OOAD

HW4 : SSD & Interaction Diagram, GRASP 설계하기

20181608 소프트웨어학과 문원기

0. 설계하고자 하는 시스템

인수인계 및 근무 시간 기록 시스템을 설계하고자 합니다.

vision

전체적인 목표는 24시간 마트에서 일하는 직원들을 대상으로, 인수인계 내용과 근무 시간을 기록하는 시스템입니다.
이 프로그램을 통해서 직원들은 인수인계 내용을 정확하게 전달받을 수 있으며, 고용주 입장에서는 직원마다 근무 시간을 정확히 기록되기 때문에 월급 계산이 용이해집니다.
또한 직원마다 근무 시간을 비교함으로써 업무 분배가 공평하게 이루어지는지를 판단하는데 도움이 됩니다.

프로그램을 설계하게 된 배경

저에게는 가족 경영으로 24시간 마트를 운영하는 친구가 있습니다. 해당 친구는 인수인계 내용이 정확히 전달되지 않아 물건을 발주하거나 상품을 진열하는 과정에서 문제가 생긴 적이 많다고 저에게 자주 칭얼대곤 합니다.
게다가 누군가 항상 카운터를 지켜야 하는 24시간 마트 특성상 서로의 불가피한 사정으로 인해 근무 시간을 조정하는 경우가 잦다고 합니다. 이런 상황 속에서 근무 시간을 메모지 하나에 의존하여 기록하다 보니 월급 계산을 할 때 애로사항이 많다고 이야기하곤 했습니다.
이러한 친구의 수요를 기반으로 해당 가게에서 사용할 수 있는 인수인계 및 근무 시간 기록 시스템을 만들고자 합니다.

인수인계 관련 항목의 Scope

인수인계 내용을 적을 수 있는 게시판 형태의 목록 창이 있습니다.
버튼을 눌러 인수인계 내용을 추가할 수 있으며, 삭제 또한 가능합니다.

근무 시간 관련 항목의 Scope

근무 시간을 기록하는 항목이 있습니다. 근무자 목록에는 직원들의 명단이 있습니다.
근무자 목록 옆에는 근무자 출근 버튼, 근무자 퇴근 버튼, 근무자 관리 버튼이 있습니다.

- 근무자 목록에서 한 명을 선택한 뒤에 근무 시작 버튼을 누르면 해당 직원의 근무 시작 시간을 기록합니다.

- 근무 중인 직원 중 한 명을 선택한 후 근무 종료 버튼을 누르면 이 버튼을 클릭하면 해당 직원의 근무 시간 기록을 중단합니다.

- 근무자 관리 버튼을 누르면 근무자 관리 창이 뜹니다.

근무자 관리 창에서는 근무자 추가, 삭제, 근무 시간 확인, 근무 시간 초기화 버튼과 비밀번호 입력 창이 있습니다. 근무 시간 확인을 제외하고 모두 비밀번호를 입력해야 동작하며, 비밀번호를 입력하지 않으면 경고창이 화면에 나옵니다.

- 근무자 추가 버튼을 누르면 근무자 목록에 새로운 직원을 추가할 수 있습니다.

- 근무자 삭제 버튼을 누르면 근무자 목록에 있는 직원을 삭제할 수 있습니다.

- 직원을 클릭한 후에 근무 시간 확인 버튼을 누르면, 직원의 총 근무 시간을 알 수 있습니다.

- 직원을 클릭한 후에 초기화 버튼을 누르면 해당 직원의 근무 시간이 0시간으로 초기화됩니다.

- '근무 시간 설정' 칸에 바꾸고 싶은 시간으로 입력한 후에, 근무 시간 초기화를 누르면 해당 직원의 근무 시간이 입력한 값으로 바뀌게 됩니다.

비기능적 요구사항

포스 프로그램이 돌아가는 컴퓨터에서 구동될 예정이기 때문에 기본적으로 가벼워야 합니다.
그렇기 때문에 용량은 1GB를 넘지 않는 선에서 만들 생각입니다.

프로그램이 꺼졌다가 켜져도 마지막으로 저장한 상태를 불러올 수 있도록 file을 만들어서 저장해야 합니다.

UI 재배치로 인해 근무자 관리 창과 관련된 기능들이 약간씩 수정되었습니다.

1. use case 도출하기

use case name	출근	
Scenario	직원들의 출근 시간을 기록	
Triggering event	직원이 출근하여 자신의 출근 시간을 기록하고자 함	
Brief description	근무지에 도착한 직원이 근무자 목록에서 자신의 이름을 선택한 뒤 근무 시작 버튼을 누른다. 시스템이 근무 시작 시간을 기록한다.	
Actors	직원	
Related use cases	None	
Stakeholders	직원, 사장, 회계 담당자	
Preconditions	직원 목록에 자신의 이름이 이미 존재해 있어야 한다. (만약 없다면 '새로운 직원 추가하기' use case를 진행한다.) 해당 직원이 이미 출근 처리가 되어있는 상태여서는 안 된다.	
Post conditions	직원 목록에서 해당 직원의 상태가 '근무중'으로 바뀐다. 직원이 출근한 시간이 기록된다.	
Flow of activities	Actor	System
	1. 직원이 가게에 출근한다.	
	2. 직원이 직원 목록을 확인한다.	2.1 시스템이 최신화된 직원 목록창을 띄운다 2.2 시스템이 해당 직원이 근무 중인지에 대한 정보를 화면에 출력한다. 2.3 시스템이 해당 직원이 지금까지 근무한 시간을 화면에 출력한다.
	3. 직원이 직원 목록 창에서 자신의 이름을 클릭한다.	3.1 시스템이 해당 직원을 리스트에서 찾는다.
	4. 직원이 '출근' 버튼을 누른다.	4.1 시스템이 직원의 출근 시간을 기록한다. 4.2 시스템이 직원의 상태를 '근무중'으로 바꾼다.
Exception Conditioins	3.1 만약 해당 직원이 직원 목록 리스트에 존재하지 않는다면 사장에게 상황을 전달하여 "새로운 직원 추가하기" use case를 시작한다.	

use case name	퇴근	
Scenario	직원의 퇴근 시간을 기록	
Triggering event	근무를 마친 직원이 퇴근 시간을 기록하고자 함.	
Brief description	근무중이던 직원이 퇴근하고자 함. 근무자 명단에서 자신의 이름을 누른 뒤 근무 종료 버튼을 누름. 시스템이 근무 종료 시간을 기록하고 총 일한 시간을 기록하여 저장한다.	
Actors	직원	
Related use cases	None	
Stakeholders	직원, 사장, 회계 담당자	
Preconditions	퇴근 처리를 하려는 직원의 이름이 근무자 명단에 있어야 하며, '근무중' 상태여야 한다.	
Post conditions	해당 직원이 '근무중' 상태가 아니게 되며, 시스템이 오늘 일한 시간을 기록하여 해당 직원의 총 근무 시간에 더한다.	
Flow of activities	Actor	System
	1. 직원이 퇴근 시간을 기록하기 위해 '근무자 명단'에서 자신의 이름을 찾는다.	1.1 시스템이 최신화된 직원 목록창을 띄운다
	2. '근무자 명단'에서 해당 직원의 이름을 클릭한다.	
	3. 직원이 '퇴근' 버튼을 누른다.	3.1 시스템이 해당 명령에 대한 precondition을 확인한다. 3.2 시스템이 기록되어 있는 출근 시간과 현재 시간을 통해 일한 시간을 계산한다. 3.3 시스템이 계산된 시간을 해당 직원의 총 근무 시간에 더해준다. 3.4 시스템이 해당 직원의 상태를 '근무중'에서 해제한다.
Exception Conditioins	3.1 만약 '근무중' 상태가 아닌 직원을 대상으로 '퇴근' 버튼을 누른다면 시스템이 다시 아무런 작업도 하지 않고 초기 화면으로 돌아온다.	

use case name	새로운 직원 고용	
Scenario	새로운 직원을 명단에 추가하기	
Triggering event	새로운 직원이 고용되어 자신의 이름을 근무자 명단에 추가하고자 함.	
Brief description	사장이 새롭게 일하게 된 근무자를 근무자 명단에 추가하기 위해 '근무자 관리' 버튼을 누름. 시스템에 해당 근무자의 이름을 입력한 뒤 '추가' 버튼을 누르면 근무자 명단에 새로운 직원이 추가됨.	
Actors	사장	
Related use cases	Includes '신원 확인하기' use case	
Stakeholders	직원, 사장, 회계 담당자	
Preconditions	동일한 이름을 가진 직원을 추가할 경우, 이름 뒤에 (숫자)를 적는 등 구분자를 추가해주어야 한다.	
Post conditions	직원 목록에서 해당 직원이 추가된다.	
Flow of activities	Actor	System
	1. 사장이 '근무자 관리' 버튼을 누른다.	1.1 시스템이 '근무자 관리 창'을 띄운다.
	2. 새로 고용한 직원의 이름을 입력한다.	
	3. 비밀번호를 비밀번호 입력 칸에 입력한다.	
	4. 근무자 추가 버튼을 클릭한다.	4.1 '신원 확인하기' use case를 시작함. 해당 use case에 나온 동작이 완료되면 다음 동작으로 이어짐. 4.2 시스템이 근무자 명단에서 해당 직원의 entry를 추가한다.
Exception Conditioins		

use case name	해고 및 퇴직	
Scenario	직원이 일을 그만둠	
Triggering event	직원이 일을 그만두면서 직원 명단을 수정해야 함.	
Brief description	근무자 명단에 이름이 올라와 있는 직원이 일을 그만둠. 근무자 명단에서 해당 직원의 이름을 클릭한 뒤, '근무자 삭제' 버튼을 누름. 시스템이 리스트에서 해당 직원을 삭제한다.	
Actors	사장	
Related use cases	Includes '신원 확인하기' use case	
Stakeholders	직원, 사장, 회계 담당자	
Preconditions	해고 및 퇴직하려는 직원의 이름이 근무자 명단에 있어야 한다.	
Post conditions	해당 직원의 이름이 근무자 명단에서 사라진다.	
Flow of activities	Actor	System
	1. 사장이 '근무자 관리' 버튼을 누른다.	1.1 시스템이 '근무자 관리 창'을 띄운다.
	2. 삭제할 해당 직원의 이름을 클릭한다.	
	3. 비밀번호를 비밀번호 입력 칸에 입력한다.	
	4. 근무자 삭제 버튼을 클릭한다.	4.1 '신원 확인하기' use case를 시작함. 해당 use case에 나온 동작이 완료되면 다음 동작으로 이어짐. 4.2 시스템이 근무자 명단에서 해당 직원의 entry를 삭제한다.
Exception Conditioins		

use case name	인수인계 추가	
Scenario	직원이 인수인계를 작성.	
Triggering event	직원이 인수인계를 남기고자 함.	
Brief description	직원이 다음 근무자에게 전달하고자 하는 인수인계가 생김. 인수인계 내용을 입력한 뒤 '추가' 버튼을 누름. 시스템이 해당 인수인계를 인수인계 게시판에 저장함.	
Actors	직원	
Related use cases	None	
Stakeholders	직원, 사장, 회계 담당자	
Preconditions	None.	
Post conditions	해당 인수인계의 내용이 인수인계 게시판에 추가됨.	
Flow of activities	Actor	System
	1. 사장 또는 직원이 인수인계를 남기고자 인수인계 작성 칸에 인수인계 내용을 작성한다.	
	2. '추가' 버튼을 누름.	2.1. 시스템이 해당 인수인계를 인수인계 리스트에 저장함. 2.2 최신화된 인수인계 리스트를 인수인계 게시판에 출력함.
Exception Conditioins		

use case name	인수인계 삭제	
Scenario	직원이 인수인계를 삭제함	
Triggering event	직원이 인수인계를 삭제하고자 함.	
Brief description	직원이 더 이상 계속 남길 필요가 없다고 생각되는 인수인계를 지우고자 함. 인수인계를 클릭한 후, 인수인계 삭제 버튼을 누름. 시스템이 해당 인수인계를 인수인계 목록에서 지움.	
Actors	직원	
Related use cases	None	
Stakeholders	직원, 사장, 회계 담당자	
Preconditions	None.	
Post conditions	해당 인수인계의 내용이 인수인계 게시판에서 삭제됨.	
Flow of activities	Actor	System
	1. 직원이 인수인계를 목록에서 클릭함.	
	2. 인수인계 삭제 버튼을 누름.	2.1 시스템이 인수인계를 인수인계 목록에서 삭제함. 2.2 최신화된 인수인계 리스트를 인수인계 게시판에 출력함.
Exception Conditioins		

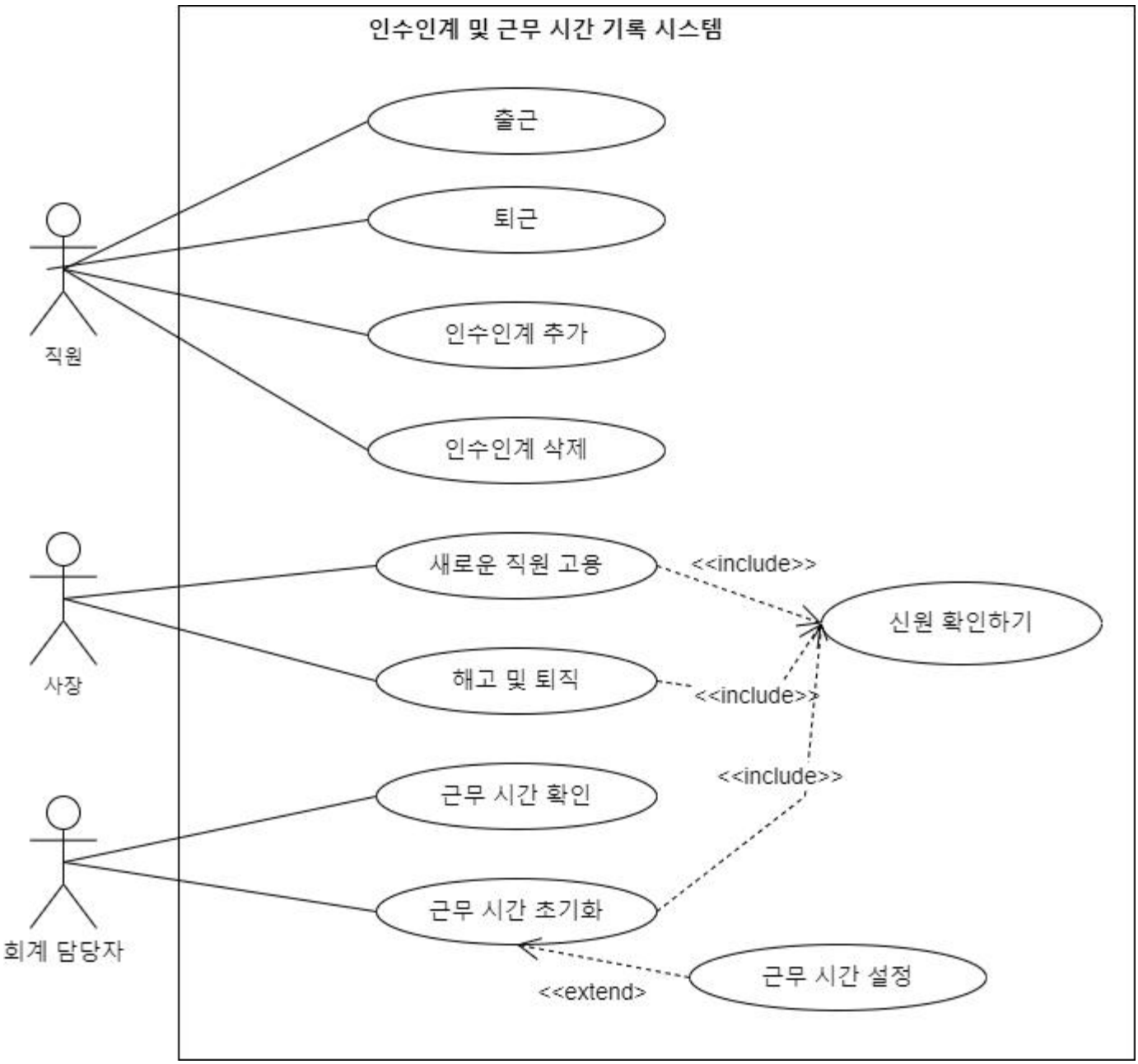
use case name	신원 확인하기	
Scenario	특정 명령을 입력한 사람이 허가받은 인원인지 확인함.	
Triggering event	허가 받은 인원만 가능한 특정 명령이 입력됨.	
Brief description	사장 또는 회계 담당자만이 가능한 '근무 시간 초기화' 또는 '새로운 직원 고용' 또는 '해고 및 퇴직' use case가 실행될 때, 입력한 인원이 허가받은 인원인지 확인하기 위해 비밀번호를 확인받음.	
Actors	사장, 회계 담당자	
Related use cases	Included by '근무 시간 초기화' & '새로운 직원 고용' & '해고 및 퇴직'	
Stakeholders	직원, 사장, 회계 담당자	
Preconditions	None.	
Post conditions	None.	
Flow of activities	Actor	System
	1. 사장 또는 회계 담당자가 비밀번호가 필요한 특정 작업을 하기 위해 "근무자 관리 창"에 있는 특정 버튼을 클릭한다.	1.1 시스템이 비밀번호가 맞는지 확인한다. 1.2 비밀번호가 맞다면 원래 해당 작업을 계속 진행한다.
Exception Conditioins	1.1 만약 비밀번호가 틀리다면 경고창을 띄우고, 기존 작업을 중지한다.	

<table>
<tr><td>use case name</td><td colspan="2">근무 시간 초기화</td></tr>
<tr><td>Scenario</td><td colspan="2">회계 담당자가 근무 시간을 초기화함.</td></tr>
<tr><td>Triggering event</td><td colspan="2">회계 담당자가 직원의 근무 시간을 초기화하고자 함.</td></tr>
<tr><td>Brief description</td><td colspan="2">회계 담당자가 다양한 사유(대체로 월급 계산 후 새롭게 근무 시간을 기록해야 하는 경우)로 직원의 근무 시간을 초기화하고자 함.</td></tr>
<tr><td>Actors</td><td colspan="2">회계 담당자</td></tr>
<tr><td>Related use cases</td><td colspan="2">includes ‘신원 확인하기’
extended by‘근무 시간 설정’</td></tr>
<tr><td>Stakeholders</td><td colspan="2">직원, 사장, 회계 담당자</td></tr>
<tr><td>Preconditions</td><td colspan="2">None.</td></tr>
<tr><td>Post conditions</td><td colspan="2">특정 직원의 근무 시간이 초기화됨.</td></tr>
<tr><td rowspan="5">Flow of activities</td><td>Actor</td><td>System</td></tr>
<tr><td>1. 회계 담당자가 근무 시간 초기화를 위해 ‘근무자 관리 버튼’을 클릭한다.</td><td>1.1 시스템이 ‘근무자 관리 창’을 띄운다.</td></tr>
<tr><td>2. 근무 시간을 초기화시킬 직원을 클릭한다.</td><td></td></tr>
<tr><td>3. 비밀번호를 입력한다.</td><td></td></tr>
<tr><td>4. 근무 시간 설정’ 버튼을 누름.</td><td>4.1 ‘신원 확인하기’ use case를 시작함. 해당 use case에 나온 동작이 완료되면 다음 동작으로 이어짐.
4.2 해당 직원의 근무 시간을 0으로 초기화한다.</td></tr>
<tr><td>Exception Conditioins</td><td colspan="2">2. 만약 0시간 0분이 아닌 다른 초기값을 설정하고 싶다면, ‘근무 시간 설정’ use case로 넘어간다.</td></tr>
</table>

use case name	근무 시간 설정	
Scenario	회계 담당자가 특정 직원의 근무 시간을 새로 설정함.	
Triggering event	회계 담당자가 직원 또는 직원들의 근무 시간을 새로 설정하고자 함.	
Brief description	회계 담당자가 다양한 사유에 의해 근무 시간을 0이 아닌 특정한 값으로 설정하고자 함. 설정할 값과 직원을 입력 받은 후 시스템이 해당 직원의 근무 시간을 입력받은 값으로 설정함.	
Actors	회계 담당자	
Related use cases	extension of '근무 시간 초기화'	
Stakeholders	직원, 사장, 회계 담당자	
Preconditions	None.	
Post conditions	해당 직원의 근무 시간이 설정값으로 변함.	
Flow of activities	Actor	System
	1. 회계 담당자가 근무 시간 설정 칸에 설정하고자 하는 초기값을 입력한다.	
	2. 비밀번호를 입력한다.	
	3. '근무 시간 설정' 버튼을 누름.	3.1 '신원 확인하기' use case를 시작함. 해당 use case에 나온 동작이 완료되면 다음 동작으로 이어짐. 3.2 해당하는 직원의 근무 시간을 설정된 값으로 초기화한다.
Exception Conditioins		

근무 시간 확인은 굳이 use case description을 작성하지 않았습니다.

2. use case diagram



3. System Sequence Diagram

시스템 시퀀스 다이어그램은 '출근' use case와 '새로운 직원 고용' use case 대상으로 만들었습니다.

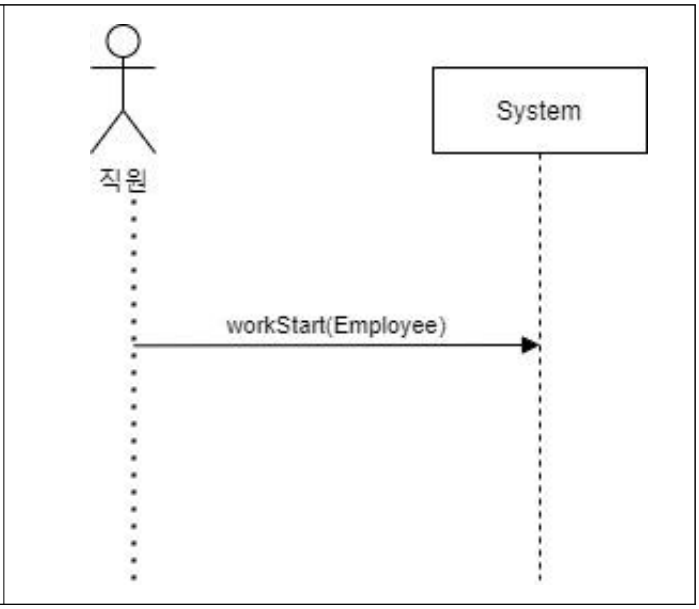
왼쪽은 use case에 대한 간략한 설명을 실었고, 오른쪽은 System sequence Diagram을 그려넣었습니다.

Use Case : '출근'

1. 이 use case는 직원이 가게에 도착하여, 자신의 출근을 기록하고자 할 때 시작합니다.

2. 직원이 자신의 이름을 직원 목록 (근무자 목록)에서 찾은 후에 클릭하여 선택합니다. 그 후에 '출근' 버튼을 눌러서 자신의 출근을 기록합니다.

3. 시스템은 직원이 시스템을 켰을 때, 최신화된 직원 목록을 로딩해 보여줍니다. 직원이 선택된 후에 '출근' 버튼이 눌리면, 내부적으로 그것을 반영한 뒤에 화면에 최신화된 직원 목록을 보여줍니다.

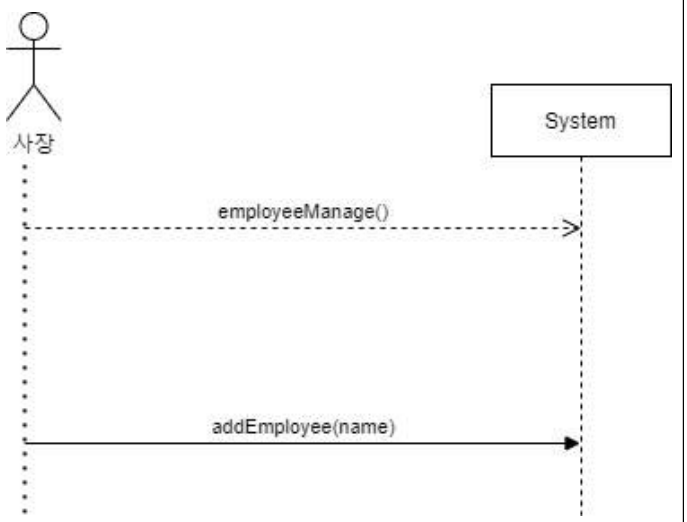


Use Case : '새로운 직원 고용'

1. 이 use case는 사장이 새로운 직원을 목록에 추가하고자 할 때 시작합니다.

2. 사장이 처음에 '직원 관리' 버튼을 누릅니다. 새로 뜨는 창에 패스워드와 직원의 이름을 입력하고, '직원 추가' 버튼을 누릅니다.

3. 시스템은 사장이 시스템을 켰을 때, 최신화된 직원 목록을 로딩해 보여줍니다. '직원 관리' 버튼이 눌리면 '직원 관리 창'을 띄웁니다. 해당 창에서 패스워드와 직원의 이름을 입력받은 뒤, '직원 추가' 버튼이 눌리면 내부적으로 그것을 반영한 뒤에 화면에 최신화된 직원 목록을 보여줍니다.



4. Operation Contracts

'출근' use case와 관련된 system sequence diagram에 대한 operation contract를 작성해 보았습니다.

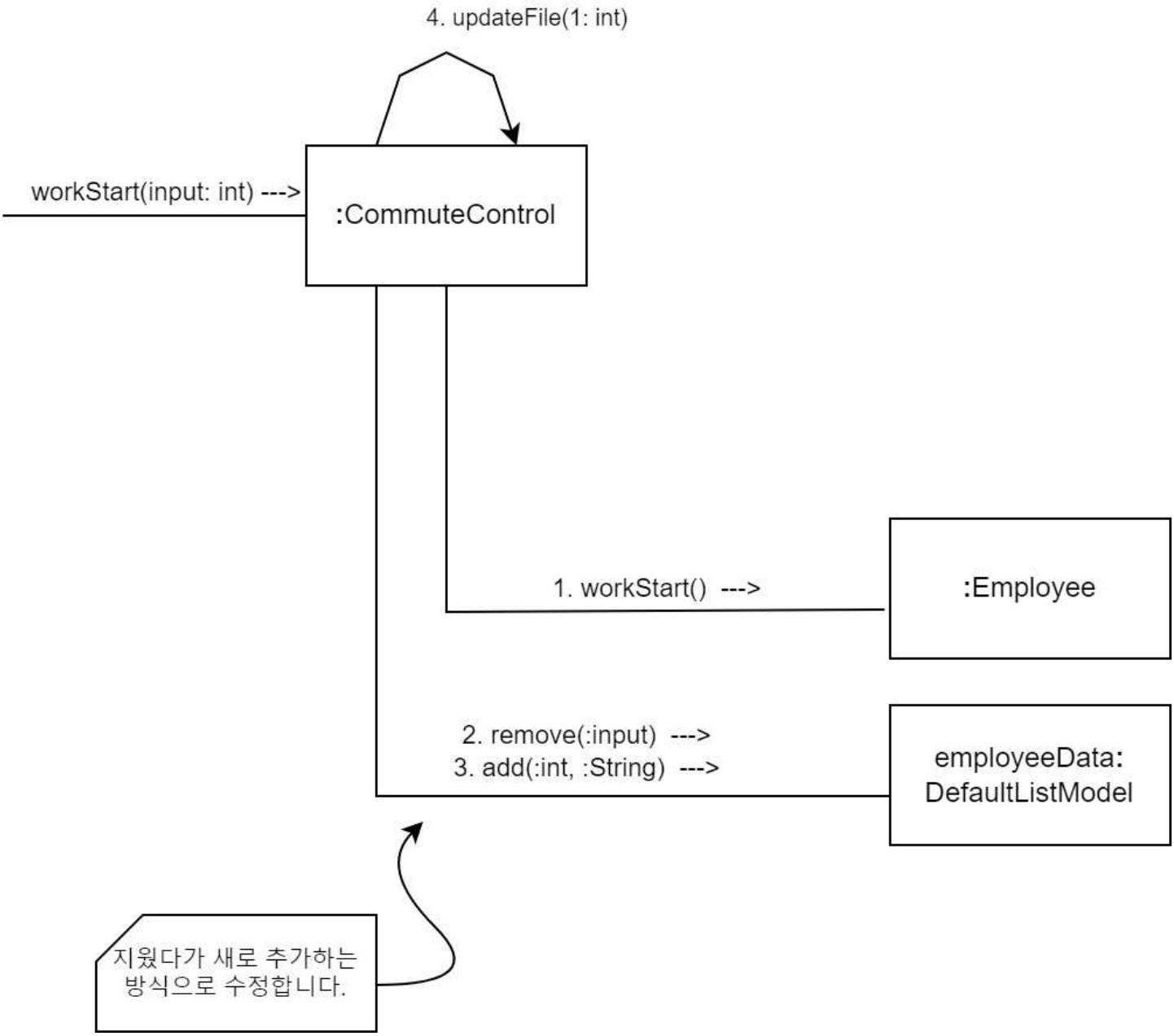
Name	**workStart(input : int)**
Responsibility	선택된 직원의 상태를 '출근중'으로 변환한 후, 출근 시간을 기록한다. 그렇게 변환된 직원 목록을 파일로 저장한다.
Type	void
Cross References	Use Case : 출근
Exception	직원이 이미 '출근중' 상태라면 출근 시간만 바꾼다.
Output	최신화된 employeeData.dat 파일, 최신화된 Employee 객체와 DefaultListModel 객체.
Pre-Condition	직원 목록에서 해당 직원을 클릭한 뒤, '출근' 버튼을 누른다.
Post-Condition	해당 직원의 정보가 담긴 Employee 객체의 상태가 '출근중' 상태로 수정된다. 또한 해당 Employee 객체가 출근 버튼이 눌린 시간을 저장한다. 직원 목록을 표시하기 위해 사용되는 DefaultListModel 객체도 해당 변경 사항을 반영한다. 이로 인해 화면에 출력되는 직원 목록 또한 최신화된다. 직원들의 정보를 저장한 "employeeData.dat" 파일을 최신화한다.
설명	출근 버튼이 클릭된다면 선택된 직원의 인덱스를 parameter로 하여 workStart 메소드가 실행이 됩니다. GUI를 구성하는 mainScreen 객체에서 버튼 클릭을 감지했다면 action call을 할 것입니다. commuteControl 객체에서는 해당 action을 listen했을 때, 내부적으로 workStart() 메소드를 호출하여 실질적인 동작을 수행합니다.

'새로운 직원 고용' use case와 관련된 system sequence diagram에 대한 operation contract를 작성해 보았습니다.

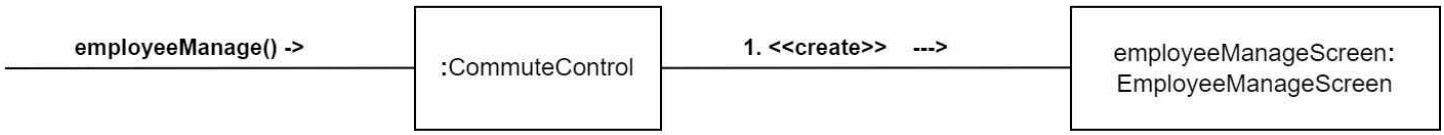
Name	**employeeManage()**
Responsibility	직원 관리 창을 띄운다.
Type	void
Cross References	Use Case : 새로운 직원 고용, 해고 및 퇴직, 근무 시간 초기화, 근무 시간 확인
Exception	None
Output	EmployeeManageScreen 객체
Pre-Condition	'직원 관리' 버튼이 클릭된다.
Post-Condition	EmployeeManageScreen 객체를 생성하여 직원 관리 창을 띄운다.

Name	**addEmployee(input: String)**
Responsibility	새로운 직원에 해당하는 Employee 객체를 만든다. 새로운 직원을 직원 목록에 추가한다. 최신화된 직원 목록 데이터를 employeeData.dat 파일로 저장한다.
Type	void
Cross References	Use Case : 새로운 직원 고용
Exception	None
Output	최신화된 employeeData.dat 파일, 최신화된 Employee 객체와 DefaultListModel 객체.
Pre-Condition	직원 관리 창에서 직원 이름 입력 칸에 직원 이름을 입력하고 비밀번호 입력 칸에 비밀번호를 입력한 뒤, '직원 추가' 버튼을 누른다. 비밀번호 확인을 위한 내부 메소드인 checkPassword()가 1을 리턴한다면, addEmployee() 메소드가 호출된다.
Post-Condition	해당 직원의 정보가 담긴 Employee 객체를 새로 생성한다. 직원 목록을 표시하기 위해 사용되는 DefaultListModel 객체에 해당 변경 사항을 반영한다. 이로 인해 화면에 출력되는 직원 목록 또한 최신화된다. 직원들의 정보를 저장한 "employeeData.dat" 파일을 최신화한다.

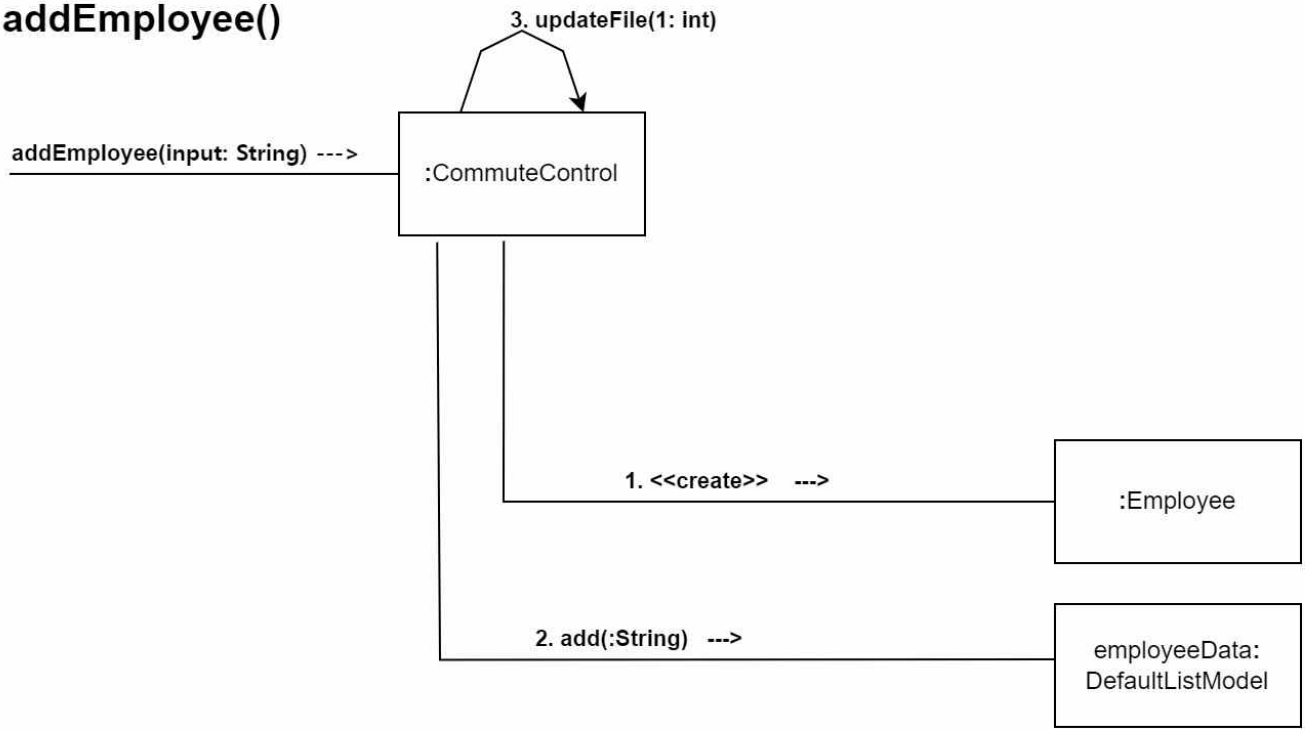
5. Interaction Diagram



EmployeeManage()

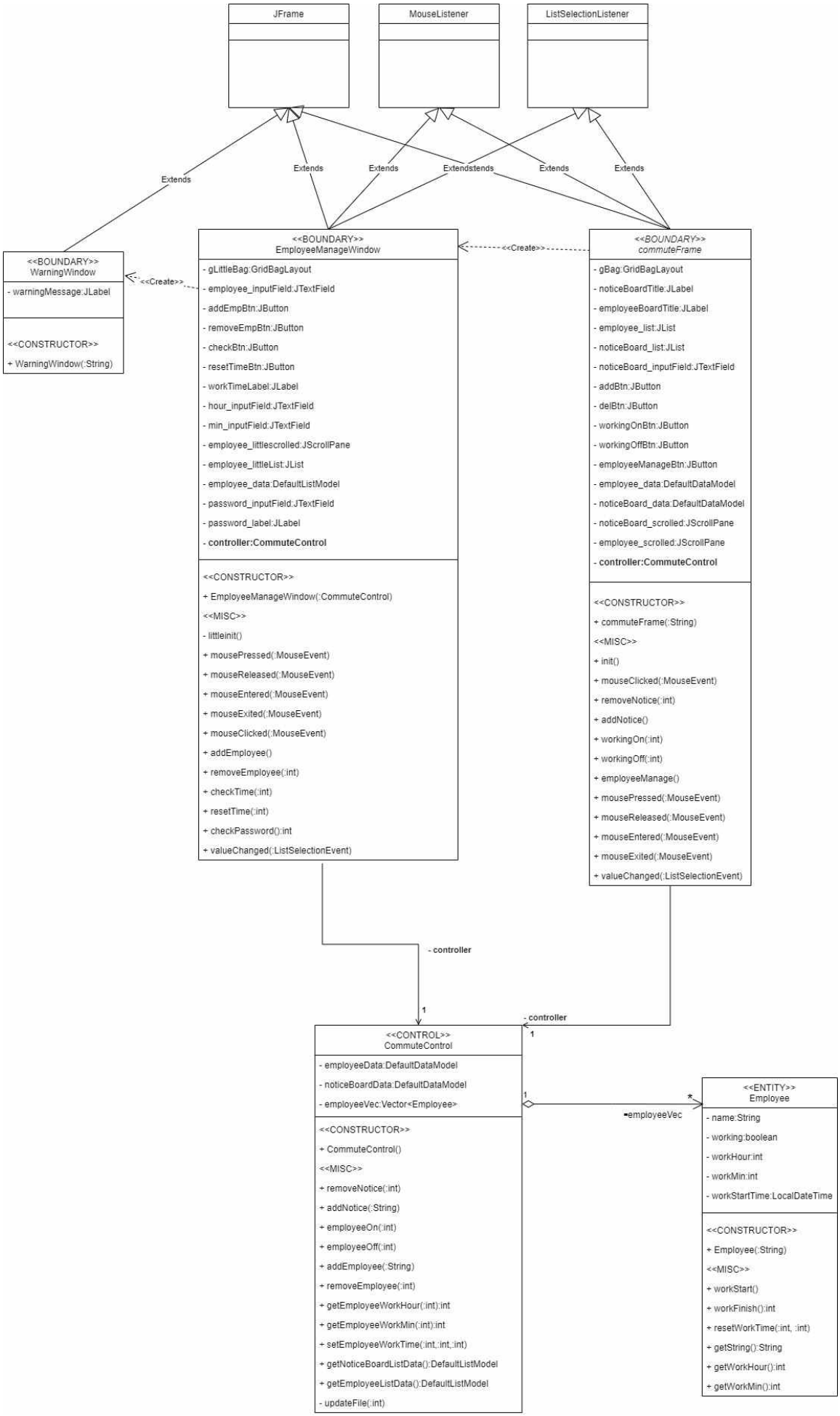


addEmployee()

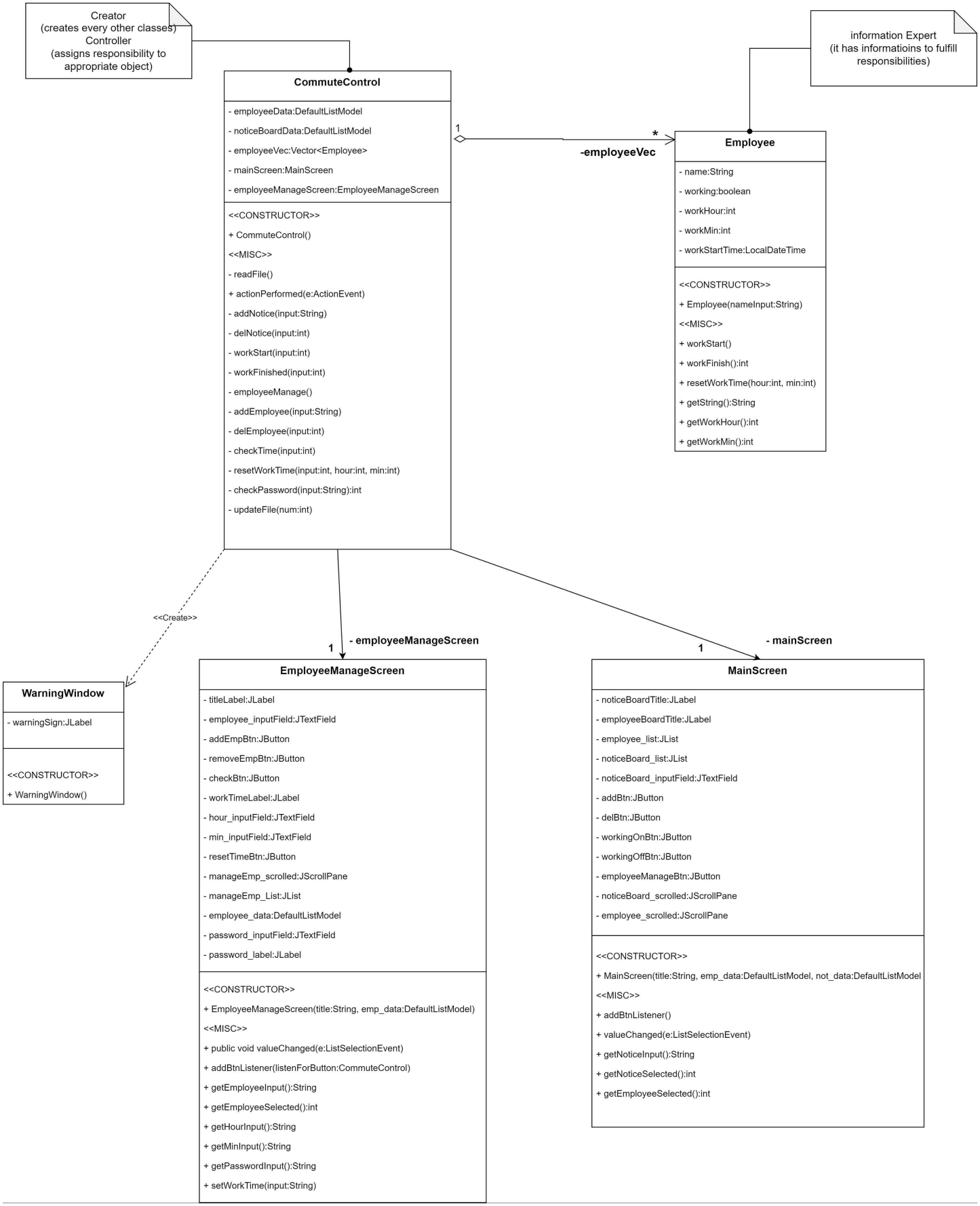


6. Class Diagram

before modification:



After modification:



swing GUI 구현을 위해 extend 하는 JFrame과 이벤트 관리를 위해 implement하는 ActionListener와 같은 라이브러리 제공 클래스들은 가독성을 위해 클래스 다이어그램에 표기하지 않았습니다.

- 실제 구동 영상입니다.
: https://youtu.be/9m8J6008aME
소리를 키고 보시면 자세한 동작 내용을 확인할 수 있습니다.