# 객체지향분석및설계

# HW 4. SystemImplementation

소프트웨어융합학부 20203025 공채연

## 1. 프로젝트 주제 (Vision and Scope)

- **프로젝트 명**

  laundry 24

- **프로젝트 내용**

  사용자가 빨래를 가져와서 세탁기에 넣고 코스를 선택한다.
  문이 잠김을 확인하고, 코스 선택화면을 띄운다.
  결제 전 세탁 취소는 Lock만 풀고 main화면 띄운다.

  코스에 맞는 금액이 제시되고 사용자는 결제를 진행한다(카드). 결제가 정상적으로 이루어지면 Timer를 실행시키고, 세탁을 시작한다. 세탁을 시작했으므로 세탁기의 문은 잠그고, Timer가 끝나면 잠금을 해제한다.
  세탁 중간에 문을 열 수 있는 Stop 버튼이 있다. Stop 버튼은 투입한 금액을 초기화 시키고 세탁을 강제 종료한다. 추가 기능(세탁, 헹굼, 탈수 횟수 추가)을 선택할 때는 추가 1개당 금액을 부여한다.

  관리자: 총 매출 확인

  주택을 제외하고 원룸, 공동주택 등에서 밤에 빨래가 힘든 상황을 완화한다.
  부피가 큰 이불 빨래 등 집에서 세탁하기에 불편한 세탁물을 대형 세탁기를 사용하여 편하게 빨래할 수 있는 편리함을 제공한다.

- **프로젝트의 범위**

  자취 비율이 높은 대학생을 주 Target으로 함
  주 고객층이 시설을 이용하면서 불편했던 사항과 선호하는 방식을 조사·분석한다.
  ex) 코인으로 결제-> 카드결제)
  세탁 시작, 결제, 추가 기능 처리, 세탁 완료까지 설계 및 구현한다.

  [추가기능]
  1. 세탁 중간에 문을 열 수 있는 Stop
  2. 세탁, 헹굼, 탈수 횟수 추가

## 2. 기능적인 요구사항 (Use Case Description)

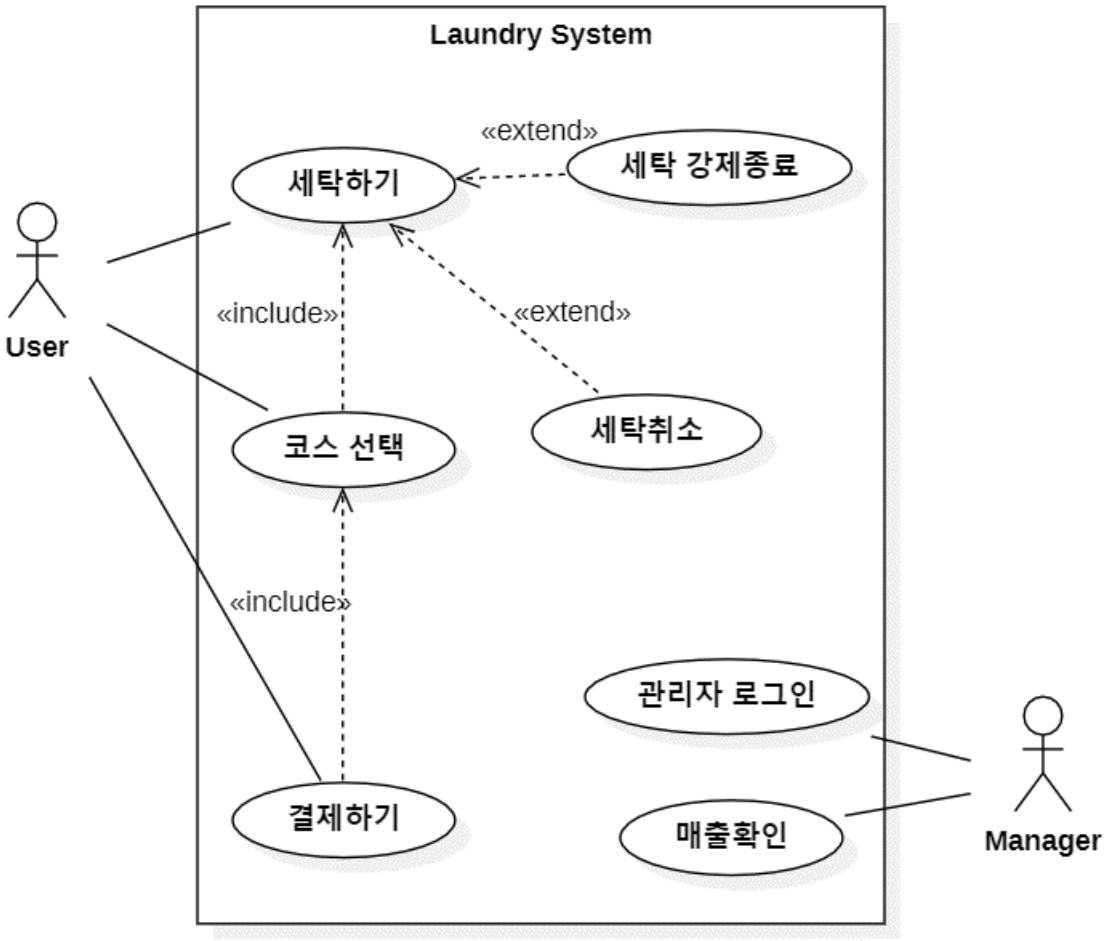
### 1. User

| | | |
|---|---|---|
| Use case name | 빨래 요청 | |
| Scenario | 고객을 위한 새로운 빨래 시작 | |
| Triggering event | 고객이 빨래를 하기위해 세탁물을 가지고 옵니다. | |
| Brief description | 고객이 빨래를 세탁기에 넣습니다. 세탁기 Lock을 확인 후 코스를 선택하고 코스별 금액을 결제합니다. 세탁을 진행 및 추가기능도 수행합니다. (추가기능 위에 명시) 세탁이 완료되면 Lock을 해제합니다. | |
| Actors | 자취생, 인근 주민 등 | |
| Related use cases | 여러 매장의 키오스크 | |
| Stakeholders | laundry 24 System | |
| Preconditions | 빨래할 세탁물과 결제할 카드가 필요합니다. | |
| Post conditions | 결제가 완료되면 빨래가 시작됩니다.<br>세탁이 진행되는 시간을 대기해야 합니다. | |
| Flow of activities | Actor | System |
| | [User]<br>1. 고객이 세탁물을 넣고 문 닫음<br><br>2. 코스 선택<br><br>3. 추가기능 선택<br><br>4. 결제 전 세탁 취소<br><br>5. 결제<br><br>6. 세탁 진행중 정지 | 1-1 다양한 세탁 코스를 띄움<br><br>2-1 추가 기능(세탁, 헹굼, 탈수 횟수 추가) 띄움<br><br>3-1 선택 코스의 결제금액을 띄움<br>3-2 결제 요청<br><br>4-1 Lock 해제<br>4-2 Main 화면 띄움<br><br>5-1 결제 완료 화면 띄움<br>5-2 세탁 시작<br><br>6-1 투입한 금액을 초기화<br>6-2 세탁을 강제 종료<br>6-3 lock 해제 |
| Exception conditions | 3-2 카드 결제가 정상적으로 이루어지지 않으면 Main화면으로 돌아 감. 결제 실패 시 lock 해제 | |

## 2. Manager

| | | |
|---|---|---|
| Use case name | 관리자 매출확인 | |
| Scenario | 빨래방의 총 매출 확인 | |
| Triggering event | 관리자임을 확인하기 위해 로그인을 합니다 | |
| Brief description | ID: Admin / PW:1234 을 입력합니다.<br>매출을 확인합니다. | |
| Actors | 관리자 | |
| Related use cases | 여러 매장의 키오스크의 관리자 모드 | |
| Stakeholders | laundry 24 System (관리자모드) | |
| Preconditions | 관리자의 ID/PW가 필요합니다. | |
| Post conditions | 매출을 수정하지 못합니다. | |
| Flow of activities | Actor | System |
| | 1. 매출확인 | 1-1 매출 list 띄움 |
| Exception conditions | 관리자모드 <-> 유저모드 모드 변경시에 프로그램을 재실행 해야합니다. | |

## 3. Use Case Diagram




## 4. Non-Functional Requirements

| NFR 내용 | 고려하는 Quality (혹은 quality의 sub characteristic) | Quality Attributes | 비고 |
|---|---|---|---|
| 단계 이동에서 반응 속도를 적절하게 처리 | Performance<br>- Time Behavior | 다음 단계를 진행하는 과정에서 Turnaround Time이 평균 2초 이하가 되도록 함. | |
| 실패 가능성을 낮추어프로그램의 신뢰성을 높임 | Reliability<br>- Maturity | 결제 기능을 실행 시 실패할 가능성이 95% 이하가 되도록 함. | |
| 다양한 의류로 코스 선택이 어려운 경우 편리하도록 처리 | Usability<br>- Appropriateness Recognizability | 코스를 적절하고 쉽게 선택할 수 있도록 코스별 추천 의류 3개 이상 나타냄. | |

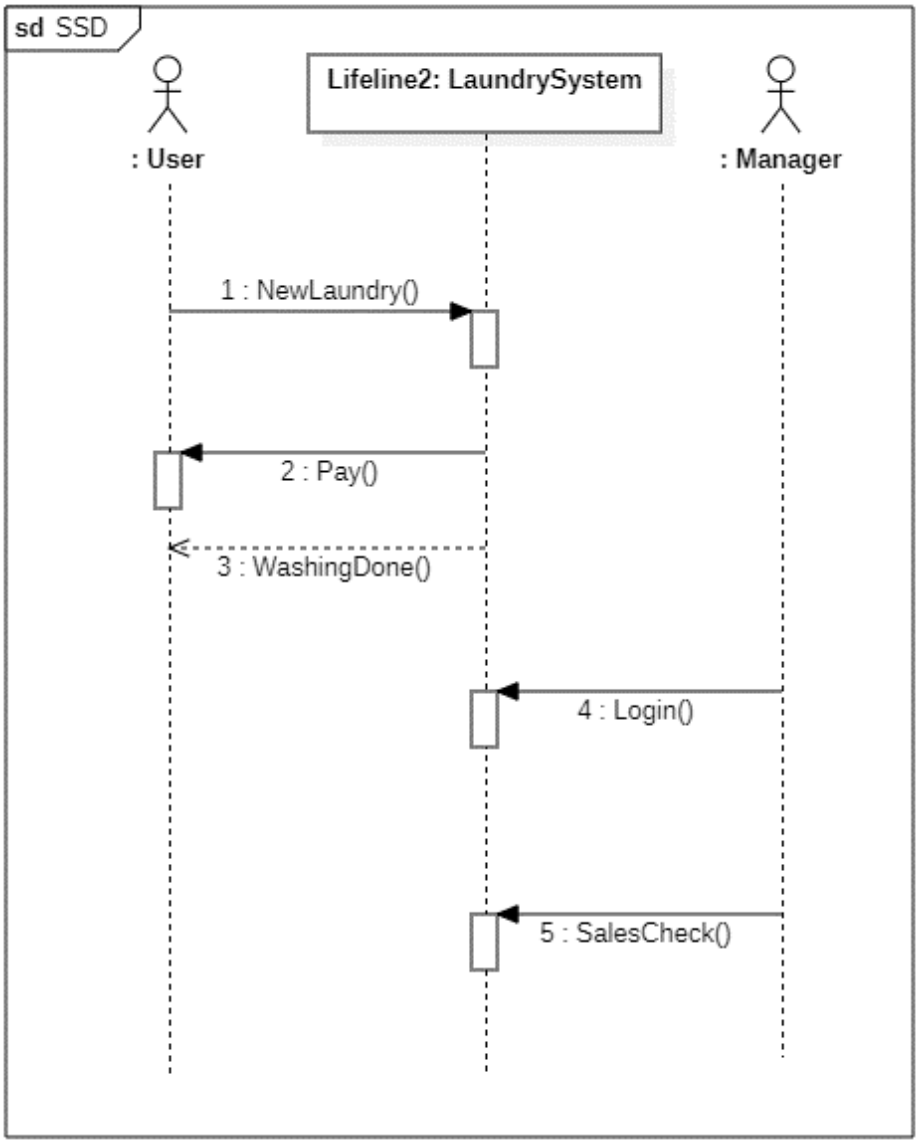
## 5. DOMAIN MODEL 설계하기

| Class 명 | Concept 설명 | 주요 attributes |
|---|---|---|
| NewLaundry | 세탁 코스를 선택 및 추가, 선택한 세탁 코스를 진행하는 controller | |
| FirstWindow | 세탁 진행 중 강제 종료 혹은 세탁 취소, 세탁 완료를 관리하는 controller | |
| Pay | 결제 처리 controller | |
| Login | 관리자를 확인하고 로그인 하는 controller | |
| SalesCheck | 매출을 확인하는 controller | 관리자 |

## 6. DESIGN MODEL 설계하기

- 주요한 Use Case 에 대하여 Interaction Diagram 설계하기

1) System Sequence Diagram 그리기



2) Operation Contract 만들기

| Name | NewLaundry |
|---|---|
| 설명 | 세탁을 시작한다. |
| Pre-conditions | 실행중인 세탁물이 없어야 한다. |
| Post-conditions | 세탁을 진행하기 위한 코스들을 선택한다. |

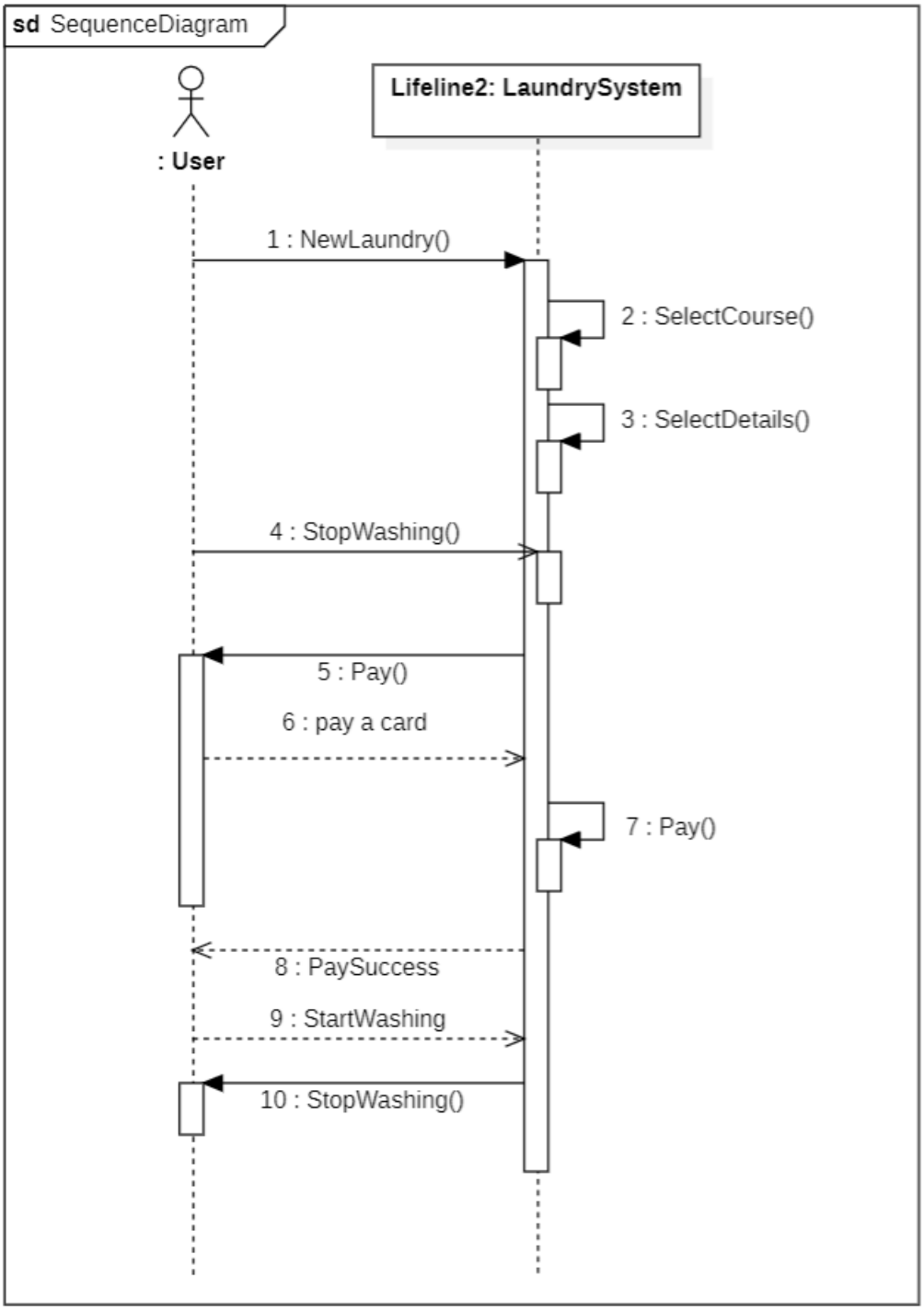
| Name | StopWashing |
|---|---|
| 설명 | 세탁을 중지한다. |
| Pre-conditions | Stop 버튼을 클릭 해야한다. |
| Post-conditions | 세탁 진행 전후를 판단해야 한다. |

| Name | Pay |
|---|---|
| 설명 | 코스별로 계산된 금액을 계산 및 결제한다. |
| Pre-conditions | 세탁물의 금액 계산이 완료되어야 한다. |
| Post-conditions | 결제 금액의 일치를 판단해야한다. |

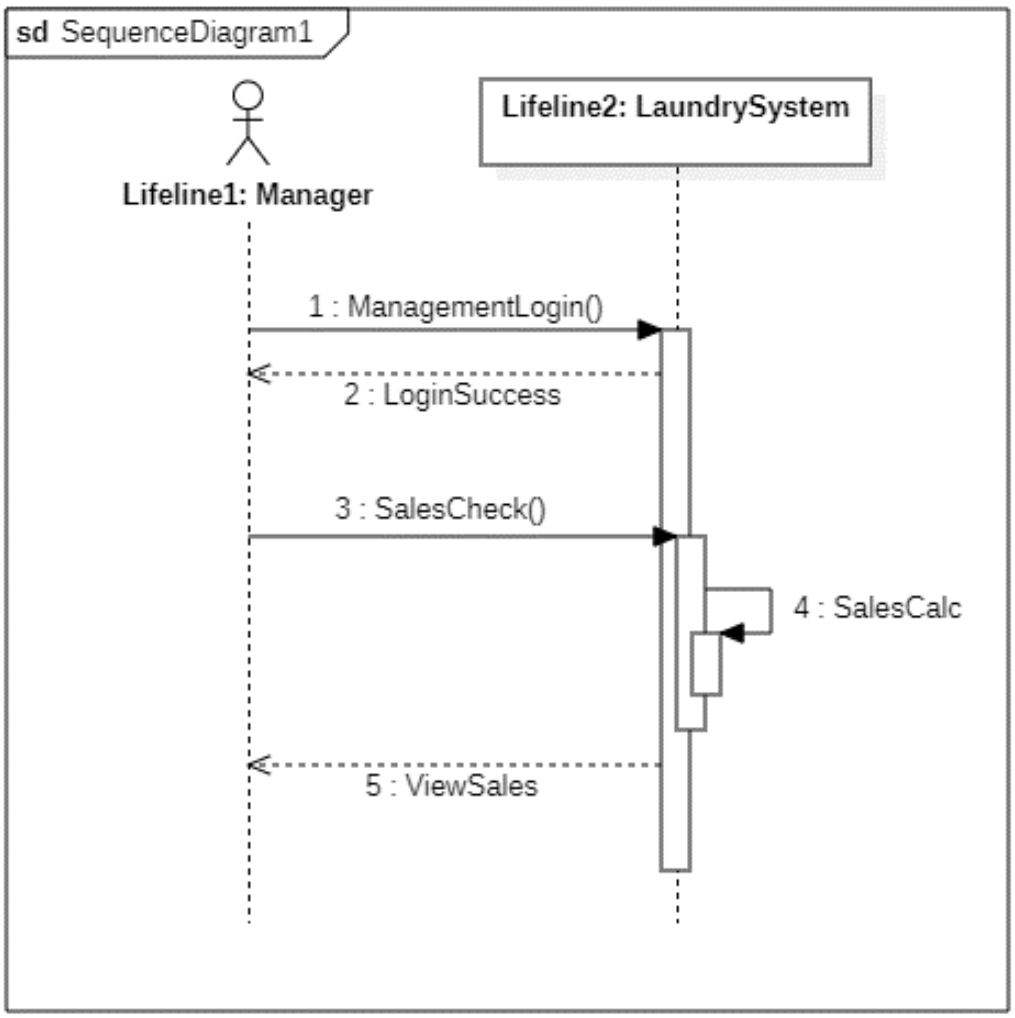
| Name | Login |
|---|---|
| 설명 | 관리자임을 확인하기 위한 절차이다. |
| Pre-conditions | 진행중인 세탁물이 없고 관리자 로그인 버튼을 클릭해야 한다 |
| Post-conditions | 관리자의 ID/PW 가 일치해야 한다. |

| Name | SalesCheck |
|---|---|
| 설명 | 관리자가 매출을 확인한다. |
| Pre-conditions | 관리자가 로그인 되었음을 확인해야 한다. |
| Post-conditions | 매출 금액을 계산하고 표시해야 한다. |

3) Sequence Diagram 설계하기 (User)

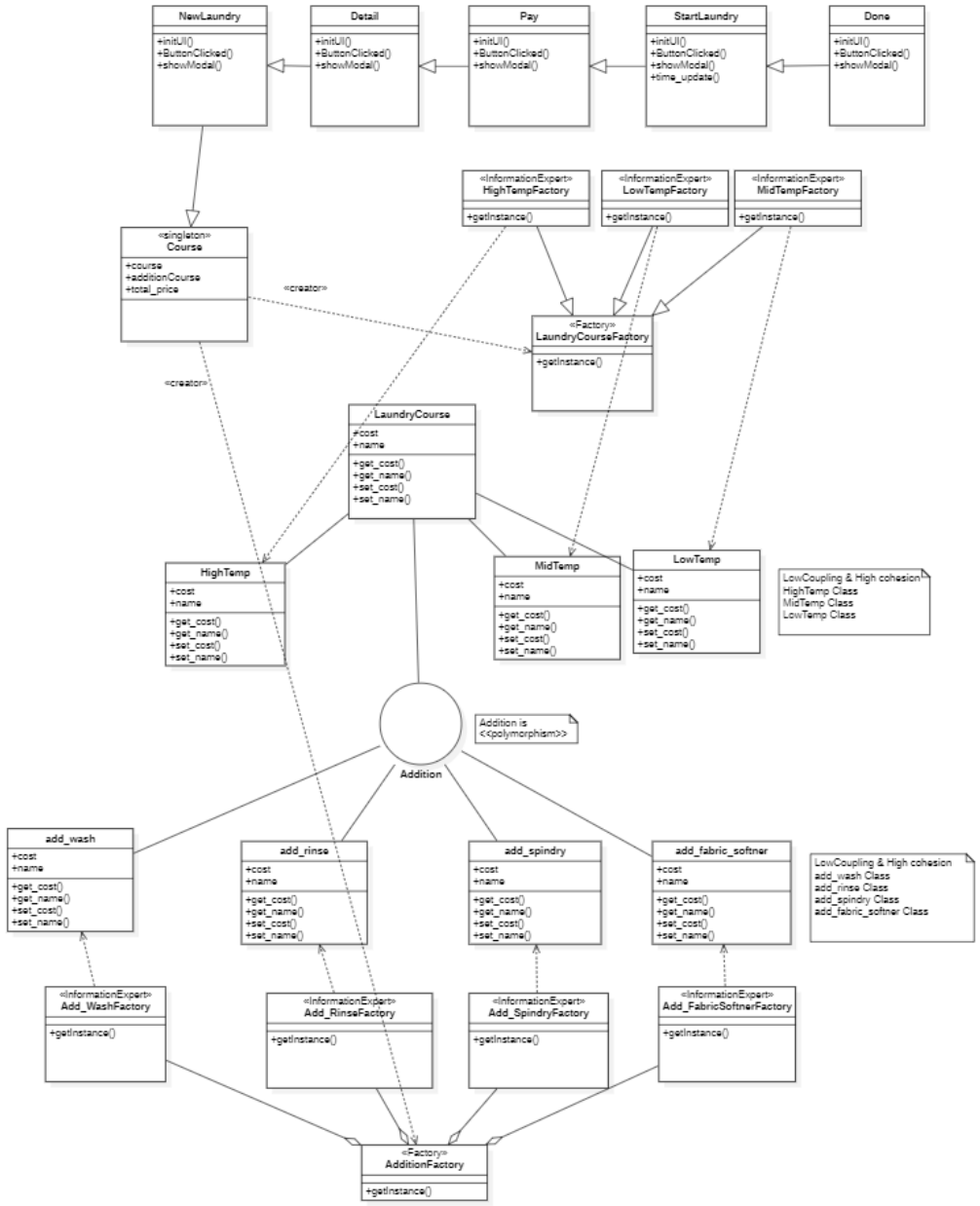


3) Sequence Diagram 설계하기 (Manager)



- 구현단계 Class Diagram

  ■ 주요한 Use Case 들의 interaction diagram 들에서 나타난 모든 class 들을 반영하여 class diagram 을 완성하세요



- Design Refinement 표

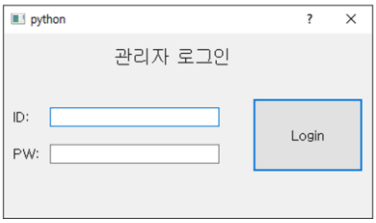
| Before 단계의 Class 등 설계 classifier | After 단계의 Class 등 설계 classifier(s) | 적용한 설계 개념 (GRASP, Design Pattern 적용) | Architecture Design Rationale (본인이 생각하는 합리성) | NFR 와 QA 에 대한 영향 분석 |
|---|---|---|---|---|
| Course | Class | **Design Patterns**<br>- Singleton | 처음부터 끝까지 C(course 객체) 하나가 세탁을 진행하기 때문에 Singleton 을 사용했다.<br>만약 객체가 바뀌려면 재실행을 해야 하기 때문에 프로그램이 껐다 켜져야 한다. 따라서 껐다 켜지면 C 도 리셋이 되기 때문에 하나의 객체가 끝까지 전역적으로 유지된다. 즉 singleton 을 만족한다. | 객체의 유일성을 보장하기 때문에 실패 가능성을 낮추어 프로그램의 신뢰성을 높여 Reliability-Maturity 를 만족한다. 따라서 실패할 가능성을 99% 이하로 갈 수 있도록 한다. |
| Course | Class | **GRASP**<br>- Creator | 선택한 코스에 맞는 객체를 생성하고, 코스에서 Detail 인 add_wash, add_rinse 등 다양한 객체를 생성하는 creator 로 지정한다. | |
| Addition Factory, Laundry Course Factory | Classes | **Design Patterns**<br>- Factory Pattern | Factory pattern 을 적용시켜 상황에 맞는 객체를 생성하는 Factory 를 생성했다. | 현재는 Synchronous 한 프로그램이라 반환 시간을 줄인다는 효과가 미미하게 나타난다. 하지만 Asynchronous 인 경우, multi Thread 를 사용하는 등 보다 효율적으로 반환시간을 줄일 수 있다. |
| Addition Factory, Laundry Course Factory | Classes | **GRASP**<br>-Information Expert | Factory 를 구성하기 위한 객체의 정보들을 가지고 있다. 전문적인 지식을 가지고 있다고 생각하여 information expert 라고 생각한다. | |
| HighTemp Class, MidTemp Class, LowTemp Class, add_wash Class add_rinse Class add_spindry Class add_fabric_softner Class | Classes | **GRASP**<br>- Low Coupling | 기존의 함수형태의 method 로 구현했던 부분의 coupling 을 낮추기 위해 detail 부분을 객체로 변경하였다. 따라서 집착성을 낮추는 효과를 가져왔다. | |
| HighTemp Class, MidTemp Class, LowTemp Class, add_wash Class add_rinse Class add_spindry Class add_fabric_softner Class | Classes | **GRASP**<br>- High cohesion | Factory Pattern 을 적용하여 복잡도를 줄이고 전문성을 높였다. 자연스레 High cohesion 이 적용되어, 객체별로 각자 할 일을 명확하게 했다. | |
| Addition | Interface | **GRASP**<br>-polymorphism | LaundryCourse 를 상속받아 필요한 method 들을 미리 정의해두고, 연결된 add_wash 등 각각의 객체에서 method 로 다른 service 를 제공할 수 있도록 했다. 따라서 다형성을 제공한다. | |

- **결과 Snapshot**

Main화면

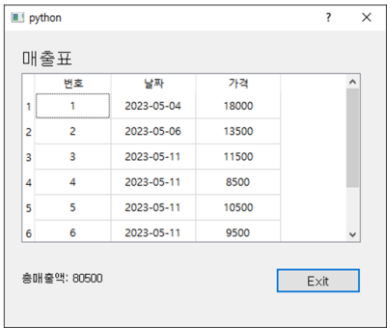

■ Main 화면에서 관리자를 선택하게 된 경우

관리자 로그인




관리자 로그인을 진행한다.

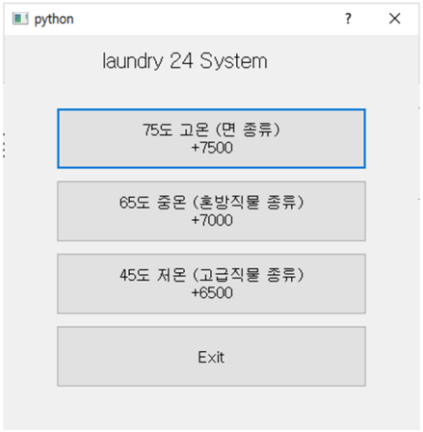
매출확인




관리자가 확인될 시, 매출확인 창으로 넘어가며, 매출을 확인할 수 있게 된다.

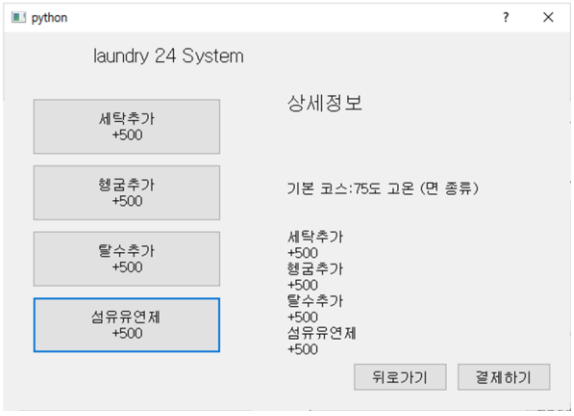
■ Main 화면에서 고객을 선택하게 된 경우

코스 선택




코스를 선택할 수 있는 코스 선택창이 뜬다.

상세 세탁 선택



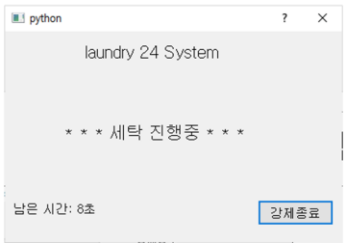

코스를 선택하게 되면 기본 코스: 에 방금 선택한 코스가 입력된다.

이어서 세부 사항들을 선택할 수 있다.

결제하기 버튼을 클릭하게 되면 결제창이 뜬다.

결제창


세탁 진행




세탁 완료


결제를 하기 위한 금액이 나타난다.

결제 후 세탁을 10초동안 진행한다.

10초가 지나거나 강제종료를 클릭했을 시 세탁이 종료되며 프로그램이 마무리된다.

**설계에 관한 설명**

전반적인 설계는 Factory pattern으로 설계하였다. Client가 '빨래 진행해줘!'라고 요청 시에 Factory가 공장처럼 가동하여 빨래를 척척 진행해준다. Client의 선택을 기반으로 적절한 object를 생성해주는 것이다. 따라서 외부에서는 세탁이 진행되는 과정의 logic을 판단할 필요가 없다.

Factory Pattern을 적용하면서 자연스레 Factory내부의 작은 Factory는 Information Expert가 적용되었다. 기능별로 Factory를 나누어 두었기 때문에 또한 복잡한 Laundry logic에 Low Coupling, High Cohesion을 적용할 수 있었다.

초반에는 주요내용을 Method 기반 설계로 진행하였지만 피드백을 참고하여 역할을 할당할 Class로 구성하는 프로젝트를 운영하기 위해 GRASP, Design Pattern을 적용했다. 그 결과 Class기반 프로젝트를 완성하였다.