📖

Round 6 Quests

Calendar	Round 6
Tag	round6 task

📝 Round 6 Quests

💻 Implementation Quest

> 외부 시스템(PG) 장애 및 지연에 대응하는 Resilience 설계를 학습하고 적용해봅니다.
> `pg-simulator` 모듈을 활용하여 다양한 비동기 시스템과의 연동 및 실패 시나리오를 구현, 점검합니다.

🎯 **Must-Have (이번 주에 무조건 가져가야 좋을 것-**무조건 하세요**)**

- Fallback
- Timeout
- CircuitBreaker

Nice-To-Have (부가적으로 가져가면 좋을 것-시간이 허락하면 꼭 해보세요**)**

- Retryer

📦 추가 요구사항

```
### 결제 요청
POST {{pg-simulator}}/api/v1/payments
X-USER-ID: 135135
Content-Type: application/json

{
```

```
  "orderId": "1351039135",
  "cardType": "SAMSUNG",
  "cardNo": "1234-5678-9814-1451",
  "amount" : "5000",
  "callbackUrl": "http://localhost:8080/api/v1/examples/callback"
}

### 결제 정보 확인
GET {{pg-simulator}}/api/v1/payments/20250816:TR:9577c5
X-USER-ID: 135135

### 주문에 엮인 결제 정보 조회
GET {{pg-simulator}}/api/v1/payments?orderId=1351039135
X-USER-ID: 135135
```

- 결제 수단으로 PG 기반 카드 결제 기능을 추가합니다.
- PG 시스템은 로컬에서 실행가능한 `pg-simulator` 모듈이 제공됩니다. (별도 SpringBootApp)
- PG 시스템은 **비동기 결제** 기능을 제공합니다.

> *비동기 결제란, 요청과 실제 처리가 분리되어 있음을 의미합니다.*
> **요청 성공 확률 : 60%**
> **요청 지연** : 100ms ~ 500ms
> **처리 지연** : 1s ~ 5s
> **처리 결과**
> * 성공 : 70%
> * 한도 초과 : 20%
> * 잘못된 카드 : 10%

📋 과제 정보

- 외부 시스템에 대해 적절한 타임아웃 기준에 대해 고려해보고, 적용합니다.
- 외부 시스템의 응답 지연 및 실패에 대해서 대처할 방법에 대해 고민해 봅니다.
- PG 결제 결과를 적절하게 시스템과 연동하고 이를 기반으로 주문 상태를 안전하게 처리할 방법에 대해 고민해 봅니다.

- 서킷브레이커를 통해 외부 시스템의 지연, 실패에 대해 대응하여 서비스 전체가 무너지지 않도록 보호합니다.

✅ Checklist

⚡ PG 연동 대응

- [ ] PG 연동 API는 RestTemplate 혹은 FeignClient 로 외부 시스템을 호출한다.
- [ ] 응답 지연에 대해 타임아웃을 설정하고, 실패 시 적절한 예외 처리 로직을 구현한다.
- [ ] 결제 요청에 대한 실패 응답에 대해 적절한 시스템 연동을 진행한다.
- [ ] 콜백 방식 + **결제 상태 확인 API**를 활용해 적절하게 시스템과 결제정보를 연동한다.

🛡️ Resilience 설계

- [ ] 서킷 브레이커 혹은 재시도 정책을 적용하여 장애 확산을 방지한다.
- [ ] 외부 시스템 장애 시에도 내부 시스템은 **정상적으로 응답**하도록 보호한다.
- [ ] 콜백이 오지 않더라도, 일정 주기 혹은 수동 API 호출로 상태를 복구할 수 있다.
- [ ] PG 에 대한 요청이 타임아웃에 의해 실패되더라도 해당 결제건에 대한 정보를 확인하여 정상적으로 시스템에 반영한다.

✍️ (Optional) Technical Writing Quest

> 이번 주에 학습한 내용, 과제 진행을 되돌아보며
> **"내가 어떤 판단을 하고 왜 그렇게 구현했는지"** 를 글로 정리해봅니다.
>
> **좋은 블로그 글은 내가 겪은 문제를, 타인도 공감할 수 있게 정리한 글입니다.**
>
> 이 글은 단순 과제가 아니라, **향후 이직에 도움이 될 수 있는 포트폴리오** 가 될 수 있어요.

▼ 📚 Technical Writing Guide

✅ 작성 기준

항목	설명
형식	블로그
길이	제한 없음, 단 꼭 **1줄 요약 (TL;DR)** 을 포함해 주세요
포인트	“무엇을 했다” 보다 **“왜 그렇게 판단했는가”** 중심

예시 포함	코드 비교, 흐름도, 리팩토링 전후 예시 등 자유롭게
톤	실력은 보이지만, 자만하지 않고, **고민이 읽히는 글**예: “처음엔 mock으로 충분하다고 생각했지만, 나중에 fake로 교체하게 된 이유는...”

✨ 좋은 톤은 이런 느낌이에요

> 내가 겪은 실전적 고민을 다른 개발자도 공감할 수 있게 풀어내자

특징	예시
🤔 내 언어로 설명한 개념	Stub과 Mock의 차이를 이번 주문 테스트에서 처음 실감했다
💭 판단 흐름이 드러나는 글	처음엔 도메인을 나누지 않았는데, 테스트가 어려워지며 분리했다
📐 정보 나열보다 인사이트 중심	테스트는 작성했지만, 구조는 만족스럽지 않다. 다음엔...

❌ 피해야 할 스타일

예시	이유
많이 부족했고, 반성합니다...	회고가 아니라 일기처럼 보입니다
Stub은 응답을 지정하고...	내 생각이 아닌 요약문처럼 보입니다
테스트가 진리다	너무 단정적이거나 오만해 보입니다

🎯 Feature Suggestions

- PG 응답이 느려서 서킷브레이커가 열렸다...?
- 응답이 안 와서 실패 처리했는데, PG에선 결제가 됐다고...?
- 주문 상태는 Pending 인데, 사용자는 결제 안내를 받았다.
- PG 장애 하나로 주문 전체가 멈춰버렸다.
- 결제가 실패하면 주문을 무조건 롤백해야 할까?
- 재시도 횟수는 몇 번이 적절했을까?
- 폴백 처리를 어떻게 했지?