|  |  |
| --- | --- |
| **Due Date** | 2014.06.23 |
| **Name** | 20093267 김성근 |
| 20093348 최슬기 |
| 20103359 여대현 |
| 20103386 최성현 |

**<Vacation Tracking System>**

**Use-Case Specification: <휴가신청>**

**Version <2.4>**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <14/05/28> | <1.0> | First Version  Use Case & Actor 작성 | 김성근 최슬기 여대현 최성현 |
| <14/06/02> | <1.1> | Detail Use Case 작성 (휴가신청) | 김성근 최슬기 여대현 최성현 |
| <14.06/03> | <1.2> | Operation Contract 작성 | 김성근 최슬기 여대현 최성현 |
| <14/06/05> | <2.1> | System sequence diagram 작성 | 김성근 최슬기 여대현 최성현 |
| <14/06/16> | <2.2> | Class Diagram 작성 | 김성근 최슬기 여대현 최성현 |
| <14/06/18> | <2.3> | Glossary 수정 | 김성근 최슬기 여대현 최성현 |
| <14/06/20> | <2.4> | 전체적인 내용 검토 | 김성근 최슬기 여대현 최성현 |

**Revision History**

**Table of Contents**

**1. 프로젝트 목표 및 개요 5**

**2. 기존 업무처리 Activity Diagram, 시스템 개발 후 업무처리 Activity Diagram 6**

2.1 기존 업무 처리 Activity Diagram

2.2 시스템 개발 후 업무 처리 Activity Diagram

**3. 시스템 개발 시 고려해야 할 현실적 제한조건 및 해결방법 8**

3.1. 서버 및 네트워크 환경은 어떻게 정할 것인가?

3.2. 시스템 로그인은 어떻게 처리할 것인가?

3.3. 결제 시스템이 어떻게 흘러갈 것인가?

**4. Actor-Goal List 9**

4.1 Actor List

4.2 Goal List

**5. Fully-dressed Format [Brief] 11**

5.1 Basic flow

5.2 Alternative flows

5.3 Scenarios

**6. Brief Description**

**7. Basic Flow of Events**

**8. Alternative Flows**

**9. Domain Model**

9.1 Version 1

9.2 Version 2

**10. Operation Contract**

**11. System Sequence Diagram**

11.1 휴가신청(Request Vacation)

11.2 휴가신청(Refer Vacation)

11.3 휴가신청(Update Vacation)

11.4 관리자 승인(Approval Vacation)

11.5 매니저 승인(Approval Vacation)

**12. Design Class Diagram**

**13. Source Code**

**14. Glossary**

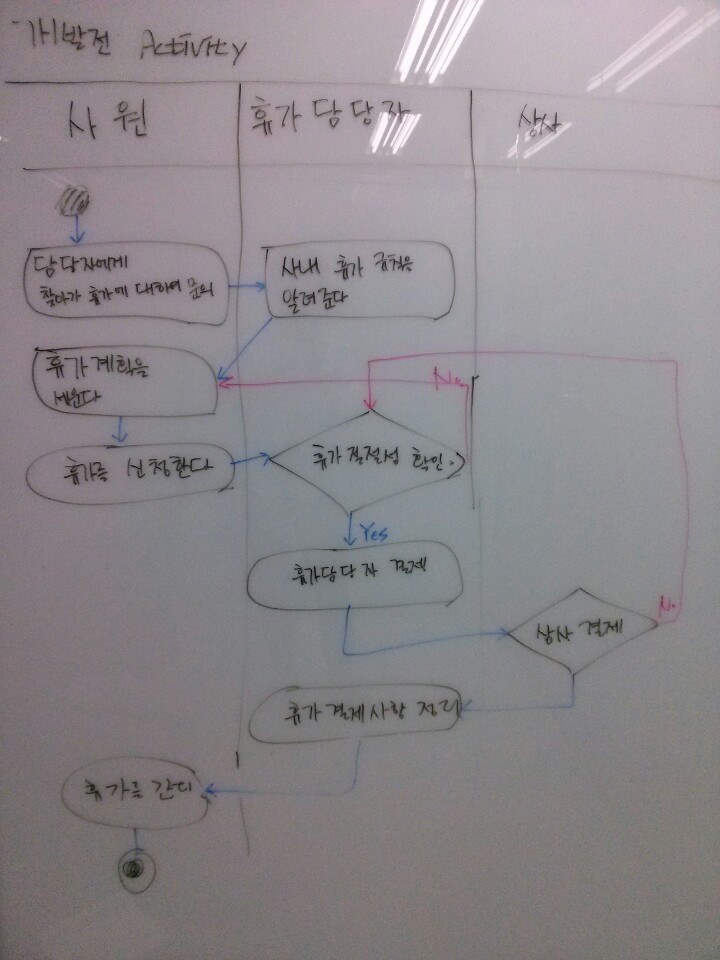
**1. 프로젝트 목표 및 개요**

- 서류와 구두로 회사에서 휴가를 결제하는 시스템 대신에, 회사 사규에 맞게 웹 서비스를 제공하여 회사 사원이 손쉽게 휴가 결제를 올릴 수 있는 시스템을 구현하는 것입니다. 또한 각 부서의 휴가 담당자가 자신의 부서에 대하여 서비스를 통하여 관리하고 통제 할 수 있는 서비스 입니다.

**2. 기존 업무처리 Activity Diagram, 시스템 개발 후 업무처리 Activity Diagram**

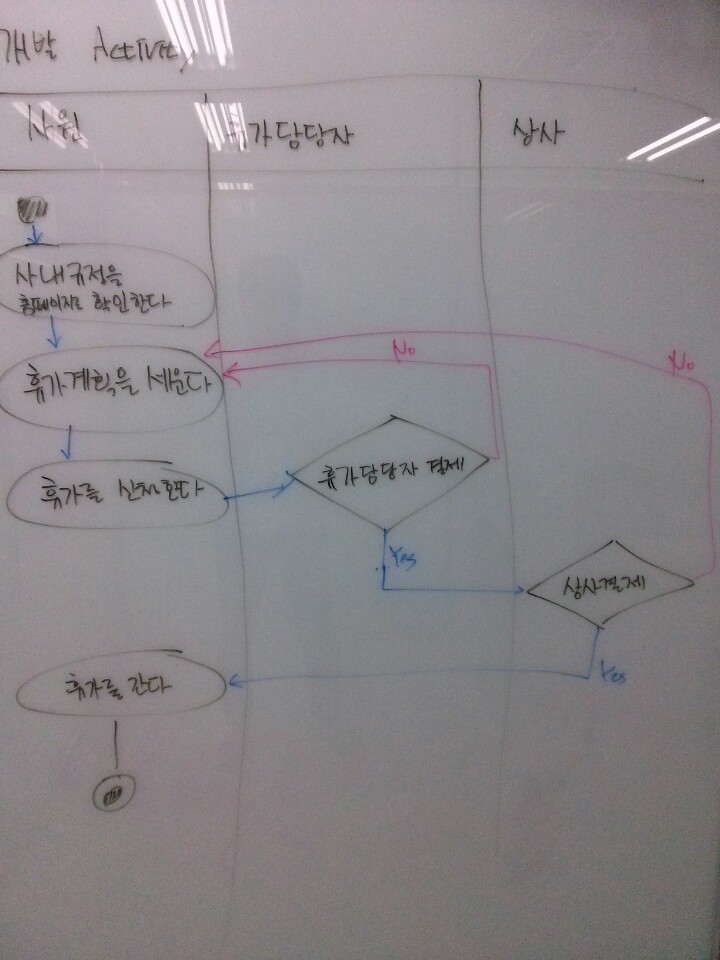
**2.1 기존 업무 처리 Activity Diagram**

기존의 업무처리는 사원이 휴가담당자에게 가서 사내 휴가 규칙을 알아본 뒤 자신의 휴가계획을 세우고 그 휴가 계획에 대해서 휴가담당자에게 확인을 받고 휴가 담당자가 상사에게 결제를 받는 흐름을 가지고 있습니다.



**2.2 시스템 개발 후 업무 처리 Activity Diagram**

사원이 시스템을 이용하여 사내규정 및 자신의 휴가를 확인하고 휴가계획을 세워 신청하게 되면, 휴가 담당자가 휴가계획을 확인하고 결제하고 상사가 결제하도록 업무 처리가 이루어져 있습니다. 해당 시스템을 사용하게 되면 휴가 담당자가 업무에 대한 큰 부담이 사라지고 사원이 올린 결제 사항에 대해서 확인만을 진행하게 되어 매우 간단해 집니다.



**3. 시스템 개발 시 고려해야 할 현실적 제한조건 및 해결방법**

**3.1. 서버 및 네트워크 환경은 어떻게 정할 것인가?**

사내 시스템이기 때문에 동시 접속하는 사람의 수가 적다고 가정 하였고, 네트워크 시스템이 인트라넷으로 구성된다고 가정하였습니다. 따라서 서버는 Linux 운영체제인 Ubuntu 와 높은 사양 시스템이 아닌 적당한 수준의 컴퓨터라고 가정하였습니다.

웹 서버의 경우에는 Tomcat6 을 사용한다고 가정하였고, Database는 Mysql을 쓴다고 가정하였으나 이것은 상황에 따라 바뀌는 것에 문제 없다고 가정하였습니다.

**3.2. 시스템 로그인은 어떻게 처리할 것인가?**

시스템 로그인 환경은 사내에 있는 데이터 베이스를 이용한다고 생각하였습니다. 로그인을 통한 상세한 정보는 사내 인트라넷에 로그인을 한 Session 정보를 통하여 가져온다고 가정하였습니다.

**3.3. 결제 시스템이 어떻게 흘러갈 것인가?**

2번에 Activity Diagram을 통하여 보였듯이 부서별 결제시스템으로 각 부서에 휴가담당자가 있으며, 휴가 담당자를 통하여 관리자에게 결제를 받아야 된다고 가정하고 구현하였습니다.

**4. Actor-Goal List**

**4.1 Actor List**

**- 사원(Staff)**

사원은 이 software를 통해 휴가를 신청하는 가장 중요한 actor이다. 이 actor는 처음 휴가를 사용한다 생각하고 이 회사 휴가 규정에 대하여 전혀 모르는 상태이다. 그래서 자신의 휴가 정보를 요청하고, 휴가 계획을 한 뒤 휴가를 신청하게 된다.

**- 직무대리인**

사원이 휴가를 나갈 때 휴가 기간 동안 사원이 진행하고 있던 일을 대신 처리할 actor이다. 휴가를 신청한 사원이 하고 있던 일을 잘 이해하고 있으며 사원의 휴가기간 동안 문제없이 일을 처리해야 한다. 그렇지 않을 경우 부적절한 직무대리인이다.

**- 휴가 관리자(Manager)**

각 부서의 휴가 관리자로 부서의 사원이 작성한 휴가 신청서를 처음 결정하는 actor이다. 각 부서의 사원들이 휴가 신청을 하였을 때 종합하여 상사에게 부서의 휴가 정보를 전달한다. 사원의 휴가가 부적절하거나 회사 내규에 적합하지 않을 경우, 많은 인원이 겹치는 경우 휴가를 제한하거나 휴가를 조정한다.

**- 상사(Superior)**

전 부서의 휴가를 관리하는 actor이다. 각 부서의 휴가 관리자로부터 부서 휴가 정보를 받아 전체적으로 회사 내규에 적합한지 확인을 하고 최종 승인을 한다.

**4.2 Goal List**

**<Primary use cases>**

**-휴가 정보 조회**

사원이 휴가 계획을 세우기 전 자신이 사용할 수 있는 휴가 일수, 종류를 확인하고 휴가를 나갈 수 있는 기간을 확인하는 것을 말한다.

**-전체 휴가목록 관리**

상사가 각 부서의 휴가 관리자로부터 휴가 신청 정보를 받아 전 부서의 휴가 목록을 관리하는 것을 말한다.

**<Supporting use cases>**

**-휴가 신청**

사원이 휴가 계획을 세운 뒤에 휴가를 나가기 위해서 software를 통해 자신이 사용할 휴가의 종류와 언제 휴가를 나갈지를 입력하는 것을 말한다.

**-휴가 확인**

부서의 휴가 관리자가 사원이 시청한 휴가의 내용을 software를 통해 확인하는 것을 말한다.

**-휴가를 결제함**

부서의 휴가 관리자가 사원들이 신청한 휴가가 회사 내규 규정에 적합한지를 확인하고 종합한 뒤 사원들의 휴가가 겹치는지 확인을 하여 사원의 휴가가 가능한지를 판단하는 것을 말한다.

**5. Fully-dressed Format [Brief]**

**5.1Basic flow**

1. 사원이 휴가 신청을 하기로 마음을 먹는다.
2. 본인의 사원 번호와 암호를 입력하여 VTS에 접속한다.
3. 자신의 잔여 휴가 일수를 확인하고, 휴가 사유와 신청 일수를 입력 후, 휴가를 신청한다.
4. 각 부서의 휴가 담당자는 자신의 부서원이 신청한 휴가 신청을 VTS를 통해 받는다.
5. 휴가 담당자는 신청 내역을 확인 하고 승인을 한다.
6. 상사는 휴가 담당자가 승인한 내역을 VTS를 통해 받는다.
7. 상사는 휴가 담당자의 처리 내역을 확인하고 승인을 한다.
8. 사원은 VTS를 통해 휴가 신청 승인 여부를 확인한다.
9. 사원은 휴가를 떠난다.

**5.2 Alternative flows**

A1. 사원이 휴가 신청을 하지 않기로 한다.

A2. 사원이 VTS에 접속을 실패 한다.

A3. 잔여 휴가 일수 부족으로 인해 VTS로부터 휴가 신청이 거부된다.

A4. 휴가 사유가 부적절하여 휴가 담당자가 휴가 승인을 거부한다.

A5. VTS의 문제로 다음 담당자가 요청을 제대로 받지 못한다.

A6. 담당자가 요청 처리를 제 때 하지 않는다.

A7. 상사가 휴가 승인을 거부한다.

A8. 휴가 신청 승인이 이루어 지는 도중에 사원이 휴가 신청을 취소한다.

**5.3 Scenarios**

1. 휴가 승인이 되고, 사원은 휴가를 떠난다. – Basic Flow
2. 사원이 다른 사정으로 인해 휴가 신청을 하지 않는다. – Alternative Flows : A1. 사원이 휴가 신청을 하지 않기로 한다.
3. 사원이 VTS에 접속하는 과정에서, 사원 번호 혹은 암호 입력을 잘못하여, VTS로부터 접속 불가 응답을 받는다. – Basic Flow, Alternative Flows : A2. 사원이 VTS에 접속을 실패한다.
4. 사원이 VTS를 통해 휴가 신청을 하는 과정에서, 본인이 신청하려는 휴가 일수보다 잔여 휴가 일수가 부족하여 VTS로부터 신청 거부 응답을 받는다. – Basic Flow, Alternative Flows : A3. 잔여 휴가 일수 부족으로 인해 VTS로부터 휴가 신청이 거부된다.
5. 휴가 담당자가 사원이 VTS를 통해 보낸 휴가 신청을 확인한다. 이 때, 휴가 사유가 부적절하여 휴가 승인을 거부한다. – Basic Flow, Alternative Flows : A4. 휴가 사유가 부적절하여 휴가 담당자가 휴가 승인을 거부한다.
6. VTS의 오류로 인해 사원의 신청이 휴가 담당자에게 제대로 전달되지 못한다. – Basic Flow, Alternative Flow : A5. VTS의 문제로 다음 담당자가 요청을 제대로 받지 못한다.
7. 휴가 담당자가 실수로 사원의 신청을 제 때 처리하지 못하여, 사원의 휴가 신청 기간이 지났다. – Basic Flow, Alternative Flow : A6. 담당자가 요청 처리를 제 때 하지 않는다.
8. VTS의 오류로 인해 휴가 담당자의 승인 내역을 상사가 제대로 받지 못한다. – Basic Flow, Alternative Flow : A5. VTS의 문제로 다음 담당자가 요청을 제대로 받지 못한다.
9. 상사가 휴가 담당자의 승인 내역을 너무 늦게 확인 했다. – Basic Flow, Alternative Flow : A6. 담당자가 요청 처리를 제 때 하지 않는다.
10. 상사가 해당 부서의 사정으로 인해 사원에게 휴가 승인을 해 줄 수 없는 상황이다. – Basic Flow, Alternative Flow : A7. 상사가 휴가 승인을 거부한다.
11. 휴가 담당자가 사원의 휴가 신청을 승인한 상황에서, 사원이 휴가 신청 요청을 취소 한다. – Basic Flow, Alternative Flow : A8. 휴가 신청 승인이 이루어 지는 도중에 사원이 휴가 신청을 취소한다.

**Use-Case Specification: <휴가 신청>**

**Scope**

• VTS(Vacation Tracking System)

**Primary Actor**

• 사원

**Stakeholders and Interests**

• 사원 : 회사의 휴가 관련한 내규를 따로 알아보지 않아도, 자신이 원하는 날짜에 더 편하게 휴가 신청을 하기를 원한다.

• 휴가 관리자 : 사원이 휴가를 신청할 때마다 휴가 잔여 일을 확인해야 하는 수고를 덜길 원한다.

• 상사 : 사원의 휴가 관리의 수고를 덜어 사원들이 업무에 더 집중하기를 원한다.

**6. Brief Description**

사원이 VTS를 통해서 손쉽게 휴가 신청을 하고, 담당자는 사원들의 휴가 관리를 더 손쉽게 한다.

**7. Basic Flow of Events**

1. 사원이 휴가 신청을 하기로 마음을 먹는다.

어떠한 이유로 사원은 휴가를 가기 위해, 회사에 휴가 신청을 하기로 마음을 먹는다.

1. 본인의 사원 번호와 암호를 입력하여 VTS에 접속한다.

사원이 휴가 신청을 위해 관리 시스템인 VTS에 접속하기 위한 자신의 사원 번호와 암호를 입력한다.

1. 휴가 사유와 신청 일수를 입력 후, 휴가를 신청한다.

자신의 잔여 휴가 일수를 확인한 후, 휴가 사유와 신청일수를 입력하여 휴가 신청을 한다.

1. 각 부서의 휴가 담당자는 자신의 부서원이 신청한 휴가 신청을 VTS를 통해 받는다.

사원의 휴가 신청은 VTS를 통해 휴가 담당자에게 전해진다.

휴가 담당자는 VTS를 통해 자신의 부서원의 휴가 신청을 받는다.

1. 휴가 담당자는 신청 내역을 확인하고 승인을 한다.

휴가 담당자는 사원의 휴가 기간과 휴가 사유를 확인한 후, 이상이 없을 경우 승인을 한다.

1. 상사는 휴가 담당자가 승인한 내역을 VTS를 통해 받는다.

휴가 담당자로부터 승인된 휴가 신청은 VTS를 통해 상사에게 전해진다.

상사는 VTS를 통해 휴가 담당자로부터 승인된 내용을 받는다.

1. 상사는 휴가 담당자의 처리 내역을 확인하고 승인을 한다.

상사는 휴가 담당자의 승인 내용이 타당할 경우, 그 내용을 승인한다.

1. 사원은 VTS를 통해 휴가 신청 승인 여부를 확인한다.

사원은 VTS를 통해서 휴가 담당자의 승인여부, 상사의 승인 여부를 확인 할 수 있다.

1. 사원은 휴가를 떠난다.

VTS를 통해 상사의 최종 승인을 확인한 사원은 휴가를 떠난다.

1. Use Case End

Use Case 종료.

**8. Alternative Flows**

1. 사원이 휴가 신청을 하지 않기로 한다.

Description : 사원이 휴가 신청을 마음 먹었었지만, 마음을 바꿔 신청을 하지 않기로 하는 상황.

Location : 2.1. 사원이 휴가 신청을 하기로 마음을 먹는다.

Condition : 사원이 실제로 휴가 신청을 하지 않는다.

Action :

1. 사원이 휴가 신청을 하지 않는다.
2. 10. Use Case End
3. 사원이 VTS에 접속을 실패 한다.

Description : 사원이 휴가 신청을 위해 VTS에 접속을 할 때, 사원 번호 혹은 암호의 불일치로 접속이 실패한 상황.

Location : 2.2. 본인의 사원 번호와 암호를 입력하여 VTS에 접속한다.

Condition : 사원이 VTS에 접속할 때, 사원 번호 혹은 암호를 잘못 입력하였다.

Action :

1. 사원이 VTS를 실행시킨다.
2. 접속을 위해 사원 번호 및 암호를 입력한다.
3. VTS에서 접속 실패 메시지를 보여주고, 다시 입력할 것인지 묻는다.
   1. 다시 입력할 것인가

* 2. 본인의 사원 번호와 암호를 입력하여 VTS에 접속한다.
  1. 휴가 신청을 취소할 것인가
* 10. Use Case End

1. 잔여 휴가 일수 부족으로 인해 VTS로부터 휴가 신청이 거부된다.

Description : 사원이 휴가를 신청할 때, 자신이 신청 하려는 일수보다 남은 휴가 일수가 더 적을 때, VTS가 자동으로 휴가 신청을 거부하는 상황.

Location : 2.3. 휴가 사유와 신청 일수를 입력 후, 휴가를 신청한다.

Condition : 사원이 자신의 잔여 휴가 일수보다 많은 휴가 일수를 신청한다.

Action :

1. 사원이 휴가 신청을 위해 휴가 사유와 휴가 일수를 입력한다.
2. 휴가 신청버튼을 누른다.
3. VTS가 휴가 신청일수가 잔여 일수보다 많은 것을 알려준다.
   1. 신청 일수를 변경할 것인가

* 3. 휴가 사유와 신청 일수를 입력 후, 휴가를 신청한다.
  1. 휴가를 취소할 것인가
* 10. Use Case End

1. 휴가 사유가 부적절하여 휴가 담당자가 휴가 승인을 거부한다.

Description : 사원이 VTS를 통해 휴가 사유와 휴가 일수를 입력하여 휴가를 신청 하면 휴가 담당자가 그 신청 내역을 확인 할 수 있다. 이 때, 휴가 사유가 적절하지 않을 경우 휴가 담당자가 신청을 거부한 상황.

Location : 2.5.휴가 담당자는 신청 내역을 확인 하고 승인을 한다.

Condition : 부적절한 휴가 사유로 인해 휴가 담당자가 사원의 신청을 거부한다.

Action :

1. VTS로부터 휴가 담당자는 사원의 휴가 신청을 받는다.
2. 휴가 담당자는 휴가 사유를 판단한다.
3. 휴가 사유가 부적절하여 휴가 승인을 거부한다.
4. VTS는 승인이 거부된 것을 사원에게 알려준다.
   1. 신청을 다시 할 것인가

* 3. 휴가 사유와 신청 일수를 입력 후, 휴가를 신청한다.
  1. 신청을 취소할 것인가
* 10. Use Case End

1. VTS의 문제로 다음 담당자가 요청을 제대로 받지 못한다.

Description : 사원이 VTS를 통해 휴가 신청을 하였지만, 어떠한 문제로 인해 VTS가 휴가 담당자에게 제대로 된 정보를 전달하지 못한 상황. 또는 휴가 담당자의 승인 내역을 VTS의 문제로 인해 상사가 제대로 전달 받지 못한 상황.

Location :

2.4. 부서의 휴가 담당자는 자신의 부서원이 신청한 휴가 신청을 VTS를 통해 받는다.

2.6. 상사는 휴가 담당자가 승인한 내역을 VTS를 통해 받는다.

Condition :

1. 휴가 담당자가 사원의 신청을 제대로 받지 못한다.
2. 상사가 휴가 담당자의 승인 내역을 제대로 받지 못한다.

Action :

1. 사원이 VTS를 통해 휴가를 신청한다.
2. 순간 정전이 되어 신청 정보가 휴가 담당자에게 제대로 전달되지 않는다.
3. VTS가 재전송을 요청한다.
4. 4. 각 부서의 휴가 담당자는 자신의 부서원이 신청한 휴가 신청을 VTS를 통해 받는다.
5. 휴가 담당자가 휴가 승인 후, VTS를 통해 상사에게 전달한다.
6. 순간 정전이 되어 신청 정보가 상사에게 제대로 전달되지 않는다.
7. VTS가 재전송을 요청한다.
8. 6. 상사는 휴가 담당자가 승인한 내역을 VTS를 통해 받는다.
9. 담당자가 요청 처리를 제 때 하지 않는다.

Description : 사원이 타당한 사유와 적절한 기간으로 휴가를 신청하였으나, 휴가 담당자가 사원이 신청한 휴가의 기한이 지난 뒤 신청 내역을 확인한 상황. 또는 휴가 담당자가 휴가 승인을 일찍 하였으나, 상사가 너무 늦게 승인을 한 상황.

Location :

2.5. 휴가 담당자는 신청 내역을 확인하고 승인을 한다.

2.7. 상사는 휴가 담당자의 처리 내역을 확인하고 승인을 한다.

Condition : 사원의 희망 휴가 날짜보다 늦게 승인이 난다.

Action :

1. A 사원이 VTS를 통해 휴가 신청을 한다.
2. 휴가 담당자가 실수로 A 사원의 휴가 승인을 건너 뛰고 업무 처리를 한다.
3. A 사원의 희망 휴가 날짜가 지나고, 휴가 담당자가 늦었다는 것을 알아 차린다.
4. A 사원에게 휴가를 갈 것인지 취소할 것인지 묻는다.
   1. 사원이 휴가를 원할 경우

* 9. 사원은 휴가를 떠난다.
  1. 사원이 휴가 취소를 원할 경우
* 10. Use Case End

1. 휴가 담당자가 VTS를 통해 휴가 요청 승인 후 그 내역을 상사에게 보낸다.
2. 상사가 휴가 담당자의 승인 요청을 보지 못한다.
3. A사원의 희망 휴가 날짜가 지나고, 상사는 그것을 알아 차린다.
4. A사원에게 휴가를 갈 것인지 취소할 것인지 묻는다.
   1. 사원이 휴가를 원할 경우

* 9. 사원은 휴가를 떠난다.
  1. 사원이 휴가 취소를 원할 경우
* 10. Use Case End

1. 상사가 휴가 승인을 거부한다.

Description : 휴가 담당자가 사원의 휴가 요청을 승인하였지만, 당시 내외부적 어떤 상황에 의해 상사가 사원의 휴가를 승인할 수 없는 상황.

Location : 2.7. 상사는 휴가 담당자의 처리 내역을 확인하고 승인을 한다.

Condition : 어떠한 이유로 상사가 휴가 담당자의 승인 요청을 거부한다.

Action :

1. 휴가 담당자가 사원의 휴가 요청 승인을 한다.
2. VTS를 통해서 상사에게 승인 요청을 보낸다.
3. 상사가 승인을 거부한다.
4. 10. Use Case End
5. 휴가 신청 승인이 이루어 지는 도중에 사원이 휴가 신청을 취소한다.

Description : 사원이 휴가 신청을 하고, 휴가 담당자 혹은 상사의 승인 대기 중에 사원이 휴가 계획을 바꾸게 되고, 자신이 제출한 휴가 신청을 취소하는 상황.

Location :

2.4. 부서의 휴가 담당자는 자신의 부서원이 신청한 휴가 신청을 VTS를 통해 받는다.

2.5. 휴가 담당자는 신청 내역을 확인하고 승인을 한다.

2.6. 상사는 휴가 담당자가 승인한 내역을 VTS를 통해 받는다.

2.7. 상사는 휴가 담당자의 처리 내역을 확인하고 승인을 한다.

2.8. 사원은 VTS를 통해 휴가 신청 승인 여부를 확인한다.

Condition : 휴가 승인이 진행되는 도중 사원이 휴가 신청을 취소 한다.

Action :

1. 사원이 VTS를 통해 휴가 신청을 한다.
2. 휴가 담당자가 VTS를 통해 사원의 휴가 요청 승인을 받았다.
   1. 사원이 휴가 취소를 한다.

* 10. Use Case End

1. 휴가 담당자가 사원의 휴가 요청 승인을 하였다.
   1. 사원이 휴가 취소를 한다.

* 10. Use Case End

1. 휴가 담당자가 VTS를 통해 상사에게 최종 승인 요청을 한다.
   1. 사원이 휴가 취소를 한다.

* 10. Use Case End

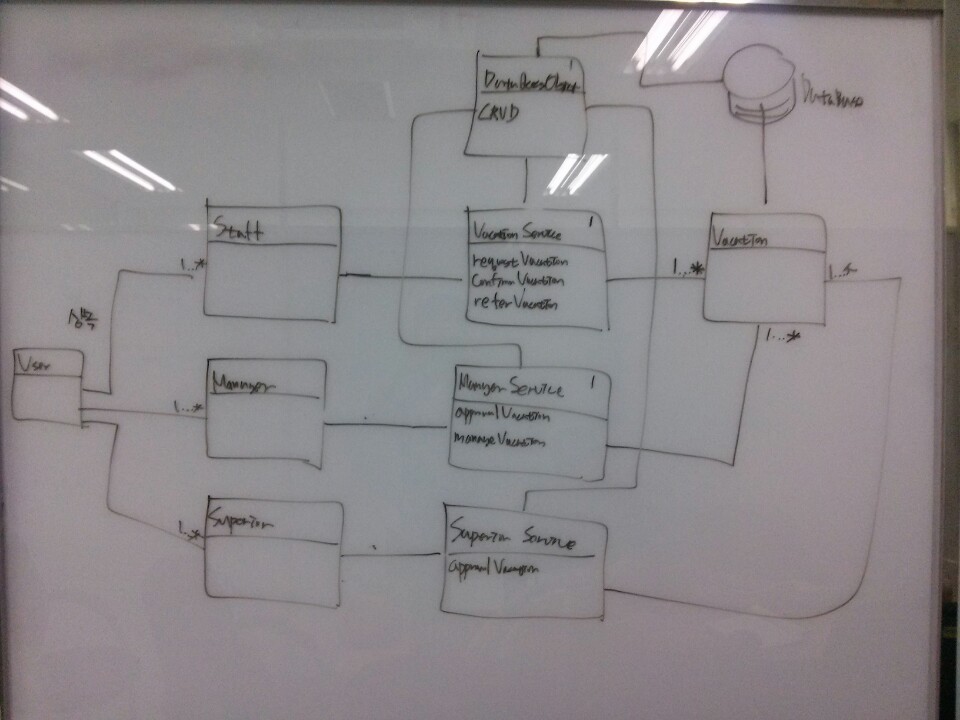
1. 상사가 VTS를 통해 휴가 담당자의 승인 내역을 확인한다.
   1. 사원이 휴가 취소를 한다.

* 10. Use Case End

1. 사원이 VTS를 통해 휴가 신청 승인 여부를 확인한다.
2. 사원이 휴가 취소를 한다.
3. 10. Use Case End

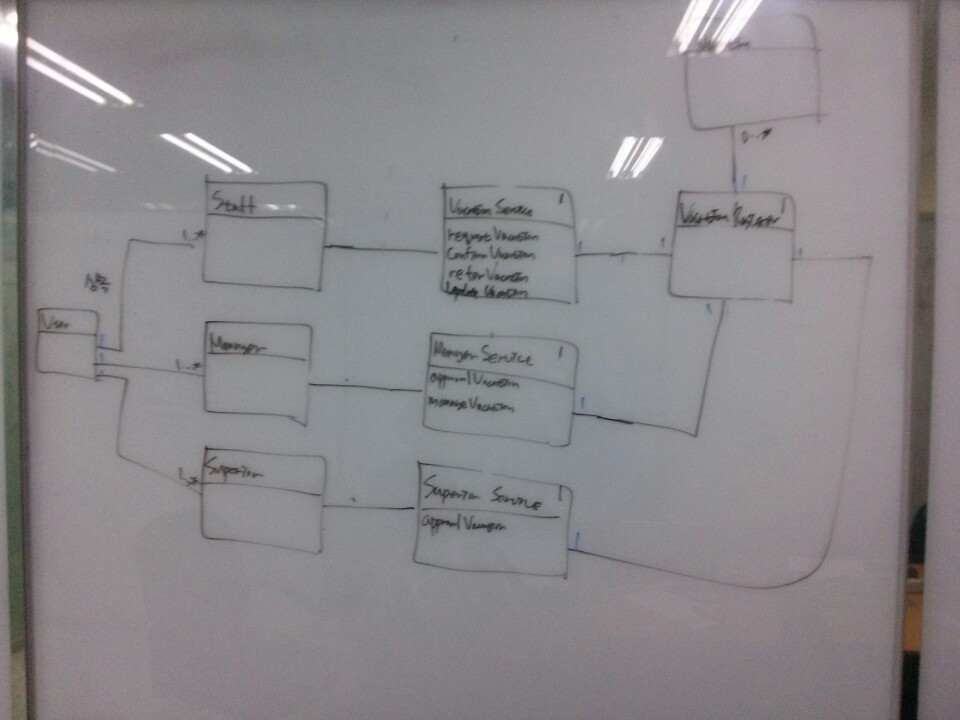
**9. Domain Model**

**9.1 Version 1**



- 설명

**9.2 Version 2**



- 설명

**10. Operation Contract**

**Operation :** RequestVacation()

**Cross reference :** 휴가 신청

**Precondition :**

1. 사원이 자신의 휴가 정보를 안다.

2. 사원이 사내 휴가 규정을 안다.

**Postcondition :**

1. Vacation 객체가 하나 생성되고, 사원의 휴가정보가 저장된다.

2. Vacation Register 의 Vacation List가 갱신된다.

**Operation :** Update Vacation(vacation)

**Cross reference :** 휴가 수정

**Precondition :**

1. 사원이 이미 휴가를 신청하여 Vacation 객체가 생성되었다.

2. 관리자 또는 상사 Approval에서 사원의 Vacation의 Approval Flag가 ‘1’이다.

**Postcondition :**

1. Vacation 객체의 정보인 기간, 종류가 수정되었다.

2. Approval Flag가 0이되어 다시 승인을 기다린다

**Operation :** Refer Vacation()

**Cross reference :** 휴가 확인

**Precondition :**

1. 처음 사원이 휴가 정보를 모름.

**Postcondition :**

1. 사원이 휴가정보를 알게 된다.

2. 사원이 휴가정보에 맞는 계획을 세울 수 있게 된다.

**Operation :** Approval Vacation(vac, flag)

**Cross reference :** 관리자 승인

**Precondition :**

1. 사원이 작성한 휴가 정보가 Vacation 객체에 담겨 있다.
2. Vacation의 Approval flag가 ‘0이다 (거절당하지 않은 상태)

**Postcondition :**

1. Vacation 의 Approval flag가 휴가 담당자가 입력한 flag로 바뀐다.

**Operation :** Approval Vacation(vac, flag)

**Cross reference :** 매니저 승인

**Precondition :**

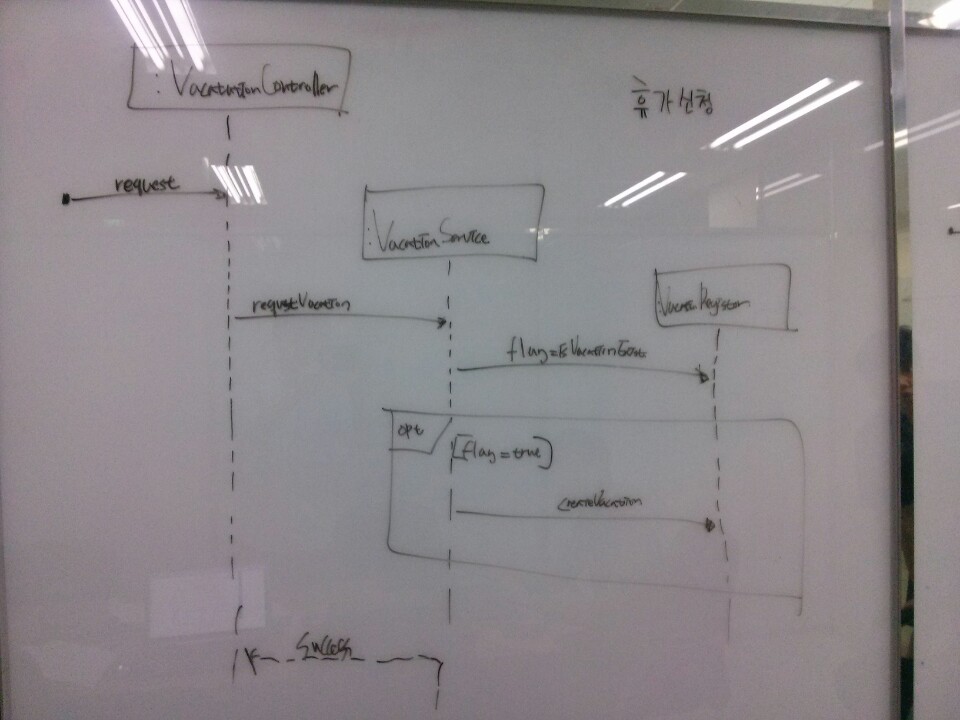
1. 사원이 작성한 휴가 정보가 Vacation 객체에 담겨 있다.
2. Vacation의 Approval flag가 ‘0이다 (거절당하지 않은 상태)

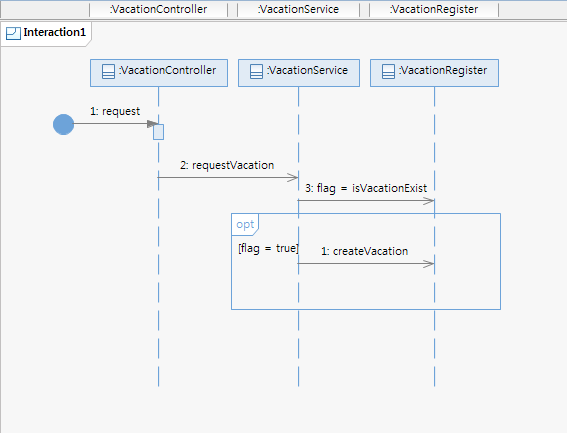
**Postcondition :**

1. Vacation 의 Approval flag가 휴가 담당자가 입력한 flag로 바뀐다.

**11. System Sequence Diagram**

**11.1 휴가신청(Request Vacation)**

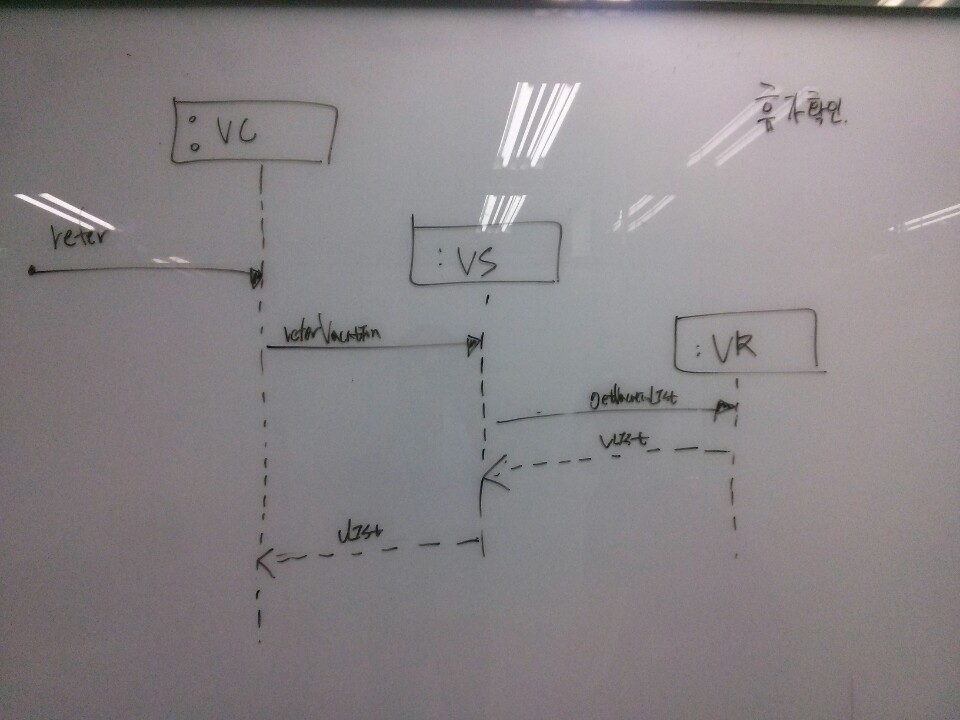


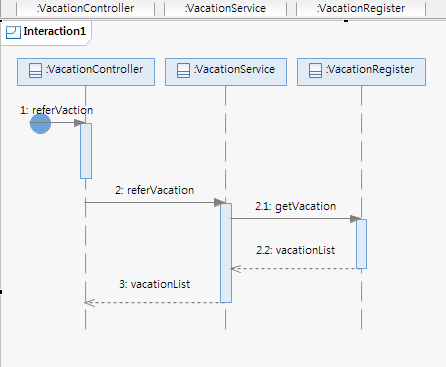


- 설명

사용자에게 요청이 들어오면 Façade Controller 인 Vacation Controller통해서 외부에 이벤트를 전달 받고, Singleton Object인 Vacation Service를 통해 새로운 Vacation(휴가계획)을 생성하게 되는 흐름입니다.

**11.2 휴가확인(Refer Vacation)**

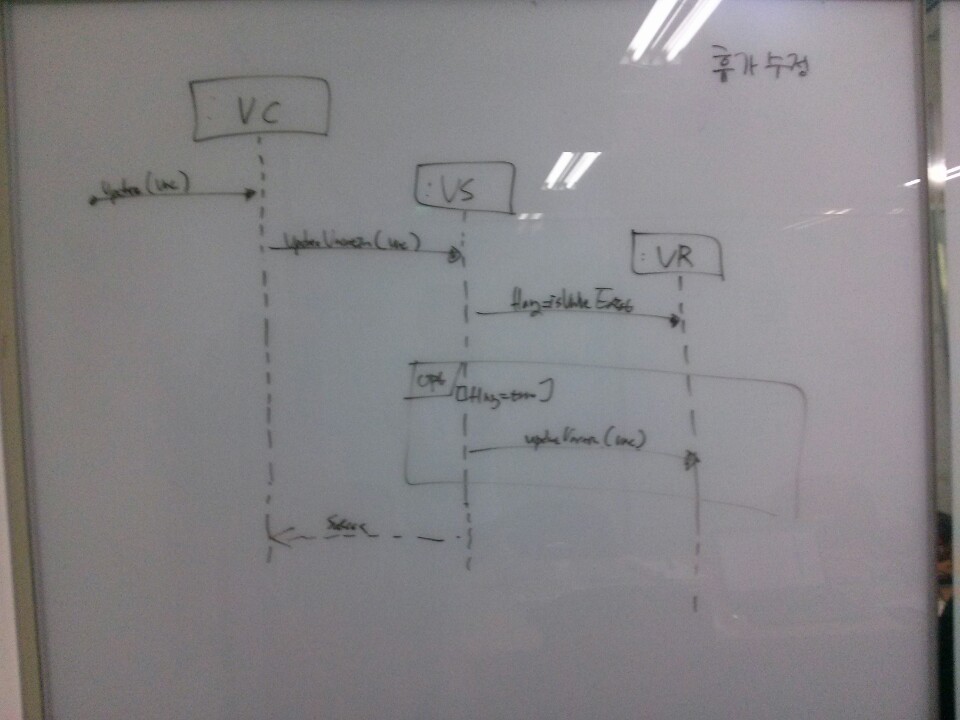


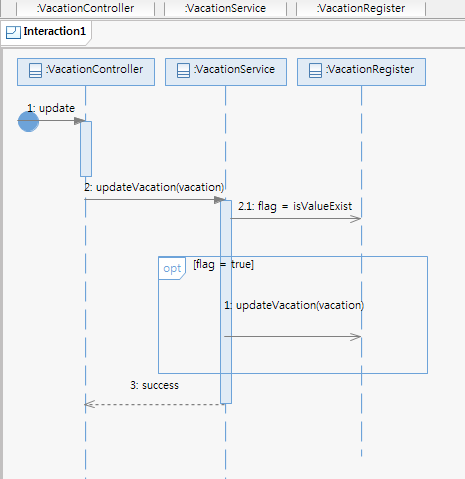


- 설명

같은 방식으로 사용자가 자신의 휴가에 대한 정보를 확인하기 위해서 요청을 하게 되면 Vacation Register 에 존재하는 VacationList를 외부 사용자가 볼 수 있도록 전달하는 것을 보여주는Diagram 입니다.

**11.3 휴가수정(Update Vacation)**

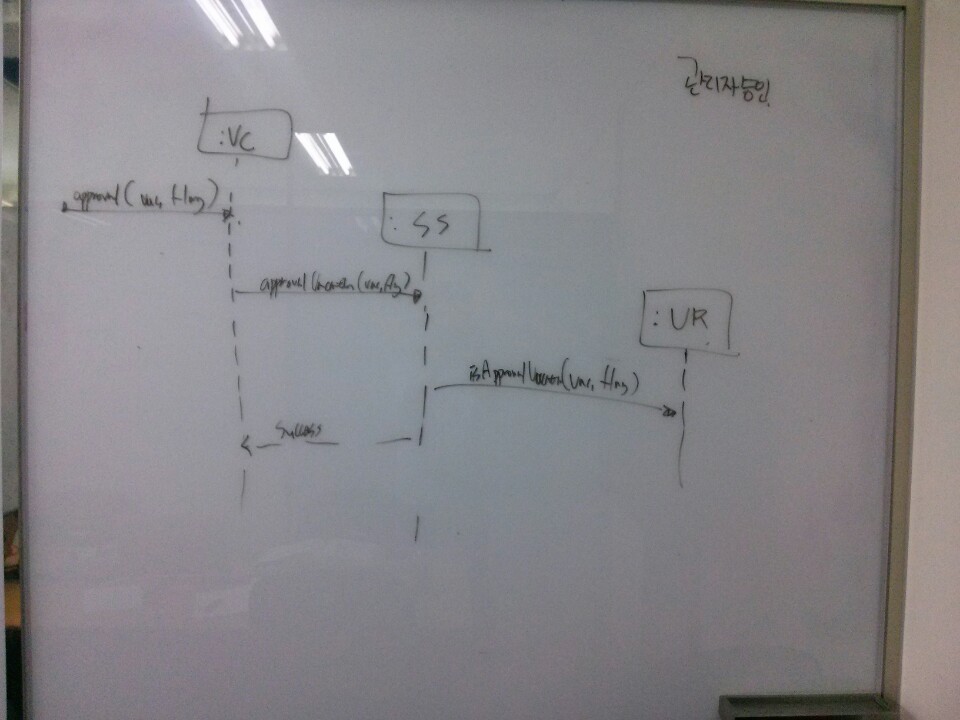


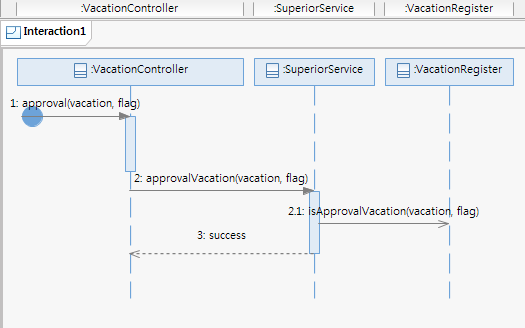


- 설명

동일하게 같은 Controller를 사용하며 Vacation의 정보를 수정하기 위하여 외부 Event 발생시에 기존 Vacation에 대한 정보와 수정할 Vacation에 대한 정보를 받아서 Vacation Register에 있던 기존에 정보에 대해서 수정하는 Diagram입니다.

**11.4 관리자 승인(Approval Vacation)**

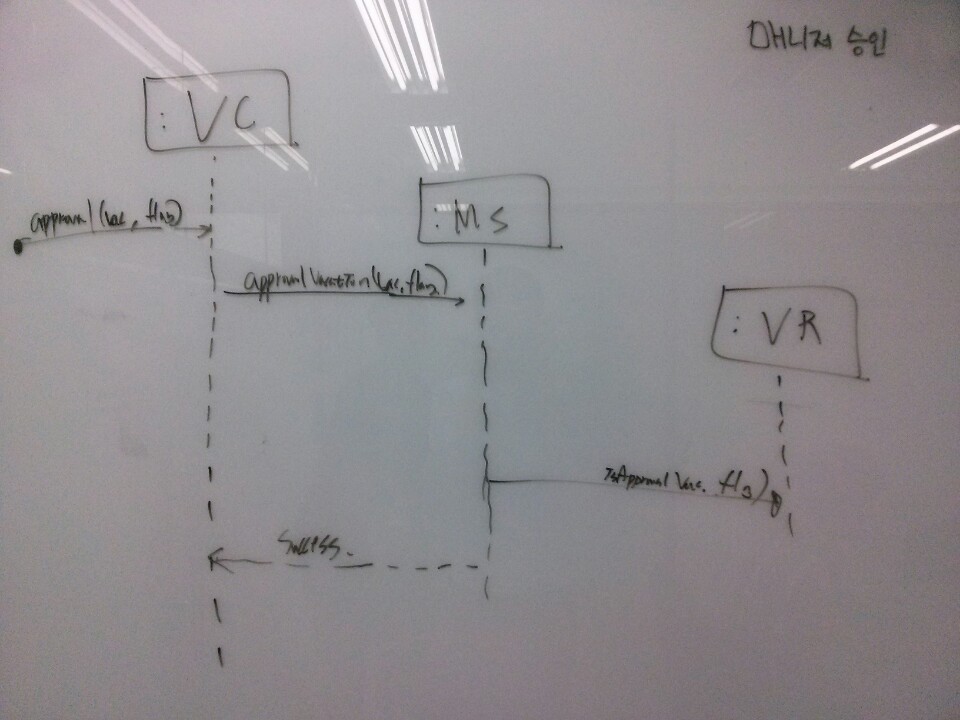


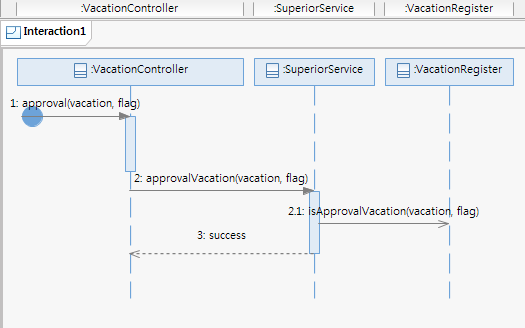


- 설명

Façade Controller로 유지시키기 위하여 VacationService 에 연결했던 Vacation Controller를 그대로 사용하며, Singleton으로 생성된 SuperiorService를 이용하여 Vacation Register에 있는 Vacation에 대한 승인을 처리해 주는 Diagram입니다.

**11.5 매니저 승인(Approval Vacation)**

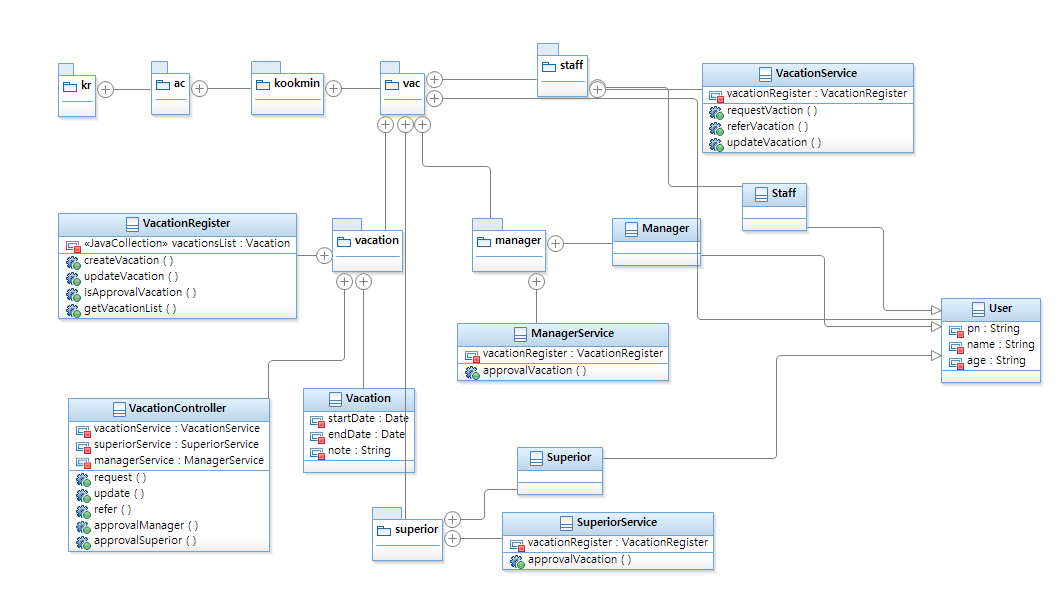




- 설명

관리자 승인과정과 동일하게 진행됩니다.

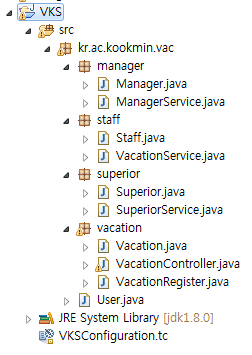
**12. Design Class Diagram**



- 설명

가장먼저 Controller Pattern을 이용하여 외부의 Event를 입력 받는 VacationController를 Class로 지정하였습니다. 이것은 FacadeController로 외부의 Event는 모두 이곳으로 전달되어집니다. High cohesion을 증가 시키기 위하여 User Class를 상속받도록 Staff, Manager, Superior 을 설계하였습니다. Low Coupling 하기 위하여 각 Actor가 수행하는 행위에 대해서 각각의 클래스를 생성하여 나누었습니다. 그리고 이것은 Information Expert에 기반하여 수행하였습니다.

**13. Source Code**



**Manager.java**

**package** kr.ac.kookmin.vac.manager;

**import** kr.ac.kookmin.vac.User;

**public** **class** Manager **extends** User {

}

**ManagerService.java**

**package** kr.ac.kookmin.vac.manager;

**import** kr.ac.kookmin.vac.vacation.Vacation;

**import** kr.ac.kookmin.vac.vacation.VacationRegister;

**public** **class** ManagerService {

**private** VacationRegister vacationRegister;

**public** **void** setVacationRegister(VacationRegister vacationRegister) {

**this**.vacationRegister = vacationRegister;

}

**public** **void** approvalVacation(Vacation vacation, Boolean flag) {

vacationRegister.isApprovalVacation(vacation, flag);

}

}

**Staff.java**

**package** kr.ac.kookmin.vac.staff;

**import** kr.ac.kookmin.vac.User;

**public** **class** Staff **extends** User {

}

**VacationService.java**

**package** kr.ac.kookmin.vac.staff;

**import** java.util.List;

**import** kr.ac.kookmin.vac.User;

**import** kr.ac.kookmin.vac.vacation.Vacation;

**import** kr.ac.kookmin.vac.vacation.VacationRegister;

**public** **class** VacationService {

**private** VacationRegister vacationRegister;

**public** **void** setVacationRegister(VacationRegister vacationRegister) {

**this**.vacationRegister = vacationRegister;

}

**public** **void** requestVaction(Vacation vacation) {

vacationRegister.createVacation(vacation);

}

**public** List<Vacation> referVacation(User user) {

**return** vacationRegister.getVacationList(user);

}

**public** **void** updateVacation(Vacation oldVacation, Vacation newVacation) {

vacationRegister.updateVacation(oldVacation, newVacation);

}

}

**Superior.java**

**package** kr.ac.kookmin.vac.superior;

**import** kr.ac.kookmin.vac.User;

**public** **class** Superior **extends** User {

}

**SuperiorService.java**

**package** kr.ac.kookmin.vac.superior;

**import** kr.ac.kookmin.vac.vacation.Vacation;

**import** kr.ac.kookmin.vac.vacation.VacationRegister;

**public** **class** SuperiorService {

**private** VacationRegister vacationRegister;

**public** **void** setVacationRegister(VacationRegister vacationRegister) {

**this**.vacationRegister = vacationRegister;

}

**public** **void** approvalVacation(Vacation vacation, Boolean flag) {

vacationRegister.isApprovalVacation(vacation, flag);

}

}

**Vacation.java**

**package** kr.ac.kookmin.vac.vacation;

**import** java.util.Date;

**public** **class** Vacation {

**private** Date startDate;

**private** Date endDate;

**private** String note;

**public** Date getStartDate() {

**return** startDate;

}

**public** **void** setStartDate(Date startDate) {

**this**.startDate = startDate;

}

**public** Date getEndDate() {

**return** endDate;

}

**public** **void** setEndDate(Date endDate) {

**this**.endDate = endDate;

}

**public** String getNote() {

**return** note;

}

**public** **void** setNote(String note) {

**this**.note = note;

}

}

**VacationRegister.java**

**package** kr.ac.kookmin.vac.vacation;

**import** java.util.List;

**import** kr.ac.kookmin.vac.User;

**public** **class** VacationRegister {

**private** List<Vacation> vacationsList;

**public** **void** setVacationsList(List<Vacation> vacationsList) {

**this**.vacationsList = vacationsList;

}

**public** **void** createVacation(Vacation vacation) {

}

**public** **void** updateVacation(Vacation oldVacation, Vacation newVacation) {

}

**public** **void** isApprovalVacation(Vacation vacation, Boolean flag) {

}

**public** List<Vacation> getVacationList(User user) {

**return** **this**.vacationsList;

}

}

**VacationController.java**

**package** kr.ac.kookmin.vac.vacation;

**import** java.util.List;

**import** kr.ac.kookmin.vac.User;

**import** kr.ac.kookmin.vac.manager.ManagerService;

**import** kr.ac.kookmin.vac.staff.VacationService;

**import** kr.ac.kookmin.vac.superior.SuperiorService;

**public** **class** VacationController {

**private** VacationService vacationService;

**private** SuperiorService superiorService;

**private** ManagerService managerService;

**public** **void** setVacationService(VacationService vacationService) {

**this**.vacationService = vacationService;

}

**public** **void** setSuperiorService(SuperiorService superiorService) {

**this**.superiorService = superiorService;

}

**public** **void** setManagerService(ManagerService managerService) {

**this**.managerService = managerService;

}

**public** **void** request(Vacation vacation) {

vacationService.requestVaction(vacation);

}

**public** **void** update(Vacation oldVacation, Vacation newVacation) {

vacationService.updateVacation(oldVacation, newVacation);

}

**public** **void** refer(User user) {

List<Vacation> userList = vacationService.referVacation(user);

}

**public** **void** approvalManager(Vacation vacation, Boolean flag) {

managerService.approvalVacation(vacation, flag);

}

**public** **void** approvalSuperior(Vacation vacation, Boolean flag) {

superiorService.approvalVacation(vacation, flag);

}

}

**User.java**

**package** kr.ac.kookmin.vac;

**public** **class** User {

**private** String pn;

**private** String name;

**private** String age;

**public** String getAge() {

**return** age;

}

**public** String getName() {

**return** name;

}

**public** String getPn() {

**return** pn;

}

**public** **void** setAge(String age) {

**this**.age = age;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** setPn(String pn) {

**this**.pn = pn;

}

}

**14. Glossary**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |