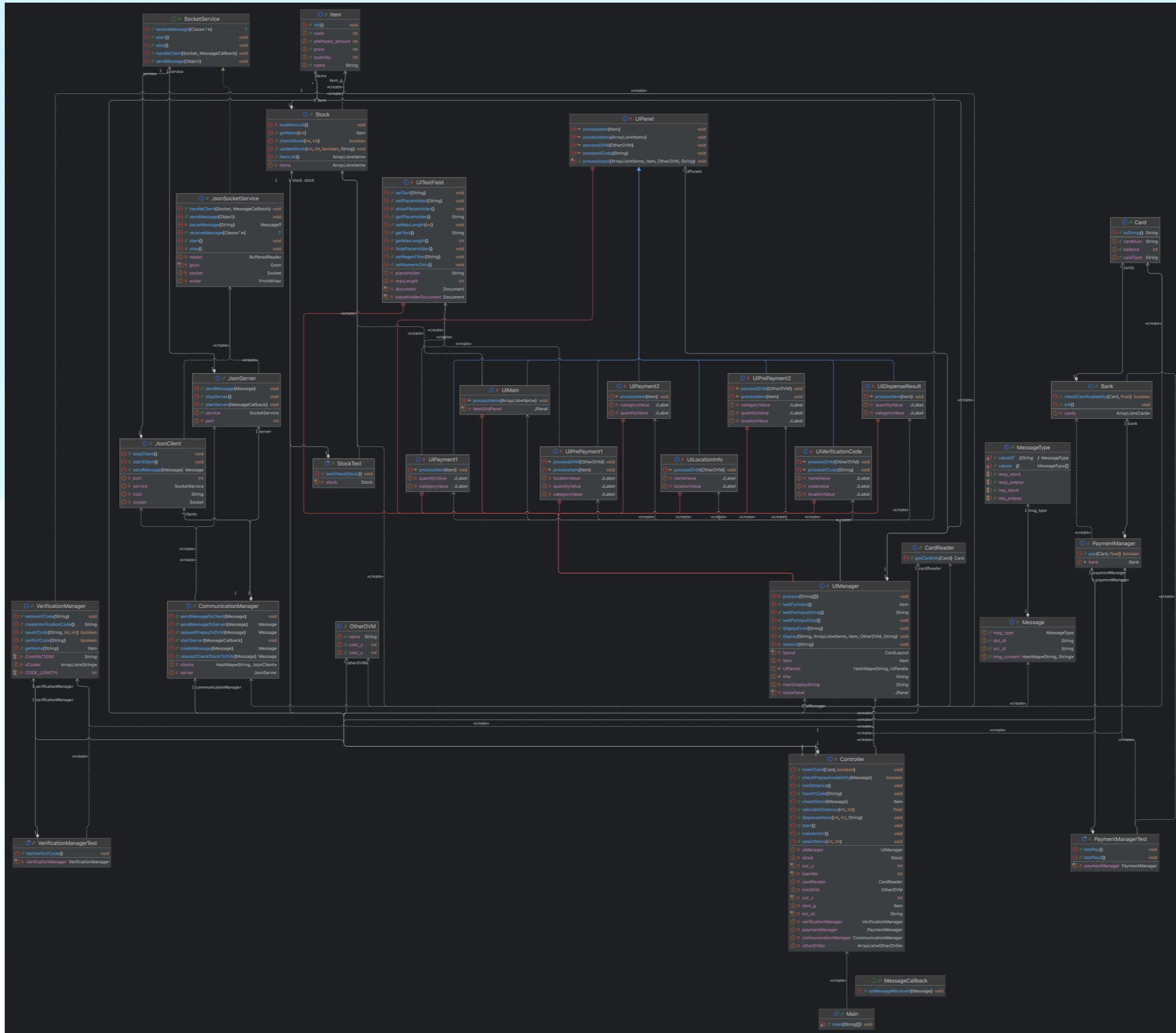
4. OOD (2040) 에서 변경/수정된 부분 정리

Class Diagram at Stage 2040.



4. OOD (2040) 에서 변경/수정된 부분 정리

Class Diagram From Code



4. OOD (2040) 에서 변경/수정된 부분 정리

UIManager at Stage 2040.

UIManager
<ul style="list-style-type: none">- Utype : string- errorMsg : string
<ul style="list-style-type: none">+ display(Utype : string, item : Item, dvm : OtherDVM, vCode : string) : Item string+ displayError(msg : string) : void- mainUdisplay(items : Item[*]) : Item string- payUI_1(item : Item) : string- payUI_2(item : Item) : void- prepayUI_1(item : Item) : string- prepayUI_2(item : Item) : void- locationInfoUI(dvm : OtherDVM) : void- vCodeUI(vCode : string, dvm : OtherDVM) : void- dispenseUI(item : Item) : void

4. OOD (2040) 에서 변경/수정된 부분 정리

UI관련

```
© UIManager
(m) process(String[]) void
(m) waitForInput() Item
(m) waitForInputString() String
(m) waitForInputVoid() void
(m) displayError(String) void
(m) display(String, ArrayList<Item>, Item, OtherDVM, String) void
(m) showUI(String) void
(f) layout CardLayout
(f) item Item
(f) UIPanels HashMap<String, UIPanel>
(f) title String
(f) mainDisplayString String
(f) showPanel JPanel
```

- 각각의 UI 화면들에 해당하는 메서드를 클래스로 재구성하고 showUI를 통해서 UI를 부를 수 있게 변경
- UI 구성에 필요한 텍스트 필드 등의 UI 객체 클래스 생성
- Controller와 정보를 주고 받기 위한 변수와 메서드(mainDisplayString, waitFor~ 등)를 추가하고 일부 필요 없는 변수(UItype, errMsg 등) 삭제

4. OOD (2040) 에서 변경/수정된 부분 정리

Controller at Stage 2040.

Controller
- otherDVMs : OtherDVM[*]
+ selectItems(item_code : int, item_num : int) : void + insertCard(card : Card) : void + inputVCode(vCode : string) : void + checkPrepayAvailability(msg_info : Message) : bool + checkStock(msg_info : Message) : bool + calculateDistance(coor_x : int , coor_y : int) : float + minDistanceDVM() : void + dispenseItems(item_code : int, item_num : int) : void

4. OOD (2040) 에서 변경/수정된 부분 정리

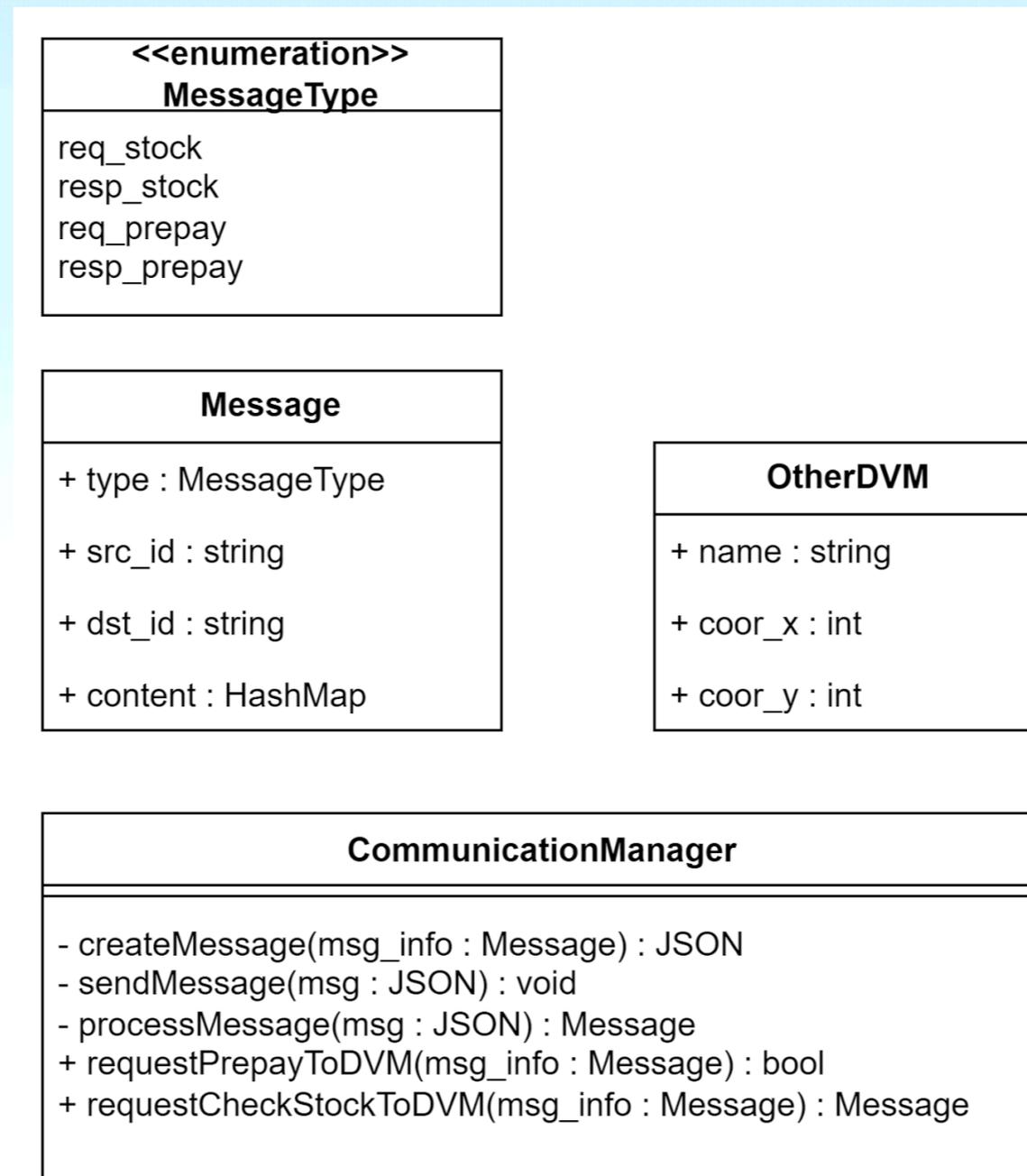
Controller

```
Controller
├── insertCard(Card, boolean) void
├── checkPrepayAvailability(Message) boolean
├── minDistance() void
├── inputVCode(String) void
├── checkStock(Message) Item
├── calculateDistance(int, int) float
├── dispenseItems(int, int, String) void
├── start() void
├── mainAction() void
├── selectItems(int, int) void
├── uiManager UIManager
├── stock Stock
├── our_y int
├── teamNo int
├── cardReader CardReader
├── minDVM OtherDVM
├── our_x int
├── item_g Item
├── src_id String
├── verificationManager VerificationManager
├── paymentManager PaymentManager
├── communicationManager CommunicationManager
├── otherDVMs ArrayList<OtherDVM>
```

- our_x, our_y, teamNo, src_id 등 현재 자판기에 대한 정보 추가
- OtherDVMs의 형식을 단순 배열에서 ArrayList로 변경
- 초기화 및 실행을 위한 start, mainAction 함수 추가
- item_g, minDVM 등 UI에서 정보를 처리하고 표시하기 위해 임시 저장하는 변수 추가

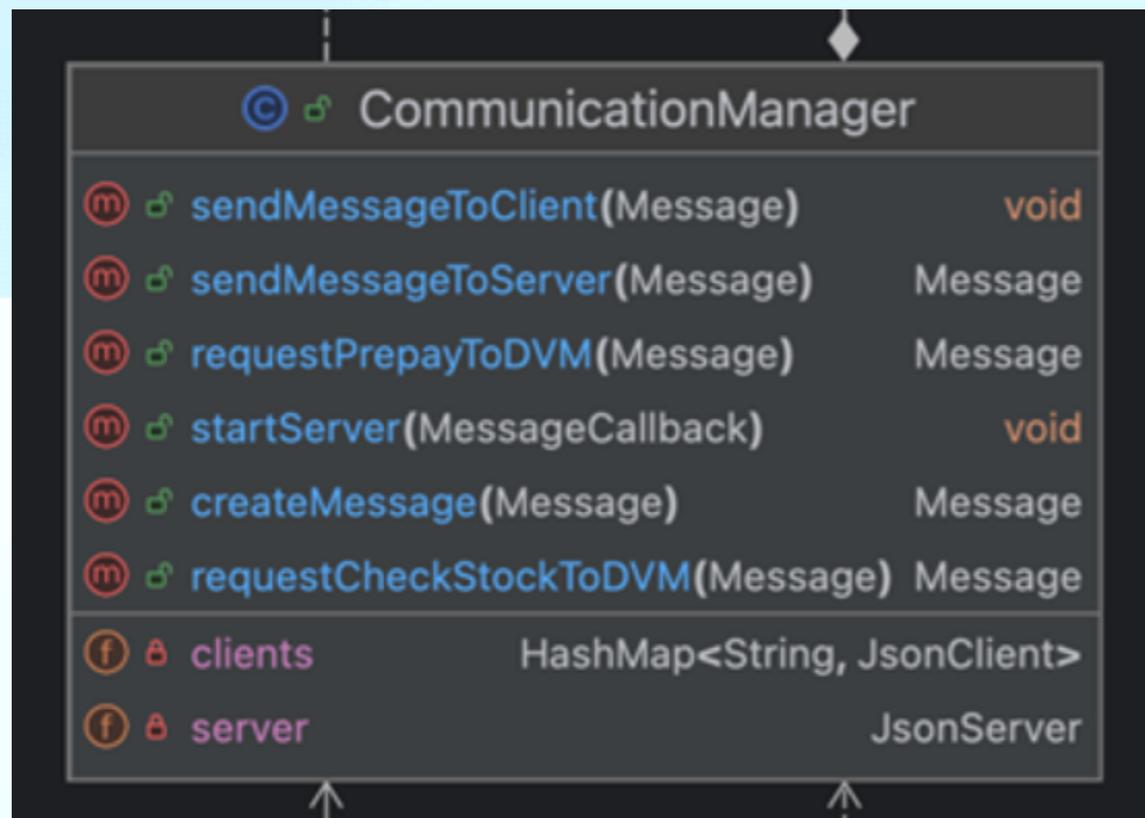
4. OOD (2040) 에서 변경/수정된 부분 정리

통신관련 at Stage 2040.



4. OOD (2040) 에서 변경/수정된 부분 정리

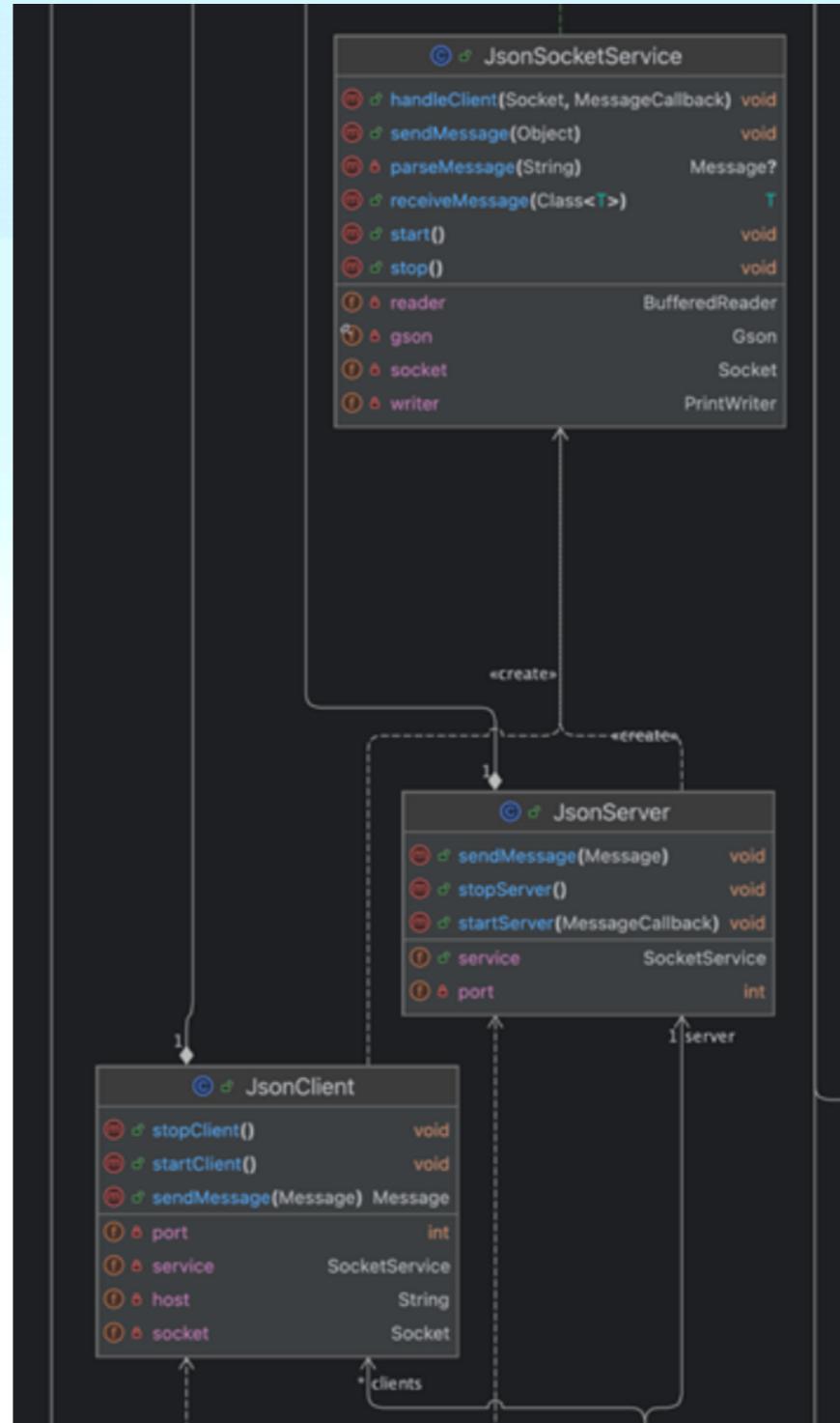
통신관련



- 제공된 SocketService, JsonSocketService, JsonServer 및 JsonClient 추가
- 서버에서 메시지를 받았을 경우 콜백을 위한 MessageCallback 인터페이스 클래스 추가

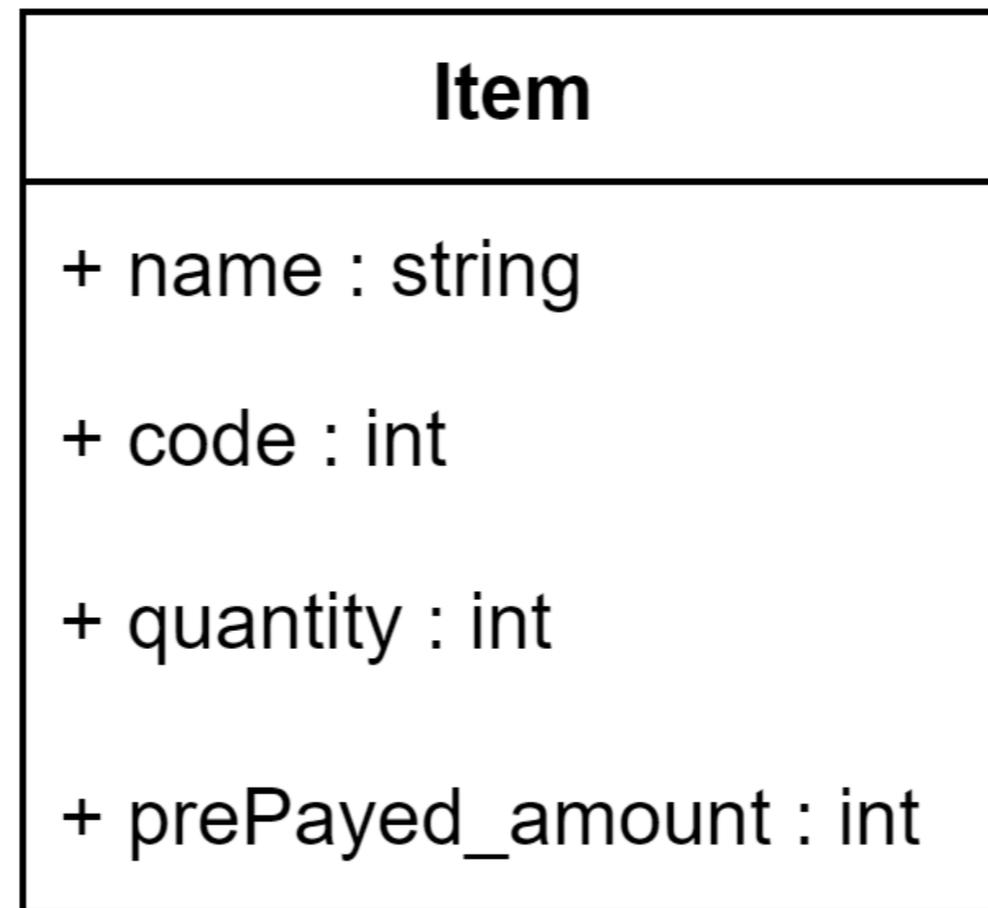
4. OOD (2040) 에서 변경/수정된 부분 정리

통신관련



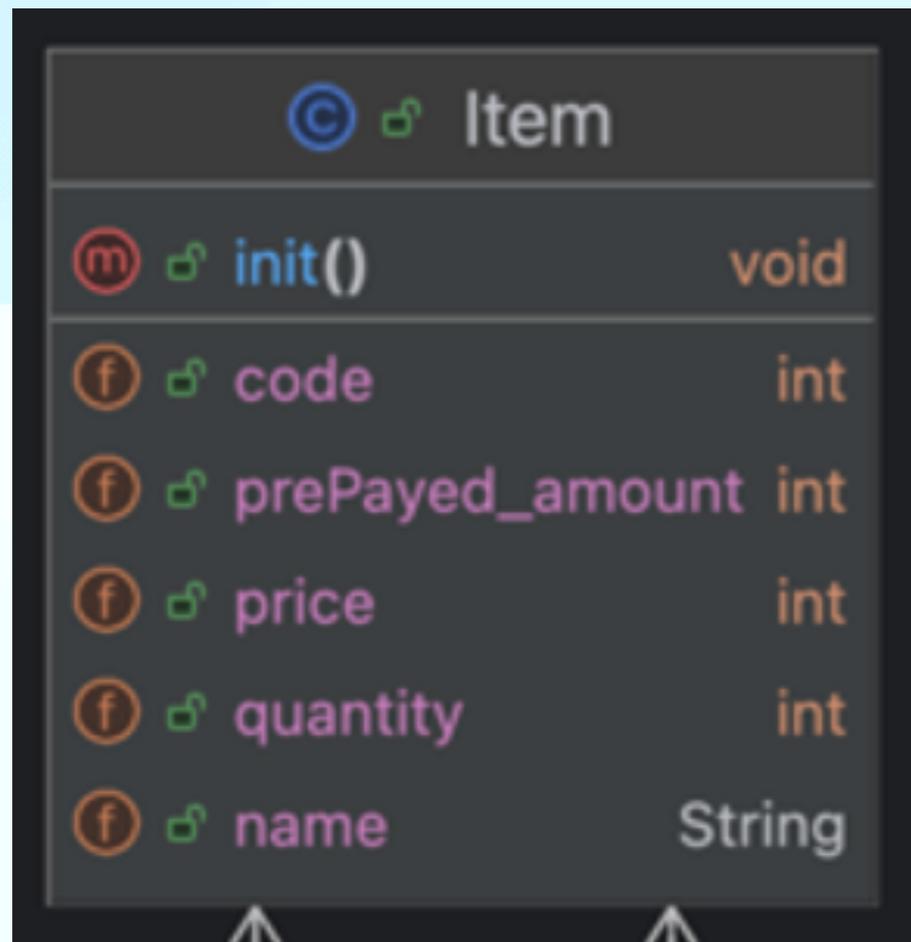
4. OOD (2040) 에서 변경/수정된 부분 정리

Item at Stage 2040.



4. OOD (2040) 에서 변경/수정된 부분 정리

Item



The screenshot shows the class definition for 'Item' in an IDE. The class has a constructor 'init()' and several attributes: 'code', 'prePayed_amount', 'price', 'quantity', and 'name'. The 'price' attribute is highlighted in pink, indicating it is a new addition. The return types for the attributes are 'int' for 'code', 'prePayed_amount', 'quantity', and 'String' for 'name'. The return type for 'price' is 'int'. There are two upward-pointing arrows at the bottom of the class definition, suggesting inheritance or a relationship with another class.

Symbol	Member	Type
Ⓜ	init()	void
Ⓣ	code	int
Ⓣ	prePayed_amount	int
Ⓣ	price	int
Ⓣ	quantity	int
Ⓣ	name	String

- 누락된 가격 정보(price) 추가

4. OOD (2040) 에서 변경/수정된 부분 정리

Stock at Stage 2040.

Stock
- items : Item[*]
+ checkStock(item_code : int, item_num : int) : bool + getItem(item_code: int): Item + updateStock(item_code : int, item_num : int, isPrepayed : bool, vCode : string) + itemList() : items

4. OOD (2040) 에서 변경/수정된 부분 정리

Stock

```
© Stock
(m) loadItemList() void
(m) getItems(int) Item
(m) checkStock(int, int) boolean
(m) updateStock(int, int, boolean, String) void
(m) itemList() ArrayList<Item>
(f) items ArrayList<Item>
```

- items의 형식을 단순 배열에서 ArrayList로 변경
- 파일로 저장된 재고를 불러오기 위해 loadItemList 메서드 추가

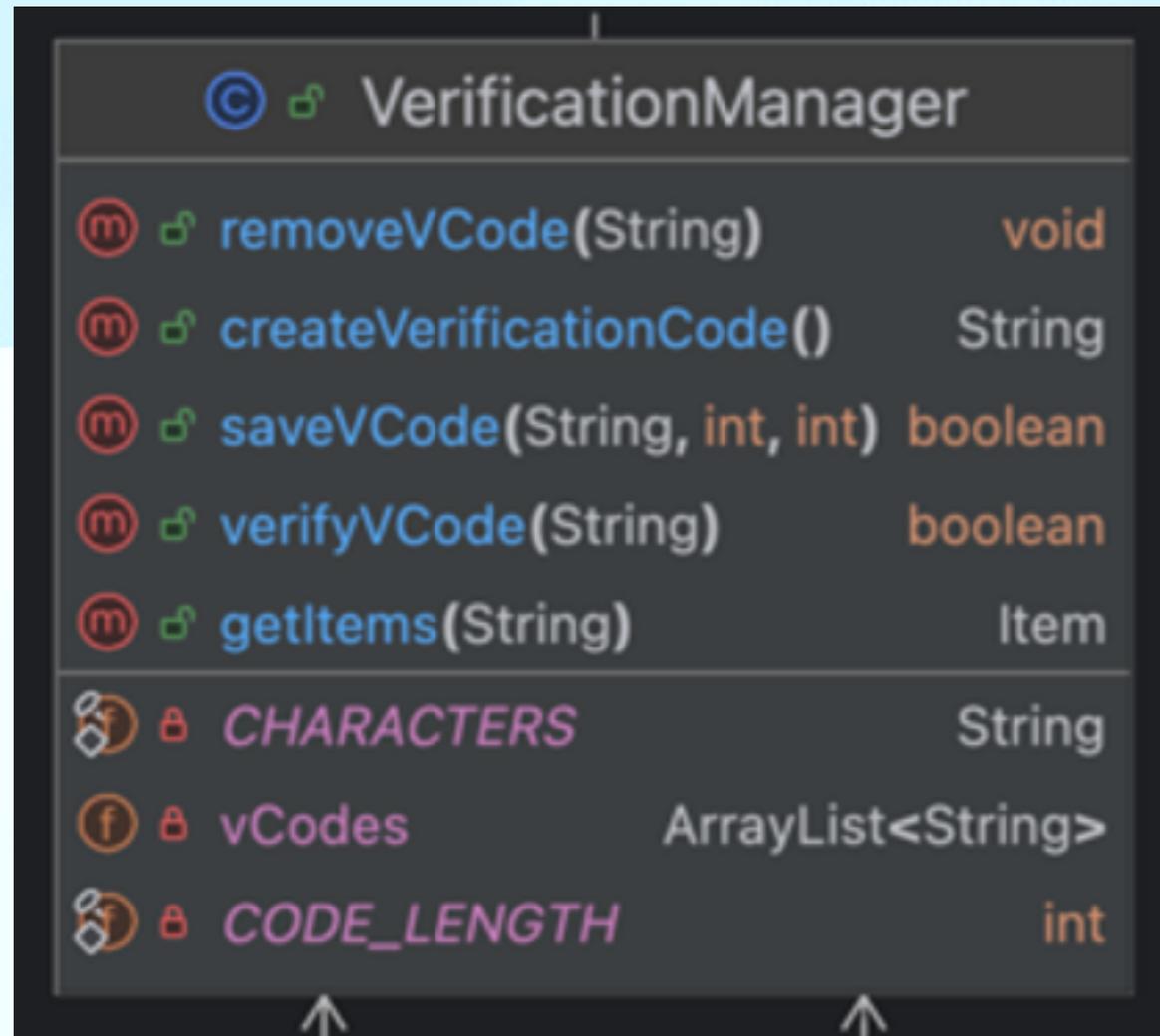
4. OOD (2040) 에서 변경/수정된 부분 정리

VerificationManager at Stage 2040.

VerificationManager
- vCode : List<string>
+ verifyVCode(vCode : string) : bool + saveVCode(vCode : string) : bool + createVerificationCode(void) : String

4. OOD (2040) 에서 변경/수정된 부분 정리

VerificationManager



```
VerificationManager  
  
removeVCode(String) void  
createVerificationCode() String  
saveVCode(String, int, int) boolean  
verifyVCode(String) boolean  
getItems(String) Item  
  
CHARACTERS String  
vCodes ArrayList<String>  
CODE_LENGTH int
```

- 현재 목록에서 인증코드를 제거하는 기능 추가
- saveVCode 매개변수 일부 수정
- 그외 랜덤한 인증 코드 생성을 위한 변수 추가

4. OOD (2040) 에서 변경/수정된 부분 정리

Bank at Stage 2040.

Bank
- cards : Card[*]
+ checkCardAvailability(cardInfo : Card, totalCost : int) : bool

4. OOD (2040) 에서 변경/수정된 부분 정리

Bank

```
© Bank
(m) checkCardAvailability(Card, float) boolean
(m) init() void
(f) cards ArrayList<Card>
```

- 초기화를 위한 init 추가

5. 구현 시 예상보다 어려웠던 점

- 자바와 Swing을 잘 사용하지 않았어서 초기 개발 환경을 구축하고 구현하는것이 어려웠다.
- OOD단계에서의 구상이 완벽하지 않았기에 구현하면서 설계 단계에서 생각보다 누락된 부분이 매우 많았으며 더 나은 방법이 분명 있는데, 왜 이걸 그때 생각하지 못했을까라는 생각을 계속하게 되었다.
- 팀원 모두 통신을 해본적이 없어서 처음에 통신하는 코드를 보고 어떻게 해야할지 잘 생각하지 못했다.

6. 구현 시 예상보다 쉬웠던 점

- 아무래도 기본적인 큰 틀이 존재하기에 구현의 초기단계가 수월하다
- CD로부터 자동 생성을 통해서 스켈레톤 코드를 만들고 SD를 보면서 하나씩 채워나가니 생각보다 금방 만들어낼 수 있었다.
- 물론 버그가 많이 있었지만 주어진 PFR을 정리한 문서가 있다보니 버그의 원인을 보다 쉽게 찾아낼 수 있었고 빠르게 수정할 수 있었다.

7. 객체지향개발방법론의 장단점 + 개인적인 소감들

객체지향개발방법론

- 소프트웨어 개발에서 객체지향 개념을 적용하여 소프트웨어를 설계하고 구현하는 방법론
- 복잡한 현실세계를 객체로 추상화하여 시스템을 개발하는 방법론

7. 객체지향개발방법론의 장단점 + 개인적인 소감들

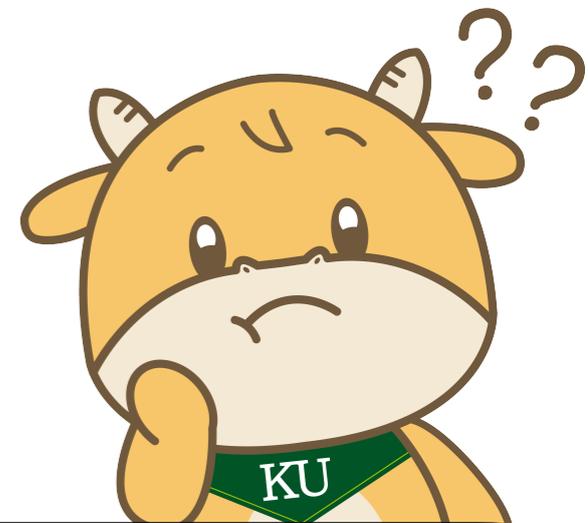
<이론적인 장점>

- 재사용성 (Reusability)
- 유지보수성 (Maintainability)
- 확장성 (Scalability)
- 직관성 (Intuitiveness)
- 캡슐화 (Encapsulation)
- 병렬 개발 (Parallel Development)

7. 객체지향개발방법론의 장단점 + 개인적인 소감들

<이론적인 장점>

- <대규모 소프트웨어 시스템>
- 재사용성 (Reusability)
- 유지보수성 (Maintainability)
- 확장성 (Scalability)
- 관점에서 좋을 수 있다는 걸 체감



But, DVM은 규모가 작은 시스템...

7. 객체지향개발방법론의 장단점 + 개인적인 소감들

<이론적인 장점>

- 병렬 개발 (Parallel Development)
- Package 별로 개발할 수 있어 해당 관점의 장점을 체감

자판기의 각 모듈(객체)들이 서로 상호작용하면서 우리가 생각하는 하나의 자판기를 이룬다고 생각!

7. 객체지향개발방법론의 장단점 + 개인적인 소감들

<이론적인 단점>

- 복잡성 (Complexity)
- 처음 배우기 어려움 (**Steep Learning Curve**)
- 성능 저하 (Performance Overhead)
- 과잉 설계 (Over-Engineering)
- 디버깅 어려움 (Debugging Difficulties)
- 자원 소모 (Resource Consumption)

7. 객체지향개발방법론의 장단점 + 개인적인 소감들

<이론적인 단점>

- 확실히 어렵다.



7. 객체지향개발방법론의 장단점 + 개인적인 소감들

개인적인 소감들

- OOAD로 처음 설계, 구현을 하다 보니, 정신없이 지나가서 아쉬움이 남음
- 여러 팀에서 개발해서 통신을 하에 메시지 포맷이 조금만 달라도 호환이 안되는 경우가 빈번
- 따라서 이러한 타협이 협업에 있어서 굉장히 중요한 요소임을 알게 됨
- 객체지향개발방법론에 대해서 배웠지만 아직 부족한 부분이 많아서 공부가 더 필요할 것 같음
- 교수님께서 말씀하셨던 것처럼 구현 하면서 습관적으로 절차지향적으로 생각하게 됨
- 하이브리드 형식의 코드를 만들어내는 듯 했고, 그러다 Maintenance Nightmare가 눈 앞에 아른 거리기도...
- 그럼에도 이번 학기 동안 객체지향에 대해 배우고 실제로 적용해보면서 많은 것을 알게 됨

7. 객체지향개발방법론의 장단점 + 개인적인 소감들

개인적인 소감들

- 만약 앞으로 다른 프로젝트를 진행하게 된다면 배운 내용을 꼭 이용해서 정말로 객체지향적인 개발을 해보고 싶다.

2024-1학기 객체지향개발방법론(3227-001)

감사합니다

모두 한학기 수고 많으셨습니다!

Team [T4] 202011282 김희준 202011286 남경식 202011377 지상준

