

# Software Modeling & Analysis

OSP stage 2050 & 2060

[Construct & Test]

- 1 to 10 CPT Tool -

<b>Team.#</b>	6
<b>과목명</b>	소프트웨어 모델링 및 분석
<b>담당교수</b>	유준범 교수님
<b>팀원</b>	201211938 황준익
	201310350 손성호
	201414135 이광제
	201212088 이용주

## Contents

Activity 2050. Construct .....	20
Activity 2051. Implement Class & Methods Definition .....	20
Activity 2052. Implement Windows.....	39
Activity 2055. Write Unit Test Code.....	50
Activity 2060. Test.....	56
Activity 2061. Unit Testing .....	56
Activity 2063. System Testing.....	58
Activity 2067. Testing Traceability Analysis .....	61

**Activity 2050. Construct****Activity 2051. Implement Class & Methods Definition****- Class Definition**

<b>Type</b>	Class
<b>Name</b>	Task
<b>Purpose</b>	전체 Category, Constraints, Property 들을 관리하는 클래스
<b>Overview</b>	N/A
<b>Cross References</b>	<p>Functions : R 1.1, R 1.2, R 2.1, R 2.2, R 2.3, R 3.1, R 3.2, R 3.3, R 4.1, R 4.2, R 4.3, R 5.1, R 5.2, R 5.3, R 6.1, R 6.2.1, R 6.2.2, R 6.4, R 7.1</p> <p>Use Cases : Load File, Save File, New Category, Remove Category, Modify Category, New Value, Remove Value, Modify Value, New Constraints, Remove Constraints, Modify Constraints, New Property, Remove Property, Modify Property, Calculate, Show Result, Filter Representative Value, Filter Property, Save Result, Provide Help, Check True/False</p>
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Class
<b>Name</b>	Category
<b>Purpose</b>	각 Category 들의 정보를 저장하는 클래스.
<b>Overview</b>	N/A
<b>Cross References</b>	<p>Functions : R 1.1, R 2.1, R 2.2, R 2.3, R 3.1, R 3.2, R 3.3</p> <p>Use Cases : Load File, New Category, Remove Category, Modify Category, New Rep. Value, Remove Rep. Value, Modify Rep. Value</p>
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Class
<b>Name</b>	RepValue
<b>Purpose</b>	각 Representative Value 들의 정보를 저장하는 클래스.
<b>Overview</b>	N/A
<b>Cross References</b>	Functions : R 1.1, R 3.1, R 3.2, R 3.3, R 4.1, R 4.2, R 4.3, R 5.2, R 6.1, R 6.2, R 6.2.1, R 7.1 Use Cases : Load File, New Rep. Value, Remove Rep. Value, Modify Rep. Value, New Constraints, Remove Constraints, Modify Constraints, Calculate, Show Result, Filter Representative Value, Check True/False
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Class
<b>Name</b>	Constraints
<b>Purpose</b>	각 Constraints 들의 정보를 저장하는 클래스.
<b>Overview</b>	N/A
<b>Cross References</b>	Functions : R 1.1, R 2.2, R 3.2, R 4.1, R 4.2, R 4.3, R 5.2 Use Cases : Load File, Remove Category, Remove Rep. Value, New Constraints, Remove Constraints, Modify Constraints, Remove Property
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Class
<b>Name</b>	Property
<b>Purpose</b>	각 Property 들의 정보를 저장하는 클래스.
<b>Overview</b>	N/A
<b>Cross References</b>	Functions : R 1.1, R 2.2, R 3.2, R 4.2, R 5.1, R 5.2, R 5.3, R 6.1, R 6.2.2 Use Cases : Load File, Remove Category, ,Remove Rep. Value, Remove Constraints, New Property, Remove Property, Modify Property, Calculate, Filter Property
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Class
<b>Name</b>	Result
<b>Purpose</b>	Test Case 계산 결과를 저장하기 위한 클래스.
<b>Overview</b>	N/A
<b>Cross References</b>	Functions : R 6.1, R 6.2, R 6.2.1, R 6.2.2, R 7.1 Use Cases : Calculate, Show Result, Filter Representative Value, Filter Property, Check True/False
<b>Exceptional Courses of Events</b>	N/A

**- Class\_GUI Definition**

<b>Type</b>	Class_GUI
<b>Name</b>	ComponenetSet
<b>Purpose</b>	Category, Representative Value, Property 의 리스트를 가시적으로 보여주며 sel, del, mdf 버튼을 통해 유저의 행동을 수행해주는 컴포넌트 조합 클래스
<b>Overview</b>	Type 인자를 이용해서 어떤 객체를 대변하는 지 식별한다.
<b>Cross Reference</b>	Functions : R 2.2, R 2.3, R 3.2, R 3.3, R 5.2, R 5.3 Use Cases : Remove Category, Modify Category, Remove Rep. Value, Modify Rep. Value, Remove Property, Modify Property
<b>Exceptional Courses of Events</b>	-

<b>Type</b>	Class_GUI
<b>Name</b>	CstComponenetSet
<b>Purpose</b>	선택된 Rep. Value 의 Constraints 관계에 대한 리스트를 가시적으로 보여주며, combo, del, sel 버튼을 통해 유저의 행동을 수행하는 컴포넌트 조합 클래스
<b>Overview</b>	Oldtype 인자를 통해 어떤 객체를 대변하는지 식별한다.
<b>Cross Reference</b>	Functions : R 4.1, R 4.2, R 4.3 Use Cases : New Constraints, Remove Constraints, Modify Constraints
<b>Exceptional Courses of Events</b>	-

---

**- Method Definition**

<b>Type</b>	Method
<b>Name</b>	Load_work
<b>Purpose</b>	User 가 기존에 작업 중이던 파일을 불러오도록 한다.
<b>Cross References</b>	Functions : R 1.1 Use Cases : Load File
<b>Input (Method)</b>	-
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. 사용자가 선택한 경로의 파일을 불러와서 정보를 각 클래스에 저장한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Save_work
<b>Purpose</b>	User 가 프로그램으로 작업 중이던 파일을 저장하도록 한다.
<b>Cross References</b>	Functions : R 1.2 Use Cases : Save File
<b>Input (Method)</b>	-
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. 사용자가 선택한 경로에 현재 작업 중이던 Task 정보를 파일로 저장한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Select_file
<b>Purpose</b>	파일 저장 혹은 불러오기 시, 파일의 저장 경로와 이름을 선택할 수 있도록 한다.
<b>Cross References</b>	Functions : R 1.1, R 1.2, R 6.3 Use Cases : Load File, Save File, Save Result
<b>Input (Method)</b>	-
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. 파일 경로를 GUI 를 통해 선택한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	New_category
<b>Purpose</b>	User 가 새로운 Category 를 생성할 수 있도록 한다.
<b>Cross References</b>	Functions : R 2.1 Use Cases : New Category
<b>Input (Method)</b>	Name : String
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. name 을 이름으로 갖는 카테고리를 생성한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Remove_category
<b>Purpose</b>	User 가 Category 를 삭제할 수 있도록 한다.
<b>Cross References</b>	Functions : R 2.2 Use Cases : Remove Category
<b>Input (Method)</b>	Cindex : int
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. Cindex 에 해당하는 Category 를 삭제한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Modify_category
<b>Purpose</b>	User 가 Category 정보를 수정할 수 있도록 한다.
<b>Cross References</b>	Functions : R 2.3 Use Cases : Modify Category
<b>Input (Method)</b>	Cindex : int, name : String
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. Cindex 에 해당하는 카테고리의 이름을 name 으로 수정한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	New_rep_value
<b>Purpose</b>	User 가 새로운 Rep. Value 를 생성할 수 있도록 한다.
<b>Cross References</b>	Functions : R 3.1 Use Cases : New Rep. Value
<b>Input (Method)</b>	Cindex : int
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. cindex 에 해당하는 카테고리에 new_rep_value 메소드를 실행한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Remove_rep_value
<b>Purpose</b>	User 가 Rep. Value 를 삭제할 수 있도록 한다.
<b>Cross References</b>	Functions : R 2.2, R 3.2 Use Cases : Remove Category, Remove Rep. Value
<b>Input (Method)</b>	Cindex : int, rindex : int
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. cindex 에 해당하는 카테고리의 remove_rep_value 메소드를 실행한다.
<b>Exceptional Courses of Events</b>	



<b>Type</b>	Method
<b>Name</b>	Modify_rep_value
<b>Purpose</b>	User 가 Rep. Value 의 정보를 수정할 수 있도록 한다.
<b>Cross References</b>	Functions : R 3.3 Use Cases : Modify Rep. Value
<b>Input (Method)</b>	Cindex : int, rindex : int, name : String
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. cindex 에 해당하는 카테고리의 modify_rep_value 메소드를 실행한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	New_const
<b>Purpose</b>	User 가 새로운 Constraints 를 생성할 수 있도록 한다.
<b>Cross References</b>	Functions : R 4.1 Use Cases : New Constraints
<b>Input (Method)</b>	Cindex : int, rindex : int
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. cindex, rindex 에 해당하는 Rep. Value 를 찾는다. 2. Constraints 에 add_const 메소드를 실행한다. 3. 해당하는 Rep. Value 의 add_const 메소드를 실행한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Remove_const
<b>Purpose</b>	User 가 Constraints 를 삭제할 수 있도록 한다.
<b>Cross References</b>	Functions : R 4.2 Use Cases : Remove Constraints
<b>Input (Method)</b>	Cindex : int, rindex : int, type : String
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. cindex, rindex 에 해당하는 Rep. Value 를 찾는다. 2. Constraints 에 remove_const 메소드를 실행한다. 3. 해당하는 Rep. Value 의 remove_const 메소드를 실행한다.
<b>Exceptional Courses of Events</b>	E1. Type 이 property / if_property 인 경우 2. Property 에 remove_const 메소드를 실행한다. 3. 해당하는 Rep. Value 의 remove_const 메소드를 실행한다.

<b>Type</b>	Method
<b>Name</b>	Modify_const
<b>Purpose</b>	User 가 Constraints 의 상태를 변경할 수 있도록 한다.
<b>Cross References</b>	Functions : R 4.3 Use Cases : Modify Constraints
<b>Input (Method)</b>	Cindex : int, rindex : int, oldpindex : int, oldtype : String, type : String, prop : Property
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. cindex, rindex 에 해당하는 Rep. Value 를 찾는다. 2. Constraints 에 remove_const 메소드를 실행한다. 3. Rep. Value 의 remove_const 메소드를 실행한다. 4. Constraints 에 add_const 메소드를 실행한다. 5. Rep. Value 의 add_const 메소드를 실행한다. 6. calculate 메소드를 실행한다.
<b>Exceptional Courses of Events</b>	E1. Oldtype 이 property / if_property 인 경우 2. Property 에 remove_const 메소드를 실행한다. 3. Rep. Value 의 remove_const 메소드를 실행한다. E2. Type 이 property / if_property 인 경우 4. Property 에 add_const 메소드를 실행한다. 5. Rep. Value 의 add_const 메소드를 실행한다.

<b>Type</b>	Method
<b>Name</b>	New_property
<b>Purpose</b>	User 가 새로운 Property 를 생성할 수 있도록 한다.
<b>Cross References</b>	Functions : R 5.1 Use Cases : New Property
<b>Input (Method)</b>	Name : String
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. name 을 이름으로 갖는 Property 를 생성한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Remove_property
<b>Purpose</b>	User 가 Property 를 삭제할 수 있도록 한다.
<b>Cross References</b>	Functions : R 2.2, R 5.2 Use Cases : Remove Category, Remove Property
<b>Input (Method)</b>	Pindex : int
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. pindex 에 해당하는 Property 를 찾는다. 2. 해당 Property 를 갖고 있는 Rep. Value 들에 대해서 모두 해당 정보를 삭제한다. 3. calculate 메소드를 실행한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Modify_property
<b>Purpose</b>	User 가 Property 정보를 변경할 수 있도록 한다.
<b>Cross References</b>	Functions : R 5.3 Use Cases : Modify Property
<b>Input (Method)</b>	Pindex : int, name : String
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. pindex 에 해당하는 Property 를 찾는다. 2. 해당 Property 의 이름을 name 으로 변경한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Calculate
<b>Purpose</b>	입력된 정보를 바탕으로 Test Case 개수를 계산한다.
<b>Cross References</b>	Functions : R 2.2, R 3.1, R 3.2, R 4.1, R 4.2, R 4.3, R 5.2, R 6.1, R 6.2 Use Cases : Remove Category, New Rep. Value, Remove Rep. Value, New Constraints, Remove Constraints, Modify Constraints, Remove Property, Calculate, Show Result
<b>Input (Method)</b>	-
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. Task 에 저장된 정보를 바탕으로 Test Case 를 계산한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Combination()
<b>Purpose</b>	입력된 정보를 바탕으로 Test Case 조합 결과를 생성한다.
<b>Cross References</b>	Functions : R 6.1, R 6.2 Use Cases : Calculate, Show Result
<b>Input (Method)</b>	-
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. Task 에 저장된 정보를 바탕으로 Test Case 조합 결과를 생성한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Filtering
<b>Purpose</b>	User 가 체크한 filter 를 바탕으로 Test Case 조합 결과를 보여준다.
<b>Cross References</b>	Functions : R 6.2.1, R 6.2.2 Use Cases : Filter Rep. Value, Filter Property
<b>Input (Method)</b>	-
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. Task 에 저장된 Rep. Value, Property 들의 filter 여부를 확인해서 result List 를 변경한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Check_filter
<b>Purpose</b>	User 가 Window2 에서 필터링할 Rep. Value 혹은 Property 를 체크할 수 있도록 한다.
<b>Cross References</b>	Functions : R 6.2.1, R 6.2.2 Use Cases : Filter Rep. Value, Filter Property
<b>Input (Method)</b>	Type : String, cindex : int, rindex : int, pindex : int
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. cindex, rindex 에 해당하는 Rep. Value 를 찾는다. 2. 해당 Rep. Value 의 mark_filter 메소드를 실행한다. 3. Result 의 make_filter_list 메소드를 실행한다.
<b>Exceptional Courses of Events</b>	E1. Type 이 Property 인 경우 1. pindex 에 해당하는 Property 를 찾는다. 2. 해당 Property 의 mark_filter 메소드를 실행한다. ...

<b>Type</b>	Method
<b>Name</b>	Save_result
<b>Purpose</b>	User 가 Testing 결과를 저장할 수 있도록 한다.
<b>Cross References</b>	Functions : R 6.3 Use Cases : Save Result
<b>Input (Method)</b>	-
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. 사용자가 선택한 경로에 결과 데이터를 파일로 저장한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	ProvideHelp
<b>Purpose</b>	User 에게 도움말 화면을 제공한다.
<b>Cross References</b>	Functions : R 6.4 Use Cases : Provide Help
<b>Input (Method)</b>	-
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. User 에게 도움말 화면을 보여준다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Check_true_false
<b>Purpose</b>	User 가 Test Case 조합에 대해 테스트 결과를 입력할 수 있도록 한다.
<b>Cross References</b>	Functions : R 7.1 Use Cases : Check True/False
<b>Input (Method)</b>	Index : int
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. index 에 해당하는 Result 를 찾는다. 2. 해당 Result 의 check 메소드를 실행한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	New_rep_value
<b>Purpose</b>	해당하는 Category 에 하위 Rep. Value 를 생성한다.
<b>Cross References</b>	Functions : R 3.1 Use Cases : New Rep. Value
<b>Input (Method)</b>	-
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. 새로운 Rep. Value 를 생성한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Remove_rep_value
<b>Purpose</b>	해당하는 Category 의 하위 Rep. Value 를 삭제한다.
<b>Cross References</b>	Functions : R 3.2 Use Cases : Remove Rep. Value
<b>Input (Method)</b>	Rindex : int
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. rindex 에 해당하는 Rep. Value 를 삭제한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Modify_rep_value
<b>Purpose</b>	해당하는 Category 의 하위 Rep. Value 의 정보를 변경한다.
<b>Cross References</b>	Functions : R 3.3 Use Cases : Modify Rep. Value
<b>Input (Method)</b>	Rindex : int, name : String
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. rindex 에 해당하는 Rep. Value 의 이름을 name 으로 수정한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Add_const – Constraints
<b>Purpose</b>	Rep. Value 에 Constraints 를 새로 입력하거나 변경할 때 Constraints 에 해당 Rep. Value 를 저장한다.
<b>Cross References</b>	Functions : R 4.1, R 4.3 Use Cases : New Constraints, Modify Constraints
<b>Input (Method)</b>	Repv : RepValue, type : String
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. type 에 해당하는 List 에 repv 를 추가한다.
<b>Exceptional Courses of Events</b>	N/A



<b>Type</b>	Method
<b>Name</b>	Remove_const – Constraints
<b>Purpose</b>	Rep. Value 에 Constraints 를 변경하거나 삭제할 때, 혹은 Rep. Value 를 삭제할 경우 Constraints 에 해당 Rep. Value 를 삭제한다.
<b>Cross References</b>	Functions : R 3.2, R 4.1, R 4.3 Use Cases : Remove Rep. Value, Remove Constraints, Modify Constraints
<b>Input (Method)</b>	Repv : RepValue, oldtype : String
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. oldtype 에 해당하는 List 에서 repv 를 삭제한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Add_const – Property
<b>Purpose</b>	Rep. Value 에 Property 가 할당된 경우, Property 에 해당 Rep. Value 를 저장한다.
<b>Cross References</b>	Functions : R 4.3 Use Cases : Modify Constraints
<b>Input (Method)</b>	Repv : RepValue, type : String
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. type 에 해당하는 List 에 repv 를 추가한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Remove_const - Property
<b>Purpose</b>	Rep. Value 에 할당되어 있던 Property 가 변경 혹은 삭제된 경우, Property 에 해당 Rep. Value 에 대한 정보를 삭제한다.
<b>Cross References</b>	Functions : R 3.2, R 4.2, R 4.3 Use Cases : Remove Rep. Value, Remove Constraints, Modify Constraints
<b>Input (Method)</b>	Repv : RepValue, oldtype : String
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. oldtype 에 해당하는 List 에서 repv 를 삭제한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Mark_filter - Property
<b>Purpose</b>	User 가 체크한 Filter 를 Property 에 적용한다.
<b>Cross References</b>	Functions : R 6.2.2 Use Cases : Filter Property
<b>Input (Method)</b>	-
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. filter 값을 변경시킨다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	Add_const – RepValue
<b>Purpose</b>	Rep. Value 에 Property 가 할당된 경우, Rep. Value 에 해당 Property 를 저장한다.
<b>Cross References</b>	Functions : R 4.3 Use Cases : Modify Constraints
<b>Input (Method)</b>	(Repv : RepValue), type : String
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. type 에 해당하는 cst 값을 true 로 변경한다.
<b>Exceptional Courses of Events</b>	E1. Repv 인자가 같이 주어진 경우 1. type 에 해당하는 List 에 해당 Property 를 추가한다.

<b>Type</b>	Method
<b>Name</b>	Remove_const - RepValue
<b>Purpose</b>	Rep. Value 에 할당되어 있던 Property 가 변경 혹은 삭제된 경우, Rep. Value 에 해당 Property 를 삭제한다.
<b>Cross References</b>	Functions : R 3.2, R 4.2, R 4.3 Use Cases : Remove Rep. Value, Remove Constraints, Modify Constraints
<b>Input (Method)</b>	(Repv : RepValue), type : String
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. type 에 해당하는 cst 값을 false 로 변경한다.
<b>Exceptional Courses of Events</b>	E1. Repv 인자가 같이 주어진 경우 1. type 에 해당하는 List 에서 해당 Property 를 삭제한다.

<b>Type</b>	Method
<b>Name</b>	Mark_filter - RepValue
<b>Purpose</b>	User 가 체크한 Filter 를 Rep Value 에 적용한다.
<b>Cross References</b>	Functions : R 6.2.1 Use Cases : Filter Rep. Value
<b>Input (Method)</b>	-
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. filter 값을 변경시킨다.
<b>Exceptional Courses of Events</b>	N/A

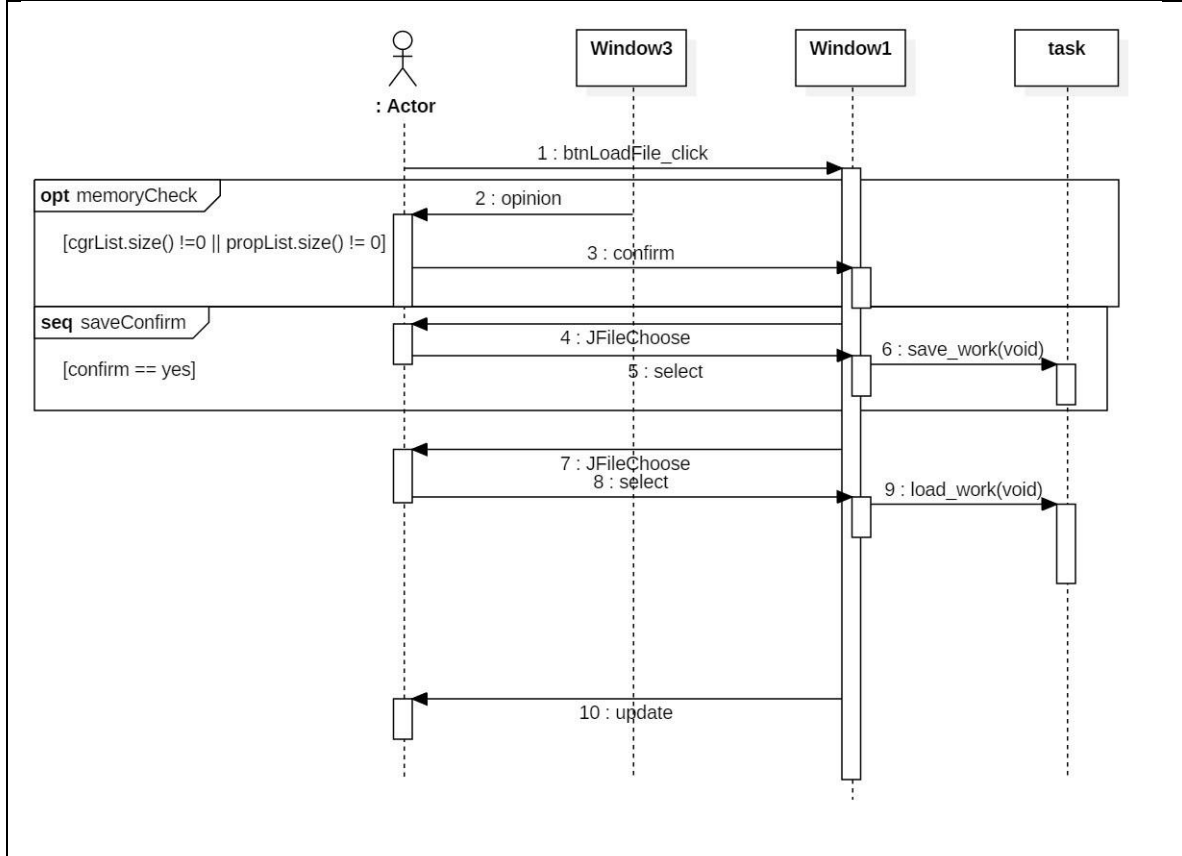
<b>Type</b>	Method
<b>Name</b>	isCompatible
<b>Purpose</b>	Rep. Value 에 Constraints 를 추가 혹은 변경할 때 정상적인 입력인 지를 확인한다.
<b>Cross References</b>	Functions : R 4.1, R 4.3 Use Cases : New Constraints, Modify Constraints
<b>Input (Method)</b>	-
<b>Output (Method)</b>	Boolean
<b>Abstract Operation (Method)</b>	1. 현재 입력된 constraints 데이터와 새로 입력된 constraints 데이터가 충돌하는 지 여부를 검사한다. 2. 충돌 여부를 리턴한다.
<b>Exceptional Courses of Events</b>	N/A

<b>Type</b>	Method
<b>Name</b>	New_resultList
<b>Purpose</b>	Test Case 계산할 때, 새로운 Test Case 조합 결과를 Result 에 저장한다.
<b>Cross References</b>	Functions : R 6.1, R 6.2 Use Cases : Calculate, Show Result
<b>Input (Method)</b>	Repv : RepValue, prop : Property
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. repv 혹은 prop 을 해당하는 List 에 추가한다.
<b>Exceptional Courses of Events</b>	N/A

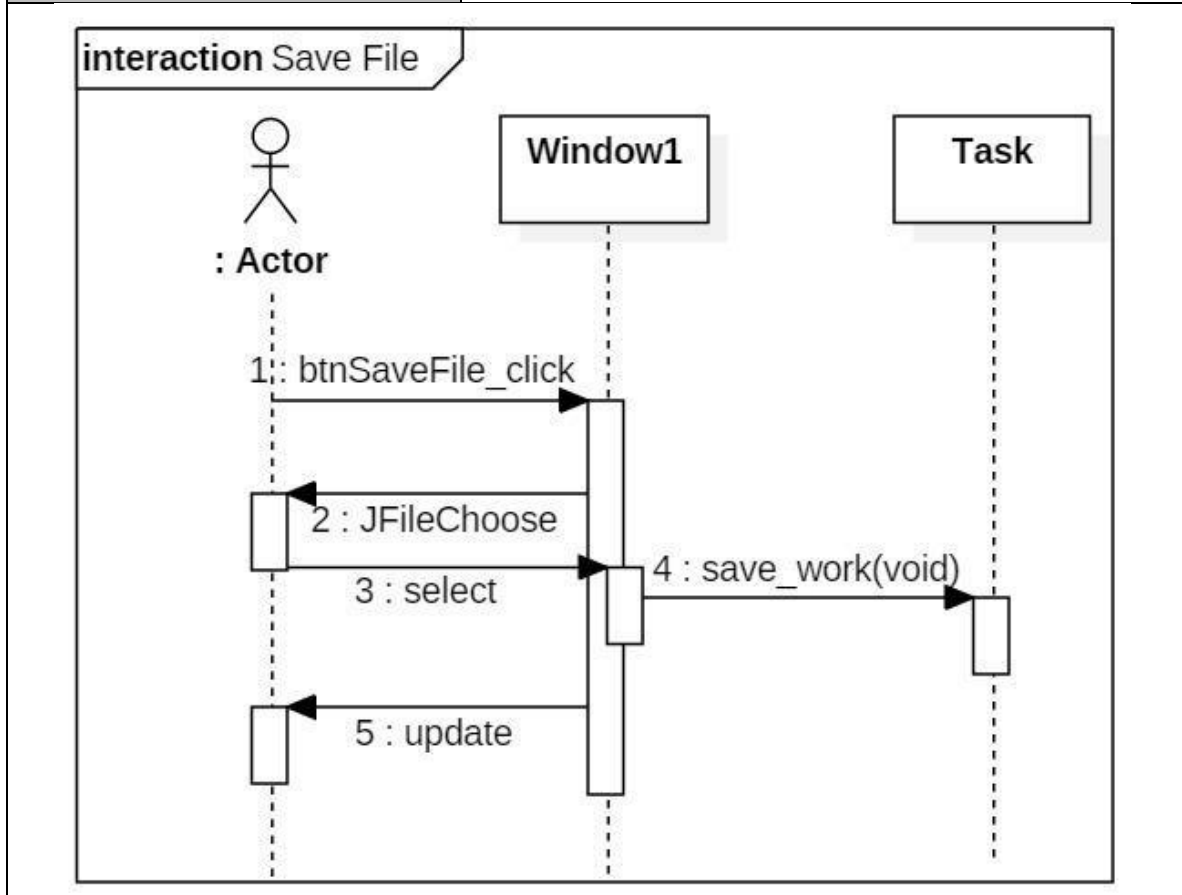
<b>Type</b>	Method
<b>Name</b>	Check
<b>Purpose</b>	User 가 Test Case 수행 결과를 저장할 수 있도록 한다.
<b>Cross References</b>	Functions : R 7.1 Use Cases : Check True/False
<b>Input (Method)</b>	-
<b>Output (Method)</b>	-
<b>Abstract Operation (Method)</b>	1. caseFlag 값을 변경시킨다.
<b>Exceptional Courses of Events</b>	N/A

**Activity 2052. Implement Windows**

<b>Name</b>	BtnLoadFile_click
<b>Responsibilities</b>	Load File 버튼을 누른다.
<b>Type</b>	GUI
<b>Cross References</b>	Functions : R 1.1 Use Cases : Load File
<b>Notes</b>	Load File 버튼을 눌러 저장할 파일을 선택 후 load_work 메소드를 호출한다.
<b>Pre-Conditions</b>	-
<b>Post-Conditions</b>	선택한 파일의 내용을 메모리로 불러온다.



<b>Name</b>	btnSaveFile_click
<b>Responsibilities</b>	Save File 버튼을 누른다.
<b>Type</b>	GUI
<b>Cross References</b>	Functions : R 1.2 Use Cases : Save File
<b>Notes</b>	Save File 버튼을 눌러 저장할 파일을 선택 후 Save_work 메소드를 호출한다.
<b>Pre-Conditions</b>	-
<b>Post-Conditions</b>	선택한 파일에 메모리 내용을 기록한다.



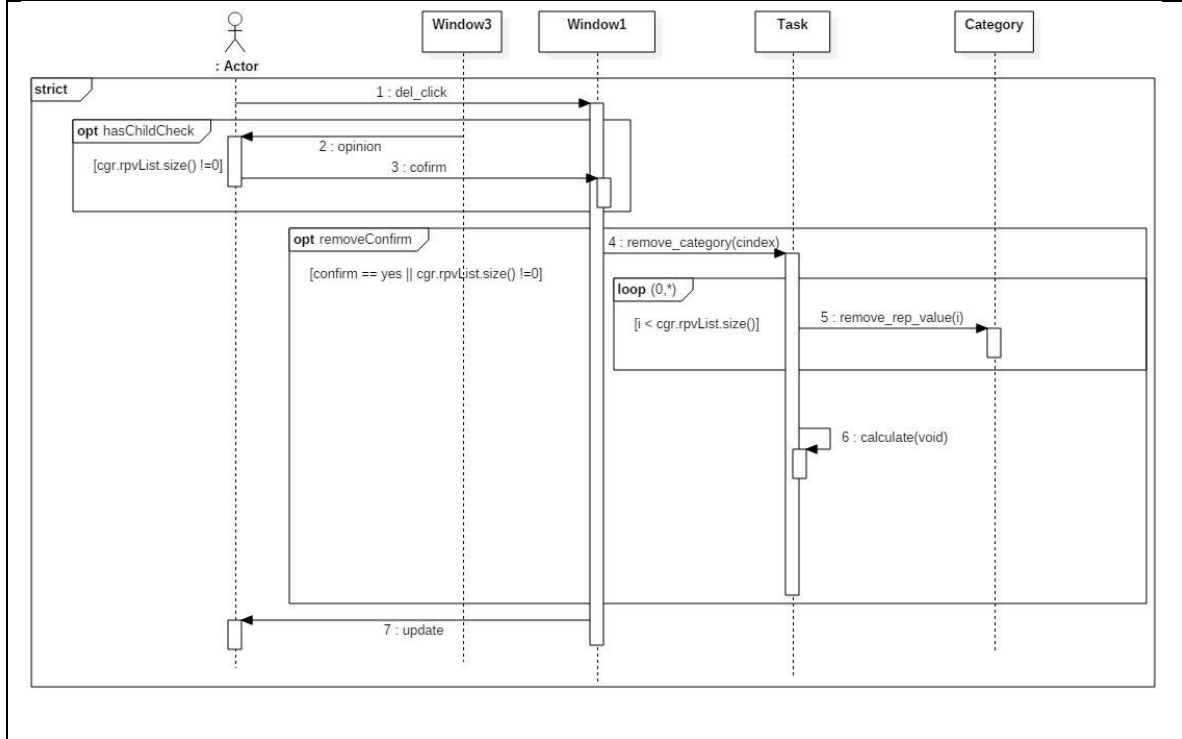
---

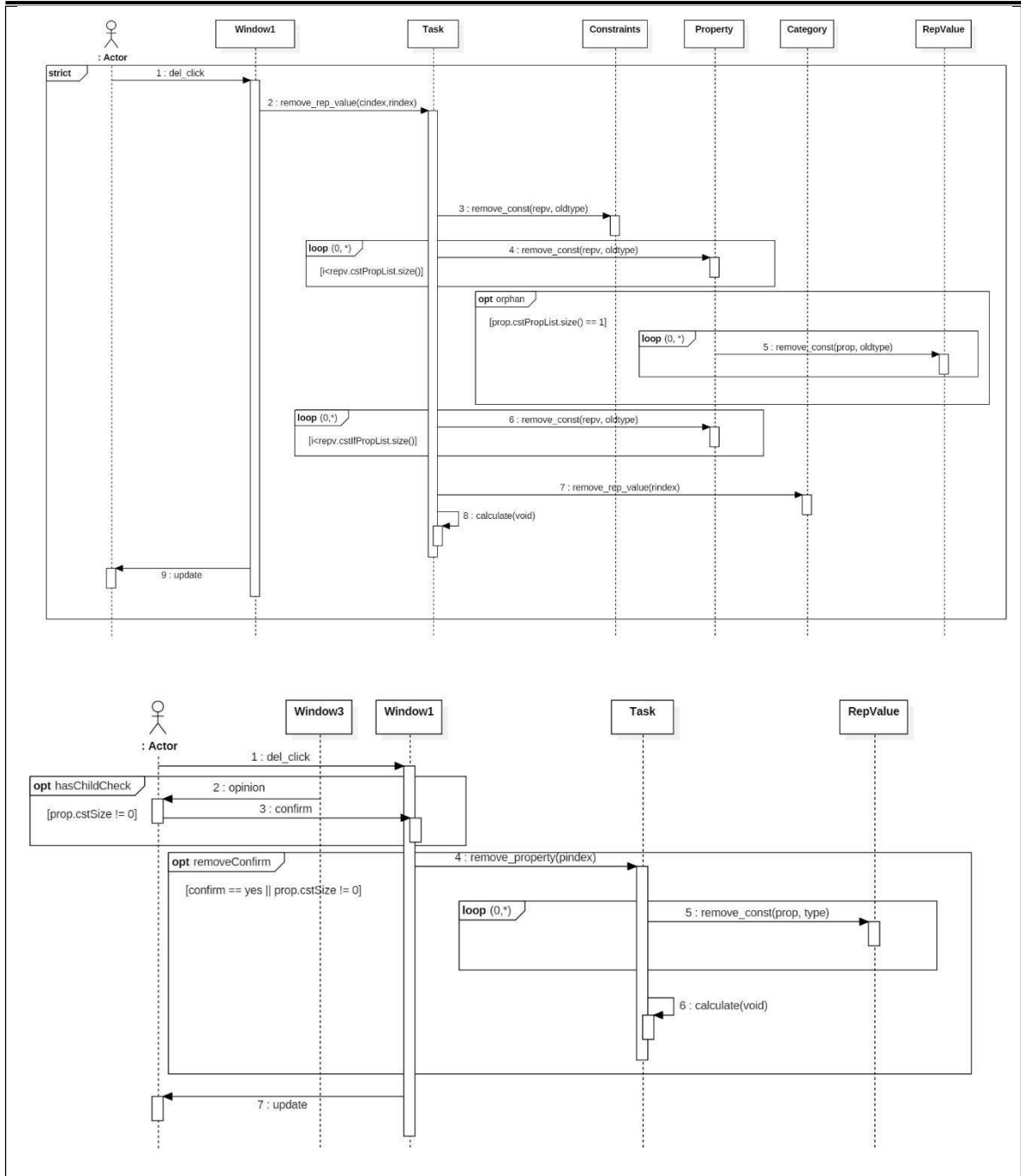
- ComponentSet 의 GUI 컴포넌트

<b>Name</b>	Sel_click
<b>Responsibilities</b>	객체의 이름 옆의 체크모양 버튼을 누른다.
<b>Type</b>	GUI
<b>Cross References</b>	-
<b>Notes</b>	객체를 선택하여 이 객체가 공유하는 다른 관계에 대해 추가/제거 및 수정이 가능하다.
<b>Pre-Conditions</b>	-
<b>Post-Conditions</b>	객체가 선택된다.

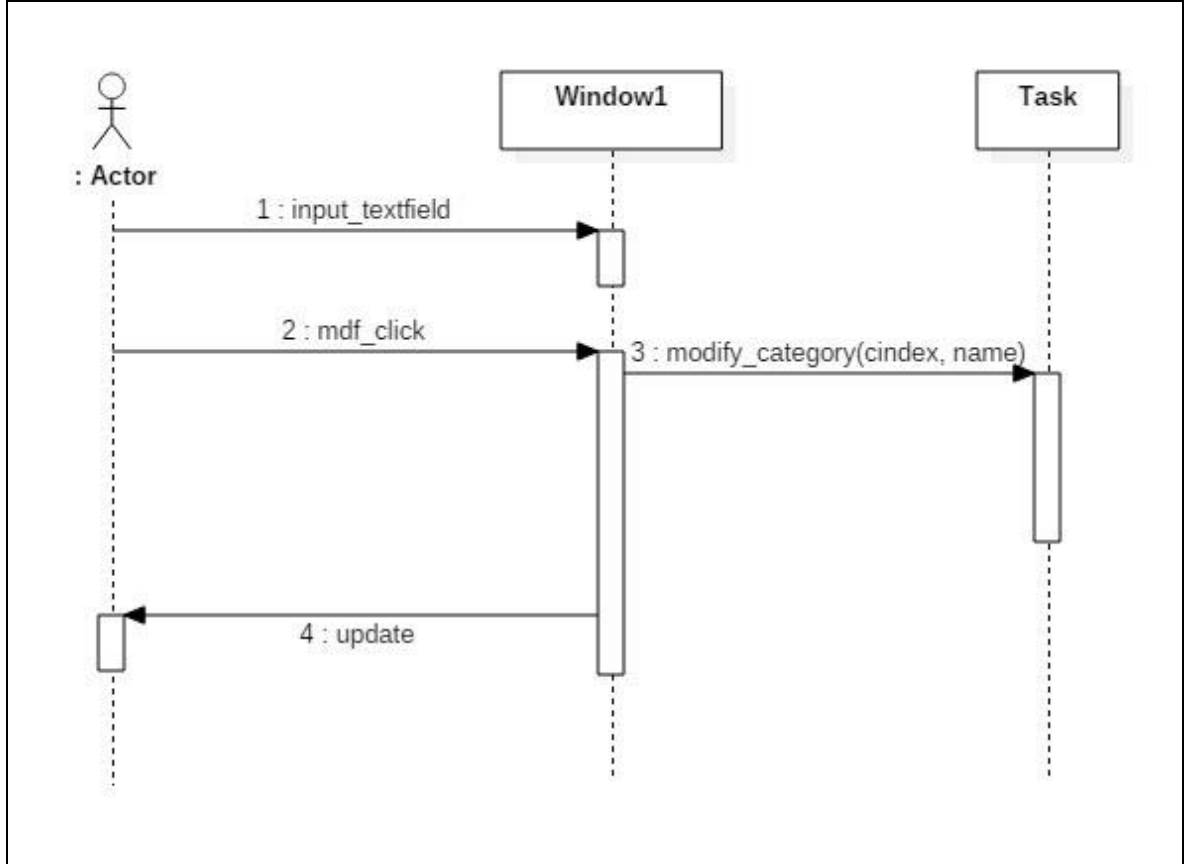


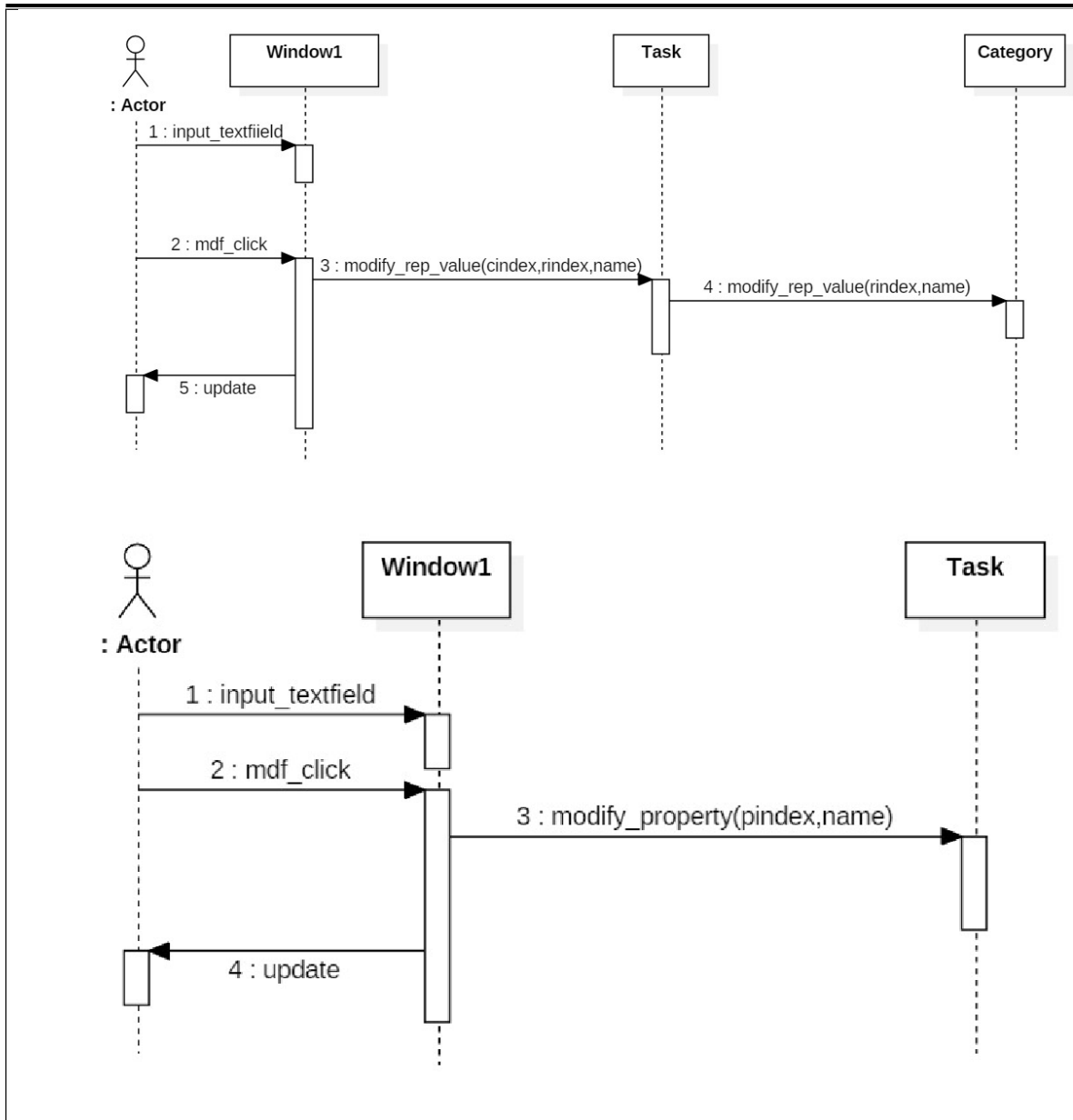
<b>Name</b>	Del_click
<b>Responsibilities</b>	객체의 이름 옆의 휴지통 버튼을 누른다.
<b>Type</b>	GUI
<b>Cross References</b>	Functions : R 2.2, R 3.2, R 5.2 Use Cases : Remove Category, Remove Rep. Value, Remove Property
<b>Notes</b>	객체를 제거한다.
<b>Pre-Conditions</b>	-
<b>Post-Conditions</b>	객체가 리스트에서 제거된다. 다른 객체의 관계도 변경된다면 이에 대한 의사확인 후 제거된다.



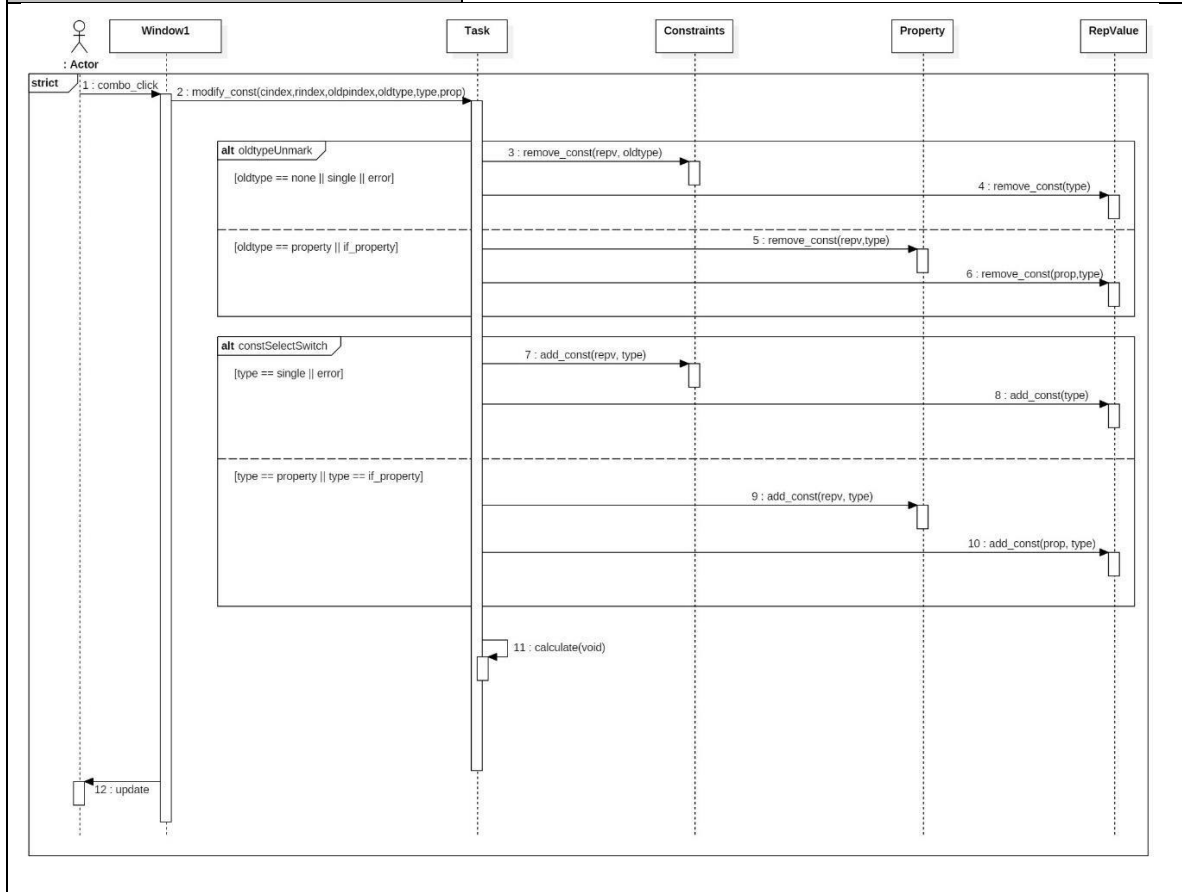


<b>Name</b>	Mdf_click
<b>Responsibilities</b>	객체 옆의 렌치 모양 버튼을 누른다.
<b>Type</b>	GUI
<b>Cross References</b>	Functions : R 2.3, R 3.3, R 5.3 Use Cases : Modify Category, Modify Rep. Value, Modify Property
<b>Notes</b>	객체의 이름을 수정한다.
<b>Pre-Conditions</b>	-
<b>Post-Conditions</b>	현재 textField 에 입력된 이름으로 객체의 이름을 수정한다.



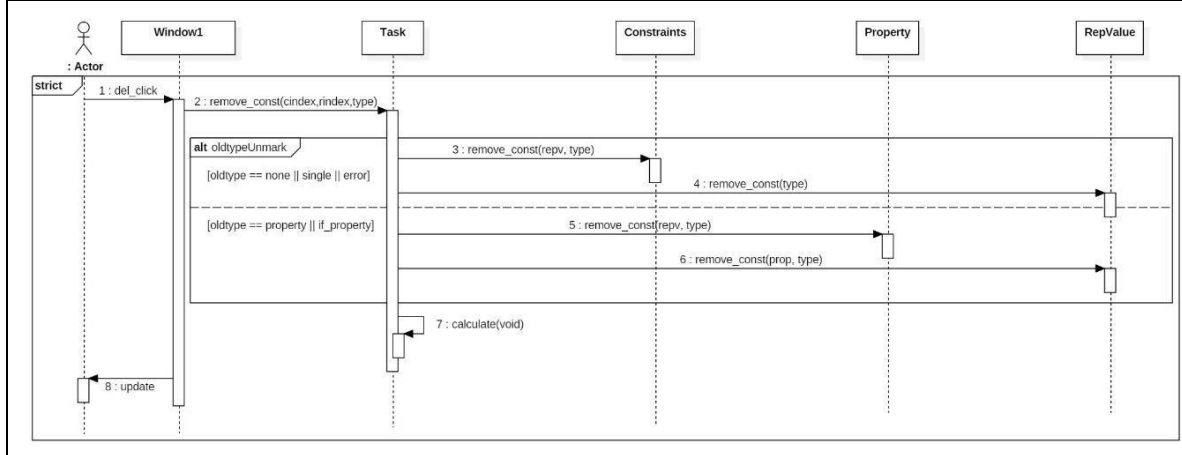


<b>Name</b>	combo_click
<b>Responsibilities</b>	제약관계 이름 옆의 콤보 버튼을 조작한다.
<b>Type</b>	GUI
<b>Cross References</b>	Functions : R 4.3 Use Cases : Modify Constraints
<b>Notes</b>	제약 관계를 결정한다.
<b>Pre-Conditions</b>	-
<b>Post-Conditions</b>	기존의 제약 관계가 제거되고 선택한 제약 관계가 새로 생성된다.

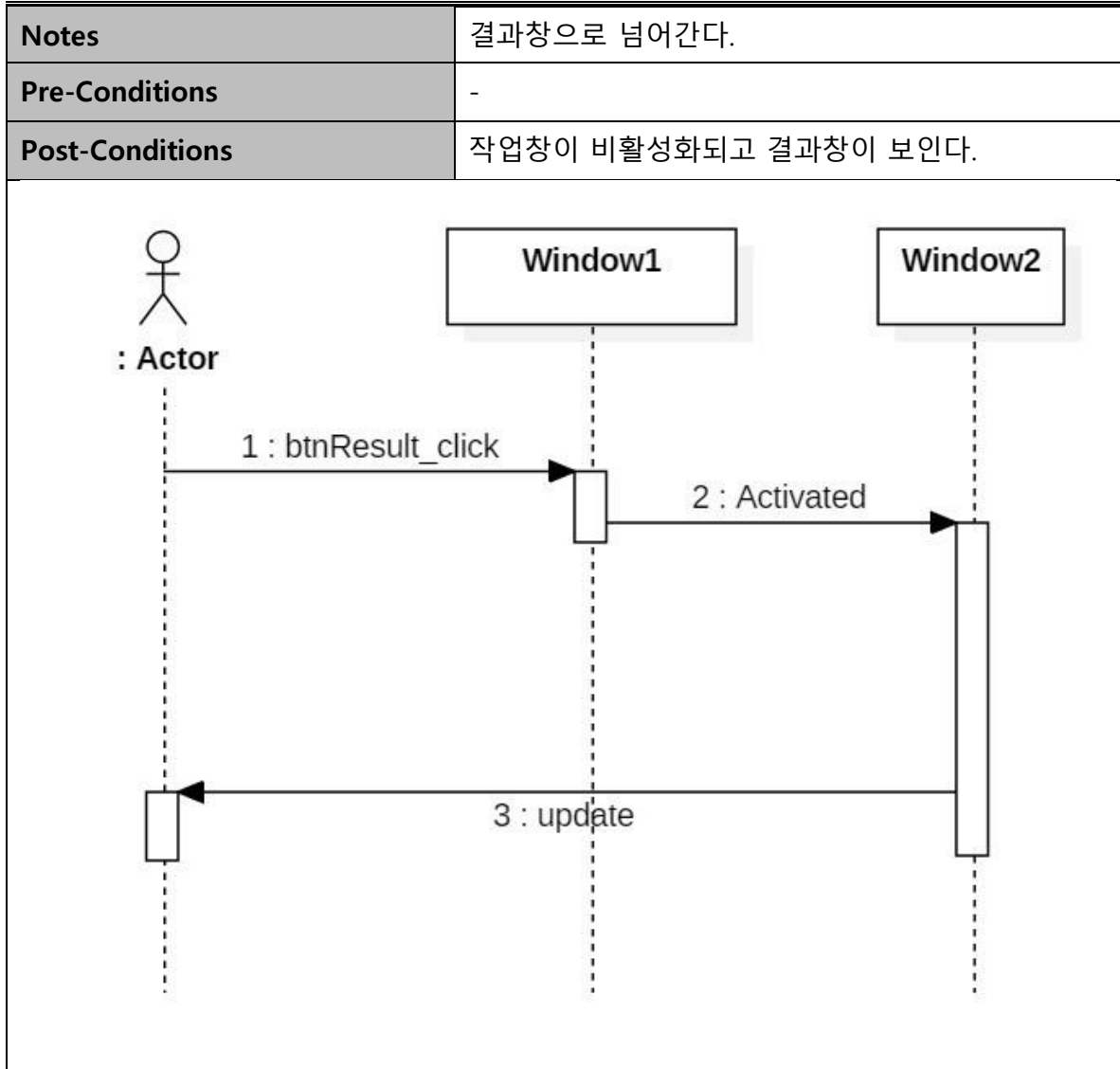


<b>Name</b>	Sel
<b>Responsibilities</b>	제약 관계 이름 옆의 체크 모양 버튼을 누른다.
<b>Type</b>	GUI
<b>Cross References</b>	Functions : - Use Cases : -
<b>Notes</b>	구체적으로 어떤 Property 와 연결되는지 설정할 제약관계를 선택한다.
<b>Pre-Conditions</b>	제약 관계가 Property, IF_Property 이다.
<b>Post-Conditions</b>	제약 관계가 선택된다.

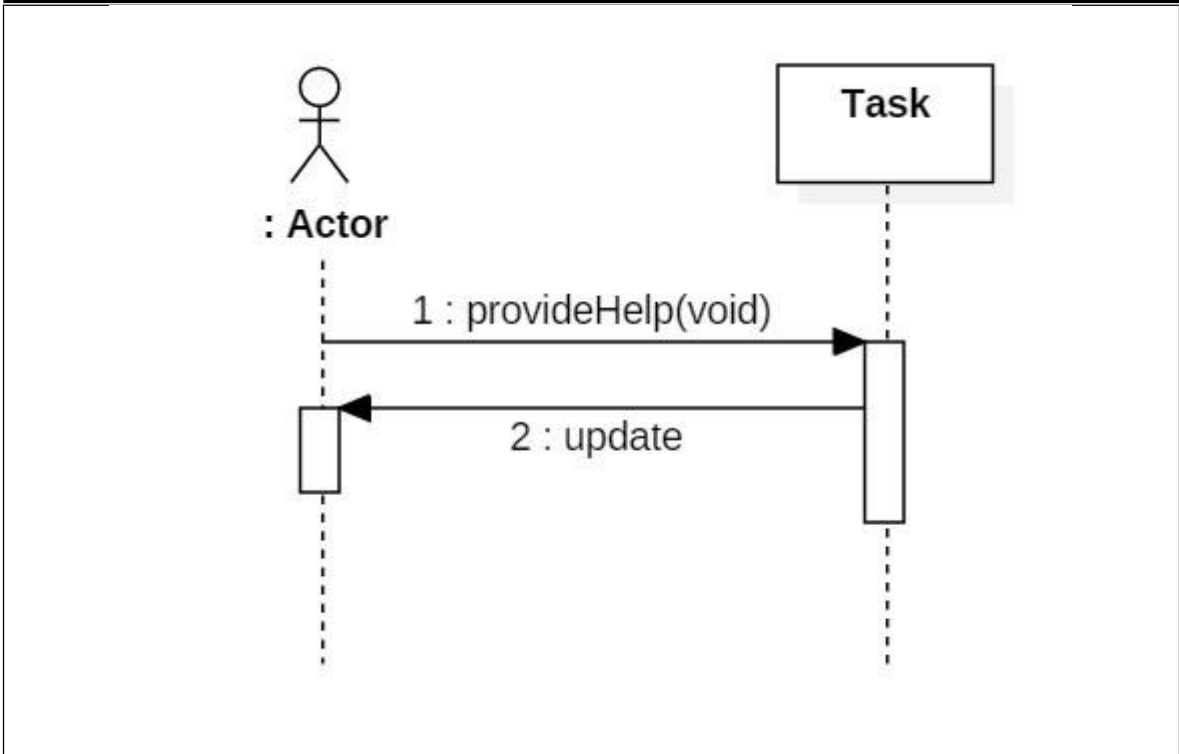
<b>Name</b>	Del
<b>Responsibilities</b>	제약 관계 이름 옆의 휴지통 모양 버튼을 누른다.
<b>Type</b>	GUI
<b>Cross References</b>	Functions : R 4.2 Use Cases : Remove Constraints
<b>Notes</b>	제약 관계를 제거한다.
<b>Pre-Conditions</b>	-
<b>Post-Conditions</b>	제약 관계가 제거된다.



<b>Name</b>	btnResult_click
<b>Responsibilities</b>	RESULT 라 적힌 버튼을 누른다.
<b>Type</b>	GUI
<b>Cross References</b>	Functions : R 6.2 Use Cases : Show Result



<b>Name</b>	provideHelp_click
<b>Responsibilities</b>	HELP 라 적힌 버튼을 누른다.
<b>Type</b>	GUI
<b>Cross References</b>	Functions : R 6.4 Use Cases : Provide Help
<b>Notes</b>	결과창으로 넘어간다.
<b>Pre-Conditions</b>	-
<b>Post-Conditions</b>	작업창이 비활성화되고 결과창이 보인다.





---

**Activity 2055. Write Unit Test Code****- Category\_Test**

```
@Test
public void testNew_rep_value() {
    Category category = new Category("test",0);
    for(int i=0;i<100;i++)
    {
        category.new_rep_value();
        assertEquals(i+1,category.getRpvListSize());
    }
    //fail("Not yet implemented");
}

@Test
public void testRemove_rep_value() {
    Category category = new Category("test",0);
    for(int i=0;i<100;i++)
        category.new_rep_value();

    for(int i=0;i<100;i++)
    {
        category.remove_rep_value(0);
    }
    assertEquals(0,category.getRpvListSize());
    //fail("Not yet implemented");
}

@Test
public void testModify_rep_value() {
    Category category = new Category("test",0);
    for(int i=0;i<100;i++)
        category.new_rep_value();
    for(int i=0;i<100;i++)
        category.modify_rep_value(i, "test" + i);
    for(int i=0;i<100;i++)
```

---

```
        assertEquals("test" + i,category.getRepValue(i).getName());
```

```
    }
```

#### - Property\_Test

```
@Test
```

```
public void testAdd_const() {
    Property ptemp = new Property("test");
    RepValue rtemp = new RepValue(0,0);
    ptemp.add_const(rtemp, "Property");
    ptemp.add_const(rtemp, "IFProperty");
    assertEquals(rtemp,ptemp.getConst(0, "Property"));
    assertEquals(rtemp,ptemp.getConst(0, "IFProperty"));
    //fail("Not yet implemented");
}
```

```
@Test
```

```
public void testRemove_const() {
    //fail("Not yet implemented");
    Property ptemp = new Property("test");
    RepValue rtemp = new RepValue(0,0);
    ptemp.add_const(rtemp, "Property");
    ptemp.add_const(rtemp, "IFProperty");
    ptemp.remove_const(rtemp, "Property");
    ptemp.remove_const(rtemp, "IFProperty");
    assertEquals(0,ptemp.getCstSize("Property"));
    assertEquals(0,ptemp.getCstSize("IFProperty"));
}
```

```
@Test
```

```
public void testMark_filter() {
    //fail("Not yet implemented");
    Property ptemp = new Property("test");
    ptemp.mark_filter();
    assertEquals(true,ptemp.isFilter());
    ptemp.mark_filter();
    assertEquals(false,ptemp.isFilter());
}
```

---

**- RepValue\_Test**

```
@Test
public void testAdd_const1() {
    RepValue rtest = new RepValue(0,0);
    rtest.add_const("None");
    rtest.add_const("Error");
    rtest.add_const("Single");

    assertEquals(true,rtest.getConst("None"));

    //fail("Not yet implemented");
}

@Test
public void testAdd_const2() {
    RepValue rtest = new RepValue(0,0);
    Property ptest = new Property("test");
    rtest.add_const(ptest, "Property");
    rtest.add_const(ptest,"IFProperty");
    assertEquals(ptest,rtest.getConst(0, "Property"));
    assertEquals(ptest,rtest.getConst(0, "IFProperty"));

    //fail("Not yet implemented");
}

@Test
public void testRemove_const1() { RepValue
    rtest = new RepValue(0,0);
    rtest.add_const("None");
    rtest.add_const("Error");
    rtest.add_const("Single");
    rtest.remove_const("None");
    rtest.remove_const("Error");
    rtest.remove_const("Single");
    assertEquals(false,rtest.getConst("None"));
    assertEquals(false,rtest.getConst("Error"));
```

```
        assertEquals(false,rtest.getConst("Single"));
        //fail("Not yet implemented");
    }
```

```
@Test
public void testRemove_const2()
    { RepValue rtest = new
      RepValue(0,0);
      Property ptest = new Property("test");
      rtest.add_const(ptest, "Property");
      rtest.add_const(ptest,"IFProperty");
      rtest.remove_const(ptest, "Property");
      rtest.remove_const(ptest, "IFProperty");
      assertEquals(0,rtest.getCstSize("Property"));
      assertEquals(0,rtest.getCstSize("IFProperty"));
      //fail("Not yet implemented");
    }
```

```
@Test
public void testMark_filter() {
    RepValue rtest = new RepValue(0,0);
    rtest.mark_filter();
    assertEquals(true,rtest.isFilter());
    rtest.mark_filter();
    assertEquals(false,rtest.isFilter());
    //fail("Not yet implemented");
}
```

#### - Task\_Test

```
@Test
public void testNew_category()
    { Task task = new Task();
      for(int i=0;i<100;i++)
      {
          task.new_category("test" +i);
          assertEquals(i+1,task.getCgrListSize());
      }
      //fail("Not yet implemented");
    }
```

---

```
}

@Test
public void testRemove_category() {
    //fail("Not yet implemented");
    Task task = new Task();
    for(int i=0;i<100;i++)
    {
        task.new_category("test"+i);
    }
    for(int i=0;i<100;i++)
    {
        task.remove_category(0);
    }
    assertEquals(0,task.getCgrListSize());
}

@Test
public void testModify_category() {
    //fail("Not yet implemented");
    Task task = new Task();

    for(int i=0;i<100;i++)
        task.new_category("test"+i);
    for(int i=0;i<100;i++)
        task.modify_category(i, "itest"+i);
    for(int i=0;i<100;i++)
        assertEquals("itest" + i,task.getCategory(i).getName());
}

@Test
public void testNew_property() {
    //fail("Not yet implemented");
    Task task = new Task();
    for(int i=0;i<100;i++)
    {
```

```
        task.new_property("test"+i);
        assertEquals(i+1,task.getPropListSize());
    }
}

@Test
public void testRemove_property() {
    //fail("Not yet implemented");
    Task task = new Task();
    for(int i=0;i<100;i++)
    {
        task.new_property("test"+i);
    }
    for(int i=0;i<100;i++)
    {
        task.remove_property(0);
    }
    assertEquals(0,task.getPropListSize());
}

@Test
public void testModify_property() {
    //fail("Not yet implemented");
    Task task = new Task();

    for(int i=0;i<100;i++)
        task.new_property("test"+i);
    for(int i=0;i<100;i++)
        task.modify_property(i, "itest"+i);
    for(int i=0;i<100;i++)
        assertEquals("itest" + i,task.getProperty(i).getName());
}
}
```

## Activity 2060. Test

### Activity 2061. Unit Testing

#### - Category\_Test

The screenshot shows an IDE with the following content:

```

1 package TEST;
2
3 import static org.junit.Assert.*;
4
5 public class Category_test {
6
7     @Test
8     public void testNew_rep_value() {
9         Category category = new Category("test",0);
10        for(int i=0;i<100;i++)
11        {
12            category.new_rep_value();
13            assertEquals(i+1,category.getRpvListSize());
14        }
15        //fail("Not yet implemented");
16    }
17
18     @Test
19     public void testRemove_rep_value() {
20         Category category = new Category("test",0);
21         for(int i=0;i<100;i++)
22             category.new_rep_value();
23
24         for(int i=0;i<100;i++)
25             category.remove_rep_value(i);
26     }
27 }

```

The Outline view on the right shows the test methods: testNew\_rep\_value(), testRemove\_rep\_value(), and testModify\_rep\_value(). The Console view at the bottom shows the execution results:

```

Category_test
Runs: 3/3      Errors: 0      Failures: 0
- TEST.Category_test [Runner: JUnit 4] (0.008 s)
  - testNew_rep_value (0.003 s)
  - testModify_rep_value (0.004 s)
  - testRemove_rep_value (0.002 s)

```

#### - Property\_Test

The screenshot shows an IDE with the following content:

```

1 package TEST;
2
3 import static org.junit.Assert.assertEquals;
4
5 public class Property_testTest {
6
7     @Test
8     public void testAdd_const() {
9         Property ptemp = new Property("test");
10        RepValue rtemp = new RepValue(0,0);
11        ptemp.add_const(rtemp, "Property");
12        ptemp.add_const(rtemp, "IFProperty");
13        assertEquals(rtemp,ptemp.getConst(0, "Property"));
14        assertEquals(rtemp,ptemp.getConst(0, "IFProperty"));
15        //fail("Not yet implemented");
16    }
17
18     @Test
19     public void testRemove_const() {
20        //fail("Not yet implemented");
21        Property ptemp = new Property("test");
22        RepValue rtemp = new RepValue(0,0);
23        ptemp.add_const(rtemp, "Property");
24        ptemp.add_const(rtemp, "IFProperty");
25        ptemp.remove_const(rtemp, "Property");
26    }
27 }

```

The Outline view on the right shows the test methods: testAdd\_const(), testRemove\_const(), and testMark\_filter(). The Console view at the bottom shows the execution results:

```

Finished after 0.028 seconds
Property_testTest
Runs: 3/3      Errors: 0      Failures: 0
- TEST.Property_testTest [Runner: JUnit 4] (0.000 s)

```

## - RepValue\_Test

```

1 package TEST;
2
3 import static org.junit.Assert.assertEquals;
4
5
6
7
8
9
10
11 public class RepValue_test {
12
13     @Test
14     public void testAdd_const1() {
15         RepValue rtest = new RepValue(0,0);
16         rtest.add_const("None");
17         rtest.add_const("Error");
18         rtest.add_const("Single");
19
20         assertEquals(true,rtest.getConst("None"));
21
22         //fail("Not yet implemented");
23     }
24
25     @Test
26     public void testAdd_const2() {
27         RepValue rtest = new RepValue(0,0);
28         Property ptest = new Property("test");
29         rtest.add_const(ptest, "Property");
30         assertEquals(ptest,rtest.getConst(0, "Property"));
31     }
32 }

```

Finished after 0.021 seconds

Runs: 5/5      \* Errors: 0      \* Failures: 0

TEST.RepValue\_test [Runner: JUnit 4] (0.001 s)

## - Task\_Test

```

1 package TEST;
2
3 import static org.junit.Assert.*;
4
5
6
7
8
9
10
11 public class Task_test {
12
13     @Test
14     public void testNew_category() {
15         Task task = new Task();
16         for(int i=0;i<100;i++)
17         {
18             task.new_category("test"+i);
19             assertEquals(i+1,task.getCgrListSize());
20         }
21         //fail("Not yet implemented");
22     }
23
24     @Test
25     public void testRemove_category() {
26         //fail("Not yet implemented");
27         Task task = new Task();
28         for(int i=0;i<100;i++)
29         {
30             task.new_category("test"+i);
31         }
32     }
33 }

```

Finished after 0.032 seconds

Runs: 6/6      \* Errors: 0      \* Failures: 0

TEST.Task\_test [Runner: JUnit 4] (0.008 s)



**Activity 2063. System Testing**

Test Number	Test 항목	Description	Use Case	System Function	Pass / Fail
1.1	작업 파일 불러오기 시험	작업이 로드 되지 않은 상태에서 파일이 로드되는 지 확인한다.	Load File	R1.1	P
1.2	작업 파일 불러오기 시험	작업이 로드 된 상태에서 파일이 로드되는 지 확인한다.	Load File	R1.1	P
1.3	작업 파일 불러오기 시험	존재하지 않는 파일에 대해 로드한 경우 에러를 표시하는 지 확인한다.	Load File	R1.1	P
2.1	작업중이던 파일 저장 시험	작업이 진행 되지 않은 상태에서 파일이 저장되는 지 확인한다.	Save File	R1.2	P
2.2	작업중이던 파일 저장 시험	작업이 로드 된 상태에서 파일이 저장되는 지 확인한다.	Save File	R1.2	P
2.3	작업중이던 파일 저장 시험	유효하지 않은 경로에 대해 파일을 저장하는 경우 에러를 표시하는 지 확인한다.	Save File	R1.2	P
3.1	새로운 카테고리 생성 시험	이름을 입력한 상태로 카테고리를 생성하는 지 확인한다.	New Category	R2.1	P
3.2	새로운 카테고리 생성 시험	아무 이름도 입력하지 않은 상태에서 카테고리를 생성하는 지 확인한다.	New Category	R2.1	P
4.1	카테고리 삭제 시험	카테고리 삭제가 정상적으로 수행되는 지 확인한다.	Remove Category	R2.2	P
5.1	카테고리 이름 수정 시험	카테고리 이름을 수정	Modify Category	R2.3	P
6.1	새로운 Rep. Value 생성 시험	Rep. Value 생성이 되는 지 확인한다.	New Representative Value	R3.1	P
7.1	Rep. Value 삭제 시험	Rep. Value 삭제가 되는 지 확인한다.	Remove Representative	R3.2	P

			Value		
8.1	Rep. Value 이름 수정 시험	Rep. Value 의 이름이 수정되는 지 확인한다.	Modify Representative Value	R3.3	P
9.1	새로운 제약조건 생성 시험	새로운 Constraints 가 생성되어 None 값을 갖는 지 확인한다.	New Constraints	R 4.1	P
10.1	제약조건 삭제 시험	Constraints 가 정상적으로 삭제되는 지 확인한다.	Remove Constraints	R 4.2	P
11.1	제약조건 항목 수정 시험	Constraints 의 항목이 수정되는 지 확인한다.	Modify Constraints	R 4.3	P
12.1	새로운 Property 생성 시험	새로운 Property 가 정상적으로 생성되는 지 확인한다.	New Property	R 5.1	P
13.1	Property 삭제 시험	Property 가 정상적으로 삭제되는 지 확인한다.	Remove Property	R 5.2	P
14.1	Property 이름 수정 시험	Property 의 이름이 수정되는 지 확인한다.	Modify Property	R 5.3	P
15.1	Calculate 시험	입력된 정보를 바탕으로 Test Case 계산이 정상적으로 수행되는 지 확인한다.	Calculate	R 6.1	P
16.1	결과 화면 보여주기 시험	수행중인 작업이 있는 상태에서 Test Case 생성 확인	Show Result	R 6.2	P
16.2	결과 화면 보여주기 시험	수행중인 작업이 없는 상태에서 Test Case 생성 확인	Show Result	R 6.2	P
17.1	필터링할 대푯값 설정 시험	필터링할 Representative Value 선택했을 때 결과 화면이 정상적으로 변하는 지 확인한다.	Filter Representative Value	R 6.2.1	P
17.2	필터링할 대푯값 설정 시험	필터링할 Representative Value 선택을 해제했을 때 결과 화면이 정상적으로 변하는 지 확인한다.	Filter Representative Value	R 6.2.1	P
18.1	필터링할	필터링할 Property 선택했을	Filter Property	R 6.2.2	P

	Property 설정 시험	때 결과 화면이 정상적으로 변하는 지 확인한다.			
18.2	필터링할 Property 설정 시험	필터링할 Property 선택을 해제했을 때 결과 화면이 정상적으로 변하는 지 확인한다.	Filter Property	R 6.2.2	P
19.1	결과 화면 파일 저장 시험	Show Result 화면에서 Save Result 버튼을 클릭하고 파일이 정상적으로 저장되는 지 확인한다.	Save Result	R 6.3	F
20.1	도움말 제공 기능 시험	작업 화면에서 도움말 버튼을 클릭했을 때 도움말 화면이 정상적으로 표시되는 지 확인한다.	Provide Help	R 6.4	P
21.1	Test Case 수행 결과 입력 시험	Show Result 화면에서 Test Case 수행 결과가 입력되는 지 확인한다.	Check True/False	R 7.1	P

### Activity 2067. Testing Traceability Analysis

