Smart DJ CoffeeMaker

Stage 2200 2nd cycle

T1

200611499 이낙원 , 200611521 최정명
200611460 김정태 , 200611481 송준현

목 차

1. Revised Requirements – p.3

2. Refine Use Case Diagram – p.4

3. Define System Sequence Diagrams – p.5

4. Design Real Use Cases – p.8

5. Define Interaction Diagrams – p.20

6. Define Design Class Diagram – p.31

7. Conclusion – p.38

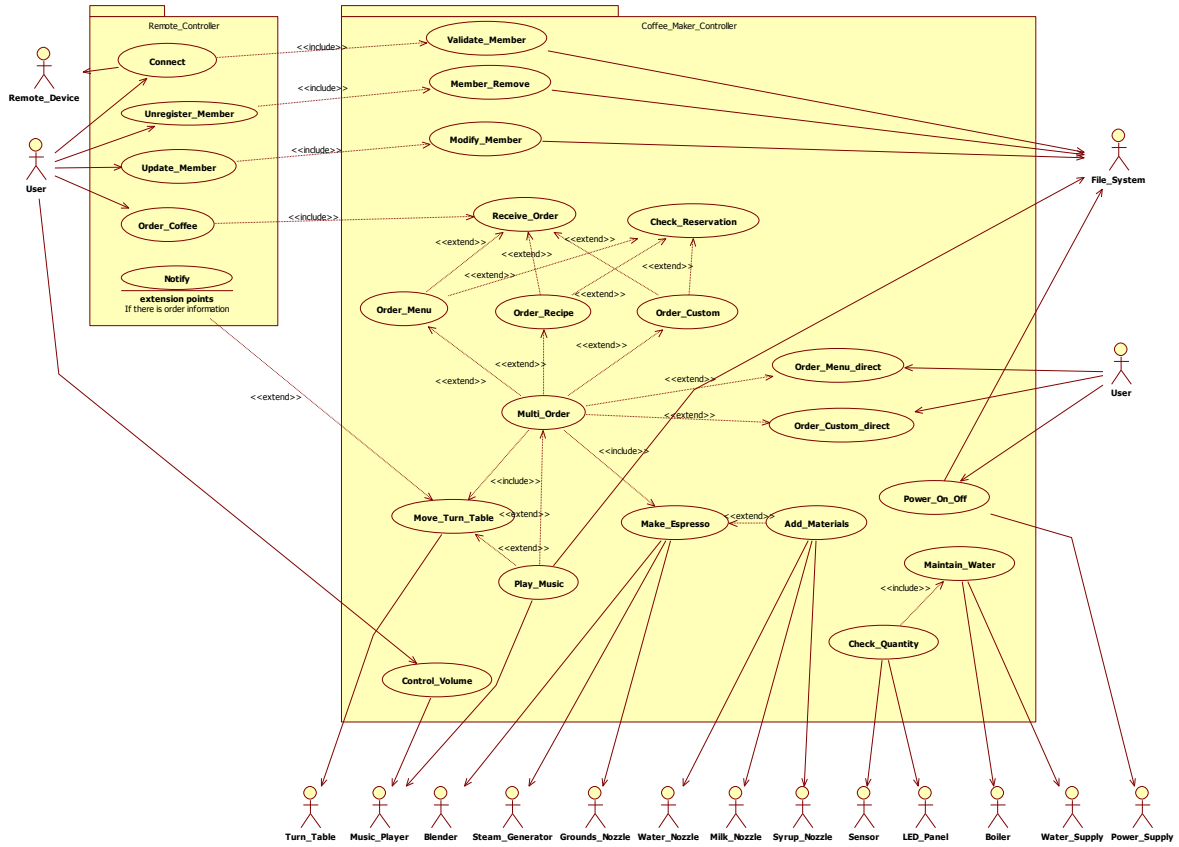
8. Appendix – p.39

Phase 2230. Analyze

<Revised Requirements>

Reference	Function	Use case Number and Name
R1	**power on / off**	**1. Power_On_Off**
R2.1	**connect**	**2. Connect**
R2.2	**validate member**	**3. Validate_Member**
R2.3	unregister member	4. Unregister_Member
R2.3	unregister member	5. Member_Remove
R3.1	order coffee + reservation	6. Order_Coffee
R3.2	receive order	7. Receive_Order
R4	**check reservation**	**8. Check_Reservation**
R5	multi order	9. Multi_Order
R6.1.1	order menu	10. Order_Menu
R6.1.2	order menu direct	11. Order_Menu_Direct
R6.2	order recipe	12. Order_Recipe
R6.3.1	order custom	13. Order_Custom
R6.3.2	order custom direct	14. Order_Custom_Direct
R7	**make coffee**	**15. Make_Espresso**
R7	**make coffee**	**16. Add_Materials**
R8	**music play during making**	**17. Play_Music**
R9	move turn table	18. Move_Turn_Table
R10	Notify	19. Notify
R11	control volume	20. Control_Volume
R12	**maintain water**	**21. Maintain_Water**
R13	check quantity of materials	22. Check_Quantity
R14	**update member info**	**23. Update_Member**
R14	**update member info**	**24. Modify_Member**

Activity 2232. Refine Use Case Diagrams

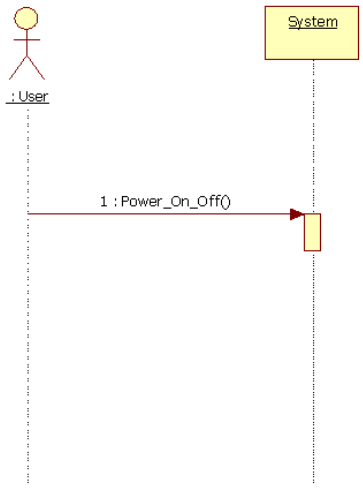


Activity 2235. Define System Sequence Diagrams

Requirement : R1 power on

USE CASE : 1. Power_On_Off

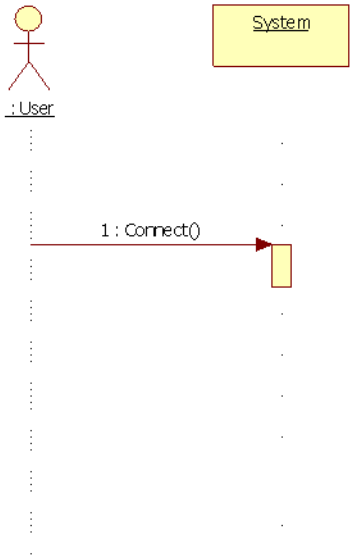
1. (A) 해당 use-case는 사용자가 전원 버튼을 누르는 동작에서 시작한다.
2. (S) 현재 상태에서 전원을 키고, 끄는 작업을 수행한다.



Requirement : R2.1 Connect

USE CASE : 2. Connect

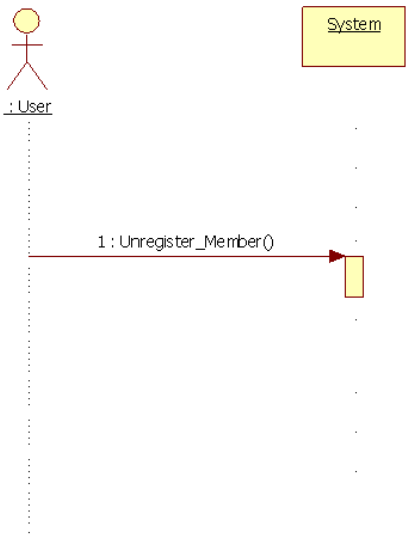
1. (A) 해당 use-case는 사용자가 리모트 디바이스를 작동시키는 동작에서 시작한다.
2. (S) "Validate_Member" use-case를 발동시킨다.
3. (S) 등록이 되어 있으면 접속을 허가하며, 등록이 안되어 있으면 등록 절차를 수행한다.



Requirement : R2.3 Unregister_Member

USE CASE : 4. Unregister

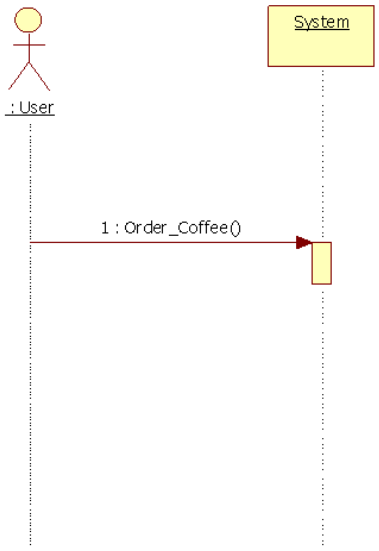
1. (A) 해당 use-case는 사용자가 정보 삭제 요청을 하는 것에서 시작한다.
2. (S) "Member_Remove" use-case를 발동한다.



Requirement : R3.1 Order Coffee + Reservation

USE CASE : 6. Order_Coffee

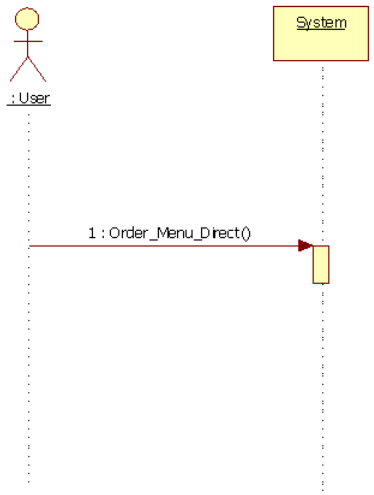
1. (A) 해당 use-case는 사용자가 커피를 주문하는 동작에서 시작한다.
2. (S) "Receive_Order" use-case를 발동한다.



Requirement : R6.1.2 Order Menu Direct

USE CASE : 11. Order_Menu_Direct

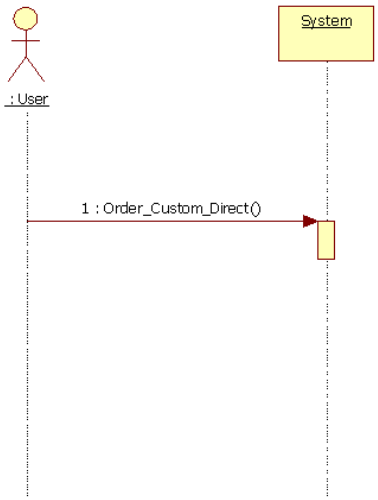
1. (A) 해당 use-case는 사용자가 커피 메이커에서 직접 커피 주문 버튼을 누르는 동작에서 시작한다.
2. 주문을 추가한다.
3. 커피가 제조 중인지 확인한다.
4. (S) "Multi_Order" use-case 를 발동한다.



Requirement : R6.3.2 Order Custom Direct

USE CASE : 14. Order_Custom_Direct

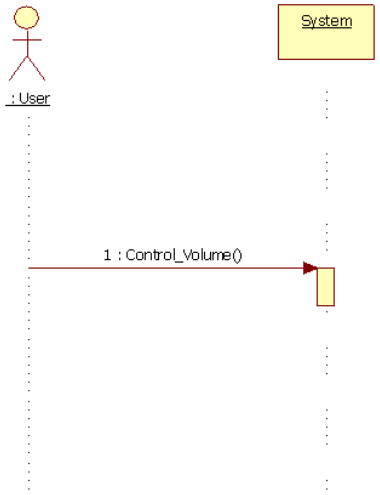
1. (A) 해당 use-case는 사용자가 커피 메이커에서 커피 재료량을 조절하여 커피를 주문하는 동작에서 시작한다.
2. 주문을 추가한다.
3. 커피가 제조 중인지 확인한다.
4. (S) "Multi_Order" use-case 를 발동한다.



Requirement : R11 Control_Volume

USE CASE : 20. Control_Volume

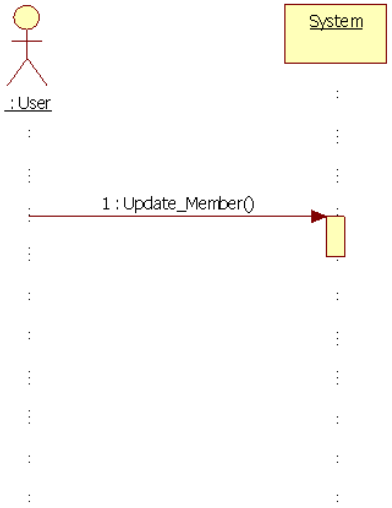
1. (A) 해당 use-case는 사용자가 볼륨 조절 버튼을 누르는 동작에서 시작한다.
2. (S) 전달받은 볼륨 정보대로 스피커의 볼륨을 조절한다.



Requirement : R14 Update Member Info

USE CASE : 23. Update_Member

1. (A) 해당 use-case는 사용자가 정보 수정 요청을 하는 동작에서 시작한다.
2. (S) "Modify_Member" use-case를 발동한다.



Phase 2240. Design

Activity 2241. Design Real Use Cases

Use Case	1. Power_On_Off
Actors	User
Purpose	- Load & Save state of *Coffee_Maker* and *Member*.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R1 Use-Case :
Pre-Requisites	N/A
Typical Courses of Events	1. (A) Push the ***power*** button. 2. (S) Check if ***Coffee_Maker.is_power*** is true or false. 3. (S) If ***Coffee_Maker.is_power*** is ***false*** 4. (S) Get member information, table number from ***File_System*** to ***Coffee_Maker.member***, ***Coffee_Maker().table_num***. 5. (S) Set ***Coffee_Maker.is_power*** to true.
Alternative Courses of Events	Line 4. If ***Coffee_Maker.is_power*** is true, save member information, table number to ***File_System***. Line 5. Set ***Coffee_Maker.is_Power*** to false.
Exceptional Courses of Events	N/A

Use Case	2. Connect
Actors	User
Purpose	Connect device to Coffee_Maker.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R2.1 Use-Case : "Validate_Member"
Pre-Requisites	N/A
Typical Courses of Events	1. (A) User executes remote control program. 2. (S) Get physical address of ***Remote_Device*** 3. (S) Send message which have physical address to coffee maker controller. 4. (S) Invoke "***Validate_Member***" use case. 5. (S) Coffee maker controller send if ***Remote_Device*** are registered and member id and coffee maker's physical address to ***Remote_Device***. 6. (S) If ***Remote_Device*** are registered set ***Remote_Control.is_connect*** to true. 7. (S) Set ***Remote_Control.member_ID*** to received member id. 8. (S) Save coffee maker's physical address to ***Remote_Control.coffee_maker_addr***.
Alternative Courses of Events	Line 6_1. (S) If ***Remote_Device*** aren't registered, request serial to User. 6_2. (S) Send serial number to coffee maker and receive result of serial number is correct. 6_3. (S) If serial number is correct, set ***Remote_Control.is_connect*** to true. 6_4. (S) Go to line 7.
Exceptional Courses of Events	Line 6_3. (S) If serial number isn't correct, keep ***Remote_Control.is_connect*** to false.

Use Case	3. Validate_Member
Actors	None
Purpose	Register user device on to Coffee_Maker.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R2.2 Use-Case : "Connect"
Pre-Requisites	N/A
Typical Courses of Events	1. (S) Remote control invoke ***Validate_Member()*** with physical address of remote device. 2. (S) Search corresponding address in ***Coffee_Maker.member.address***. 3. (S) Get Coffee_Maker.member.id of corresponding member. 4. (S) Get physical address of ***Coffee_Machine***. 5. (S) Send result of validating to ***Remote_Device***.
Alternative Courses of Events	Line 5_1. (S) If doesn't have corresponding member, Send member id that id is not allocating and message that ***Remote_Device*** aren't registered. 5_2. (S) ***Remote_Device*** send serial number to ***Coffee_Machine***. 5_3. (S) Compare received serial number with ***Coffee_Maker.serial_num***. 5_4. (S) Send result of comparison.
Exceptional Courses of Events	

Use Case	4. Unregister_Member
Actors	User
Purpose	Remove member information.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R2.3 Use-Case : "Member_Remove"
Pre-Requisites	User device's address is mapped on *Coffee_Maker.member*.
Typical Courses of Events	1. (A) User push *Unregister* button. 2. (S) Check if ***Remote_Control.is_connect*** is *true*. 3. (S) Get ***Remote_Control.coffee_maker_addr***. 4. (S) Request to send message which call member remove with member id to ***Coffee_Machine***.
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2. If ***Remote_Control.is_connect*** is false, do noting.

Use Case	5. Member_Remove
Actors	None
Purpose	Remove member information.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R2.3 Use-Case : "Unregister_Member"
Pre-Requisites	N/A
Typical Courses of Events	1. (S) Find ***Coffee_Maker.member[i]*** corresponding received id. 2. (S) Remove ***Coffee_Maker.member[i].address***. 3. (S) Initialize ***Coffee_Maker.Member[i].own_recipe*** 4. (S) Remove Music File on ***Coffee_Maker.Member[i].music_path*** from ***File_System***. 5. (S) Initialize ***Coffee_Maker.Member[i].music_path***.
Alternative Courses of Events	N/A
Exceptional Courses of Events	N/A

Use Case	6. Order_Coffee
Actors	User
Purpose	Order coffee.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R3.1 Use-Case : "Receive_Order"
Pre-Requisites	N/A
Typical Courses of Events	1. (A) User check the order type in ***Order Type***. 2. (A) User push ***Order*** button or ***Reservation*** button. 3. (S) Check if ***Remote_Control.is_connect*** is *true*. 4. (S) Get ***Remote_Control.coffee_maker_addr*** and member id. 5. (S) Invoke "Receive_Order" with member id order information and time to set by 2400.
Alternative Courses of Events	Line 1. (A) If order type is ***Menu***, user should select coffee menu in ***Menu Panel***. Line 1. (A) If order type is ***Custom***, user set quantity of coffee in ***Quantity Panel***. Line 2. (A) If user push ***Reservation*** button, show to user ***Reservation*** page. Line 5. (S) If user push ***Reservation*** button, send member id order information and time is set by user.
Exceptional Courses of Events	Line 3. If the value is false, terminate this use case.

Use Case	7. Receive_Order
Actors	None
Purpose	Classify order's kind.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R3.2 Use-Case : "Order_Custom", "Order_Recipe", "Order_Menu", "Order_Coffee"
Pre-Requisites	N/A
Typical Courses of Events	1. (S) Create "Recipe", "Order" object. 2. (S) Set ***Order.id*** to received id. 3. (S) Set ***Order.time***, ***Order.order_type***, ***Order.menu_type***, ***Order.recipe*** with received order information. 4. (S) Check ***Order.time***. 5. (S) Check ***Order.order_type***.
Alternative Courses of Events	Line 5. If ***Order.time*** is reservation, put *Order* in ***Coffee_Maker.reservation_array***. Line 6. If ***Order.order_type*** is Custom, invoke "Order_Custom" use case with ***Order***. If ***Order.order_type*** is Menu, invoke "Order_Menu" use case with ***Order***. If ***Order.order_type*** is Recipe, invoke "Order.Menu" use case with ***Order***.
Exceptional Courses of Events	N/A

Use Case	8. Check_Reservation
Actors	None
Purpose	Reserve order.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R4 Use-Case : "Order_Menu", "Order_Recipe", "Order_Custom"
Pre-Requisites	N/A
Typical Courses of Events	1. (S) Check ***Coffee.Maker.reservation_array[i].Order.time*** with current time. 2. (S) If the ***time*** corresponds with current time, check ***Coffee.Maker.reservation_array[i].Order.menu_type***. 3. (S) Go to Line 1.
Alternative Courses of Events	Line 2. If ***Order.order_type*** is Custom, invoke "Order_Custom" use case with ***Order***. If ***Order.order_type*** is Menu, invoke "Order_Menu" use case with ***Order***. If ***Order.order_type*** is Recipe, invoke "Order.Menu" use case with ***Order***.
Exceptional Courses of Events	N/A

Use Case	9. Multi_Order
Actors	None
Purpose	Make coffee according to order information.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R5 Use-Case : "Make_Espresso", "Play_Music", "Move_Turn_Table"
Pre-Requisites	N/A
Typical Courses of Events	1. (S) Check ***Coffee_Maker.order_count***. 2. (S) Set ***Coffee_Maker.is_working*** to *true*. 3. (S) Dequeue *Order* object in the ***Coffee_Maker.order_queue***. 4. (S) Decrease ***Coffee_Maker.order_count*** by 1. 5. (S) Check each materials are available with ***Order.recipe***. 6. (S) Get Order.id. 7. (S) Check if Order.id isn't 0. 8. (S) Invoke "Play_Music" with id and play that value is true . 9. (S) Invoke "Make_Espresso" use case with recipe. 10. (S) Wait for coffee making. 11. (S) Invoke "Move_Turn_Table" use-case with member id. 12. (S) Set ***Coffee_Maker.is_working*** to false. 13. (S) Check ***Coffee_Maker.order_count***. 14. (S) Go to Line 2.
Alternative Courses of Events	Line 7. If Order.id is 0, Go to line 9.
Exceptional Courses of Events	Line 5. If There is short of Materials, go to Line 1. Line 13. If ***Coffee_Maker.order_count*** is 0, terminate this use case.

Use Case	10. Order_Menu
Actors	None
Purpose	Order Coffee Menu.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R6.1.1 Use-Case : "Multi_Order"
Pre-Requisites	N/A
Typical Courses of Events	1. (S) Compare ***Order.menu_type*** with ***Coffee_Maker.Default_Recipe.menu_type***. 2. (S) If two variables are same, set ***Order.recipe*** to ***Coffee_Maker.Default_Recipe***. 3. (S) Input *Order* to ***Coffee_Maker.order_queue***. 4. (S) Increase ***Coffee_Maker.order_count*** by 1. 5. (S) Check if ***Coffee_Maker.is_working***. 6. (S) Invoke "Multi_Order" use-case.
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 5. If ***Coffee_Maker.is_Working*** is true, do not invoke "Multi_Order" use-case.

Use Case	11. Order_Menu_Direct
Actors	User
Purpose	User orders coffee menu directly.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R6.1.2 Use-Case : "Multi_Order"
Pre-Requisites	N/A
Typical Courses of Events	1. (A) User select Coffee menu at ***Menu Panel.*** 2. (S) Create ***Order*** object. 3. (S) Set ***Order.time*** with 2400. 4. (S) Set ***Order.id*** with 0. 5. (S) Set ***Order.menu_type*** and ***Order.order_type.*** 6. (S) Compare ***Order.menu_type*** with ***Coffee_Maker.Default_Recipe.menu_type.*** 7. (S) If two variables are same, set ***Order.recipe*** to ***Coffee_Maker.Default_Recipe*** 8. (S) Input ***Order*** to ***Coffee_Maker.order_queue.*** 9. (S) Increase ***Coffee_Maker.order_count*** by 1. 10. (S) Check if ***Coffee_Maker.is_working.*** 11. (S) Invoke "Multi_Order" use-case.
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 10. If ***Coffee_Maker.is_Working*** true, do not invoke "Multi_Order" use-case.

Use Case	12. Order_Recipe
Actors	None
Purpose	User order coffee by his own stored recipe.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R6.2 Use-Case : "Multi_Order"
Pre-Requisites	N/A
Typical Courses of Events	1. (S) Get ***Order.id*** from received ***Order.*** 2. (S) Search corresponding ***Order.id*** in ***member.*** 3. (S) Set ***Order.recipe*** to ***Member.own_recipe.*** 4. (S) Input ***Order*** to ***Coffee_Maker.order_queue.*** 5. (S) Increase ***Coffee_Maker.order_count*** by 1. 6. (S) Check if ***Coffee_Maker.is_working.*** 7. (S) Invoke "Multi_Order" use-case.
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 6. If ***Coffee_Maker.is_Working*** true, do not invoke "Multi_Order" use-case.

Use Case	13. Order_Custom
Actors	None
Purpose	User order coffee by customized recipe.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R6.3.1 Use-Case : "Multi_Order"
Pre-Requisites	N/A
Typical Courses of Events	1. (S) Input *Order* to ***Coffee_Maker.order_queue***. 2. (S) Increase ***Coffee_Maker.order_count*** by 1. 3. (S) Check if ***Coffee_Maker.is_working***. 4. (S) Invoke "Multi_Order" use-case.
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 3. If ***Coffee_Maker.is_Working*** true, do not invoke "Multi_Order" use-case.

Use Case	14. Order_Custom_Direct
Actors	User
Purpose	User order coffee directly at the Coffee_Machine by customized recipe.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R6.3.2 Use-Case : "Multi_Order"
Pre-Requisites	N/A
Typical Courses of Events	1. (A) User sets value of ***Coffee, Water, Milk, Syrup*** of ***Custom Panel***. 2. (A) then, user push ***Order*** button. 3. (S) Create ***Order*** object. 4. (S) Set ***Order.time*** to 2400. 5. (S) Set ***Order.id*** to 0. 6. (S) Set ***Order.menu_type*** and ***Order.order_type***. 7. (S) Create ***Recipe*** object. 8. (S) Set ***Recipe.coffee, Recipe.water, Recipe.milk, Recipe.syrup***. 9. (S) Input *Order* to ***Coffee_Maker.order_queue***. 10. (S) Increase ***Coffee_Maker.order_count*** by 1. 11. (S) Check if ***Coffee_Maker.is_working***. 12. (S) Invoke "Multi_Order" use-case.
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 11. If ***Coffee_Maker.is_Working*** true, do not invoke "Multi_Order" use-case.

Use Case	15. Make_Espresso
Actors	None
Purpose	Coffee maker make espresso.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R7 Use-Case : "Add_Materials"
Pre-Requisites	N/A
Typical Courses of Events	1. (S) Get ***Recipe.coffee*** from Received recipe. 2. (S) Quantity of coffee exchange unit to gram. 3. (S) Call ***Blend_Coffee()*** with quantity of coffee to ***Blender*** and wait for blending is done. 4. (S) Calculate time for extraction. 5. (S) Call ***Steam_Shot()*** with calculated time to ***Steam_Generator*** and wait for extraction is done. 6. (S) Call ***Dump_Grounds()*** to ***Ground_Nozzle***. 7. (S) Check if necessary to add other materials. 8. (S) Send end message to "Multi_Order" use case.
Alternative Courses of Events	Line 7. (S) If other materials are needed, invoke "Add_Materials" use case with recipe.
Exceptional Courses of Events	N/A

Use Case	16. Add_Materials
Actors	None
Purpose	Add materials except coffee to espresso.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R7 Use-Case :
Pre-Requisites	N/A
Typical Courses of Events	1. (S) Get ***Recipe.milk*** and calculate time for open nozzle with ***Recipe.milk***. 2. (S) Call ***Shot_Milk***() with calculated time to ***Milk_Nollze***. 3. (S) Get ***Recipe.water*** and calculate time for open nozzle with ***Recipe.water***. 4. (S) Call ***Shot_Water***() with calculated time to ***Water_Nollze***. 5. (S) Get ***Recipe.syrup*** and calculate time for open nozzle with ***Recipe.syrup***. 6. (S) Call ***Shot_Syrup***() with calculated time to ***Syrup_Nollze***. 7. (S) Send end message to "Make_Esspresso" use case.
Alternative Courses of Events	N/A
Exceptional Courses of Events	N/A

Use Case	17. Play_Music
Actors	None
Purpose	Play music while Coffee_Maker is working.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R8 Use-Case : "Multi_Order"
Pre-Requisites	N/A
Typical Courses of Events	1. (S) Check received variable ***play*** is true. 2. (S) Get ***Member.music_path***. 3. (S) Call ***Play_Music()*** with ***Member.music_path*** to ***Music_Player***.
Alternative Courses of Events	Line 1. (S) If received variable ***play*** is false, call ***Stop_Music()*** to ***Music_Player***.
Exceptional Courses of Events	N/A

Use Case	18. Move_Turn_Table
Actors	None
Purpose	Turn 1 cycle the turn table and stop playing music.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R9 Use-Case : "Notify", "Multi_Order", "Play_Music"
Pre-Requisites	N/A
Typical Courses of Events	1. (S) Check received *Id* isn't 0. 2. (S) Invoke "Play_Music" use case with play set value by false. 3. (S) Get ***Coffee_Maker.member[i].address*** corresponding to received id. 4. (S) Invoke "Notify" use case with ***Coffe_Maker.table_num***. 5. (S) Call ***Move_Turn_Table()*** to ***Turn_Table***. 6. (S) Increase ***Coffee_Maker.table_num*** by 1.
Alternative Courses of Events	Line 1. If received Id is 0, go to Line 5.
Exceptional Courses of Events	Line 6. If ***Coffee_Maker.table_num*** bigger than 6, turn *the value* to 1.

Use Case	19. Notify
Actors	None
Purpose	Notify of completion of coffee and turn table number.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R10 Use-Case : "Move_Turn_Table"
Pre-Requisites	N/A
Typical Courses of Events	1. (S) Show ***Notify Window*** with turn table number to user.
Alternative Courses of Events	N/A
Exceptional Courses of Events	N/A

Use Case	20. Control_Volume
Actors	User
Purpose	Control Coffee_Maker music volume.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R11 Use-Case :
Pre-Requisites	N/A
Typical Courses of Events	1. (A) User push volume button on ***Volume Panel***. 2. (S) If pushed button is ***up button***, call ***Volume_Up()*** to ***Music_Player***.
Alternative Courses of Events	Line 2. If pushed button is ***down button***, call ***Volume_Down()*** to ***Music_Player***.
Exceptional Courses of Events	N/A

Use Case	21. Maintain_Water
Actors	None
Purpose	Keep temperature and quantity of water.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R12 Use-Case : "Check_Quantity"
Pre-Requisites	*Coffee_Machine* must be working.
Typical Courses of Events	1. (S) Get ***Material.water_Q***. 2. (S) If ***water_Q*** is less then 5, call ***Start_Supply()*** to ***Water_Supply***. 3. (S) Get ***Material.water_T***. 4. (S) If ***water_T*** is less then 85, call ***Start_Boil()*** to ***Boiler***.
Alternative Courses of Events	Line 2. If ***water_Q*** is bigger than 90, call ***Stop_Supply()*** to ***Water_Supply***. Line 4. If ***water_T*** is bigger than 95, call ***Stop_Boil()*** to ***Boiler***.
Exceptional Courses of Events	N/A

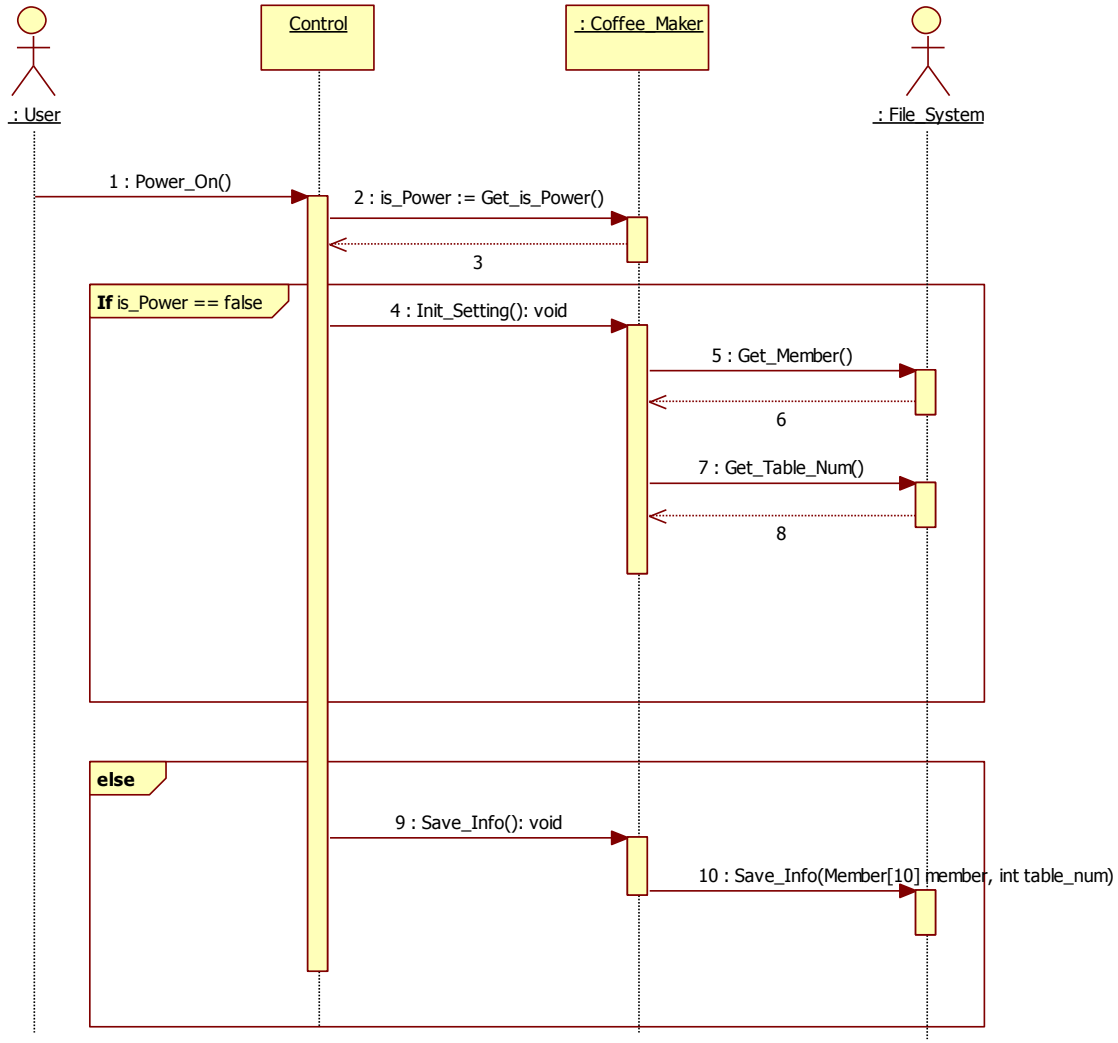
Use Case	22. Check_Quantity
Actors	None
Purpose	- *Coffee_Machine* check the materials quantity and represent short of materials. - *Coffee_Machine* check the coffee grounds and represent quantity of that.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R13 Use-Case : "Maintain_Water"
Pre-Requisites	N/A
Typical Courses of Events	1. (S) Call ***Check_Water()*** to Sensor and save return value at ***water_Q***. 2. (S) Call ***Check_Temparature()*** to Sensor and save return value at ***water_T***. 3. (S) If ***water_Q*** less than 5, call ***Water_On()*** to ***Water LED_Panel***. 4. (S) Call ***Check_Coffee()*** to Sensor and save return value at ***coffee_Q***. 5. (S) If ***coffee_Q*** less than 5, call ***Coffee_On()*** to ***Coffee LED_Panel***. 6. (S) Call ***Check_Milk()*** to Sensor and save return value at ***milk_Q***. 7. (S) If ***milk_Q*** less than 5, call ***Milk_On()*** to ***Milk LED_Panel***. 8. (S) Call ***Check_syrup()*** to Sensor and save return value at ***syrup_Q***. 9. (S) If ***syrup_Q*** less than 5, call ***Syrup_On()*** to ***Syrup LED_Panel***. 10. (S) Call ***Check_Grounds()*** to Sensor and save return value at ***Grounds_Q***. 11. (S) If ***grounds_Q*** bigger than 90, call ***Grounds_On()*** to ***Grounds LED_Panel***. 12. (S) Invoke "Maintain_Water" use case.
Alternative Courses of Events	Line 3. If ***water_Q*** bigger than 5, call ***Water_Off()*** to ***Water LED_Panel***. Line 5. If ***coffee_Q*** bigger than 5, call ***Coffee_Off()*** to ***Coffee LED_Panel***. Line 7. If ***milk_Q*** bigger than 5, call ***Milk_Off()*** to ***Milk LED_Panel***. Line 9. If ***syrup_Q*** bigger than 5, call ***Syrup_Off()*** to ***Syrup LED_Panel***. Line 11. If ***grounds_Q*** less than 90, call ***Grounds_Off()*** to ***Grounds LED_Panel***.
Exceptional Courses of Events	N/A

Use Case	23. Update_Member
Actors	User
Purpose	Update user recipe.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R14 Use-Case : "Modify_Member"
Pre-Requisites	N/A
Typical Courses of Events	1. (A) User sets quantity of materials on *Quanity* label. 2. (A) User push *Update Recipe* button. 3. (S) Check if *Remote_Control.is_connect* is *true*. 4. (S) Get Remote_Control.member_id. 5. (S) Get Remote_Control.coffee_maker_addr. 6. (S) Call Get_Music() with null path to Remote_Device. 7. (S) Invoke "Modify_Member" use case with quantity of materials.
Alternative Courses of Events	Line 2_1. (A) If user push Update Music button. 2_2. (S) Show Music Update page to user. 2_3. (A) Select music path. 2_4. (S) Go to line 4. 6_1. (S) Call Get_Music() with selected music path to Remote_Device. 6_2. (S) Go to line 7.
	Line 3. if *the value* is *false*, terminate this use case.

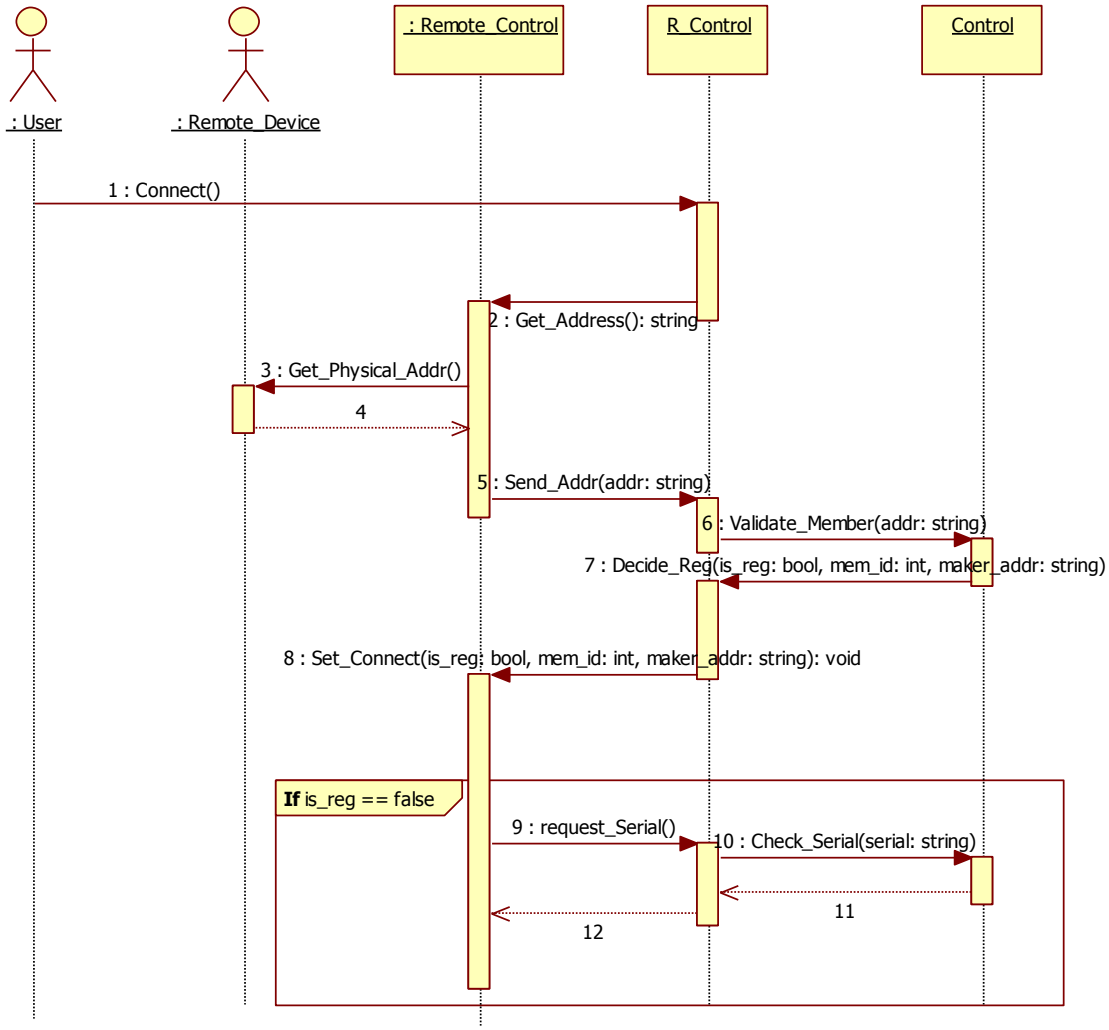
Use Case	24. Modify_Member
Actors	None
Purpose	Update user recipe.
Overview	(As in the Business use case)
Type	Primary & Essential
Cross Reference	Functional Requirement : R14 Use-Case : "Update_Member"
Pre-Requisites	N/A
Typical Courses of Events	1. (S) Check received Music File is null. 2. (S) Create Recipe object and setting quantity of recipe by received quantity of materials. 3. (S) Set Coffee_Maker.member[i].own_recipe with created Recipe object.
Alternative Courses of Events	Line 2_1. (S) If received Music File isn't null, call Get_Music_Path() with Music File to File_System. Line 2_2. (S) Set Coffee_Maker.member[i].music_path with return path.
Exceptional Courses of Events	Line 2_1. (S) If return value of Get_Music_Path() is null, terminate this use case.

Activity 2244. Define Interaction Diagrams

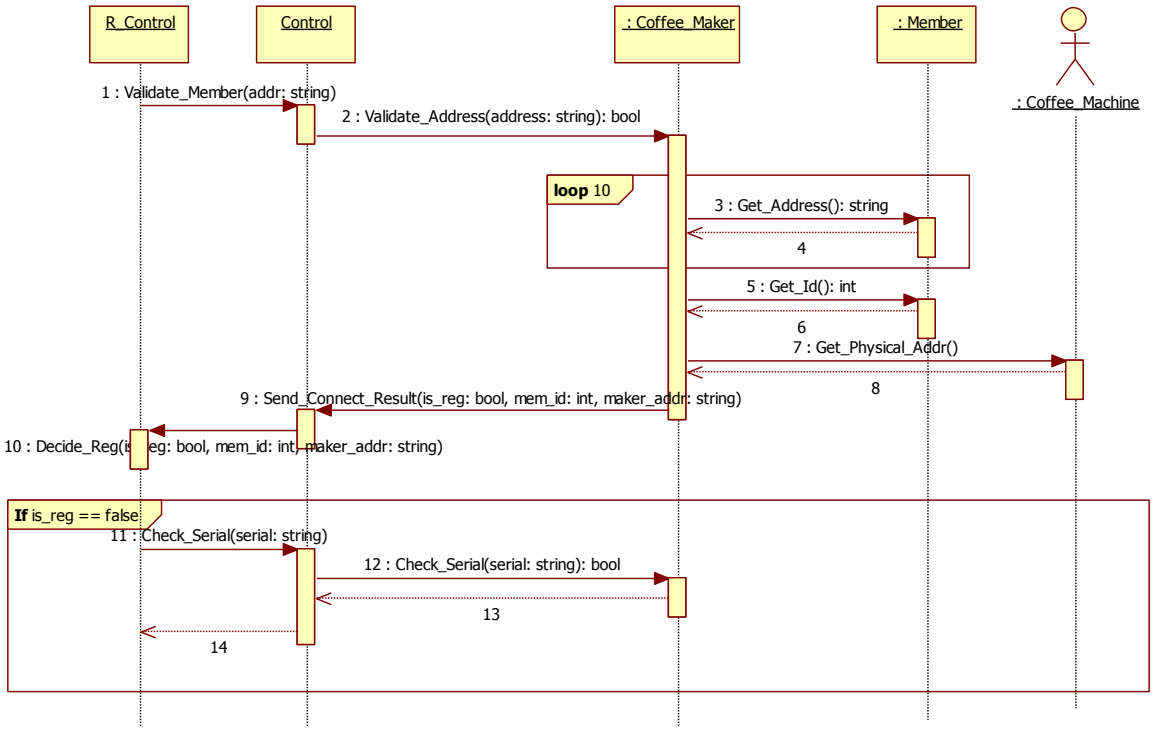
1. Power_On_Off



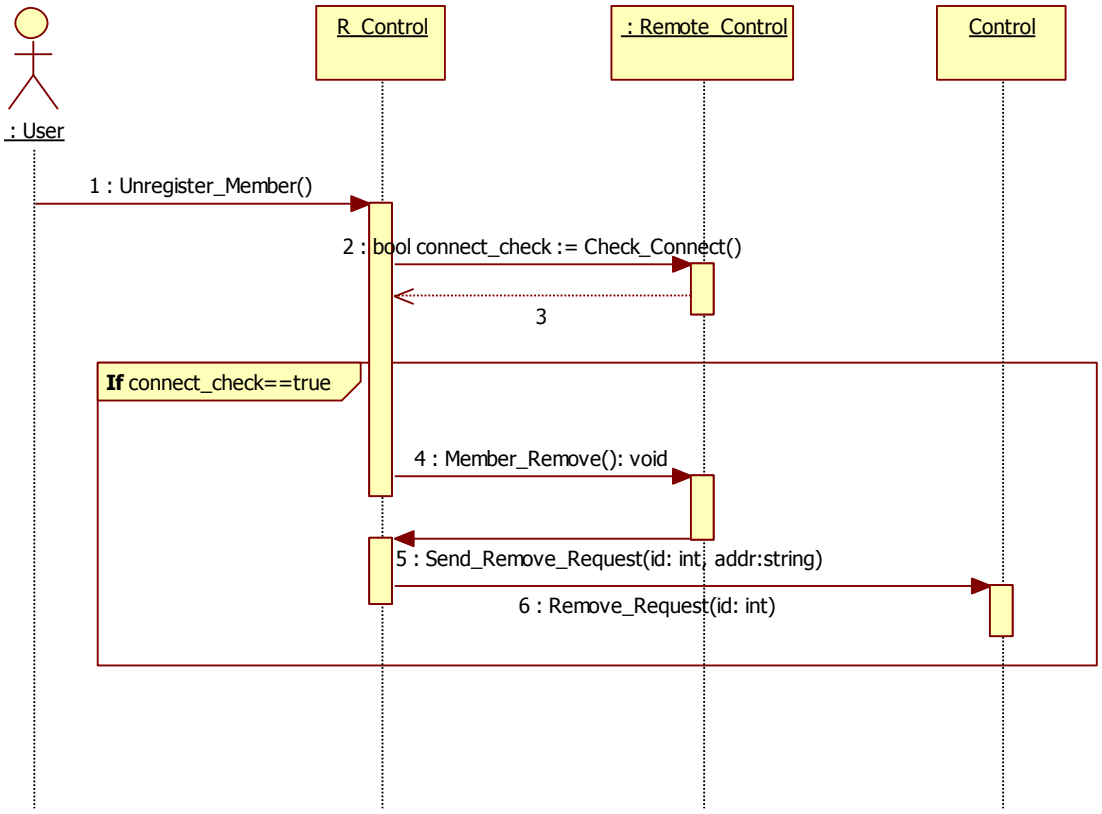
2. Connect


: User
: Remote_Device
: Remote_Control
R_Control
Control
1 : Connect()
2 : Get_Address(): string
3 : Get_Physical_Addr()
4
5 : Send_Addr(addr: string)
6 : Validate_Member(addr: string)
7 : Decide_Reg(is_reg: bool, mem_id: int, maker_addr: string)
8 : Set_Connect(is_reg: bool, mem_id: int, maker_addr: string): void
If is_reg == false
9 : request_Serial()
10 : Check_Serial(serial: string)
11
12

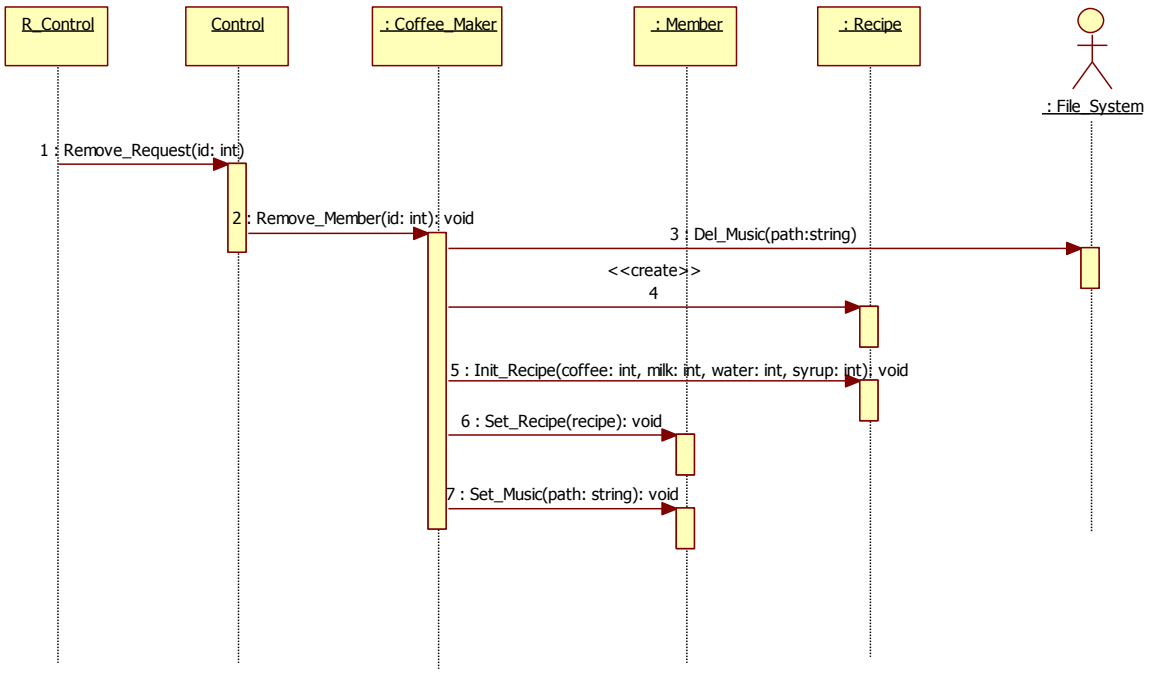
3. Validate_Member



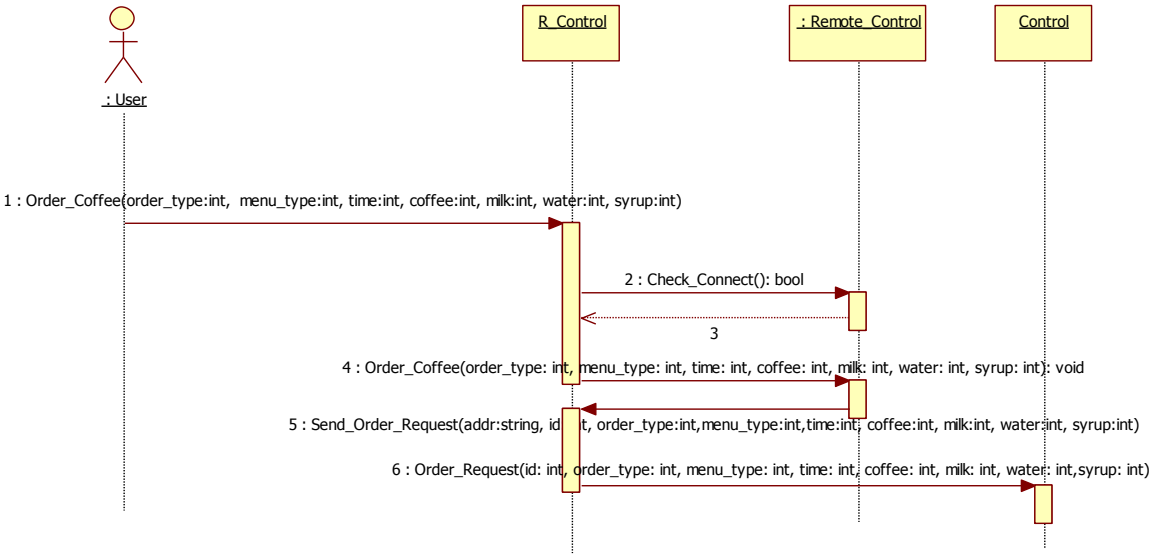
4. Unregister_Member



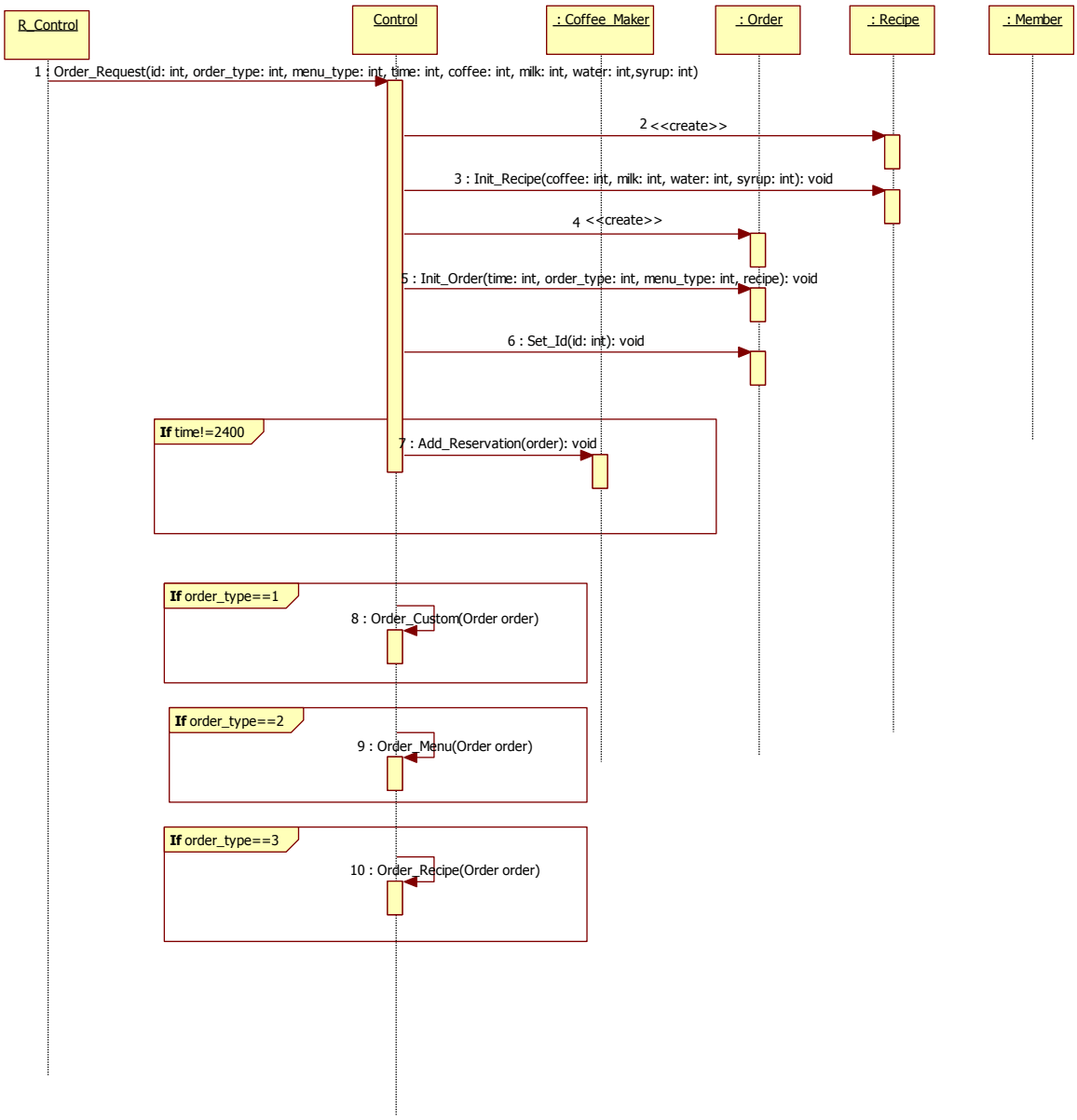
5. Member_Remove



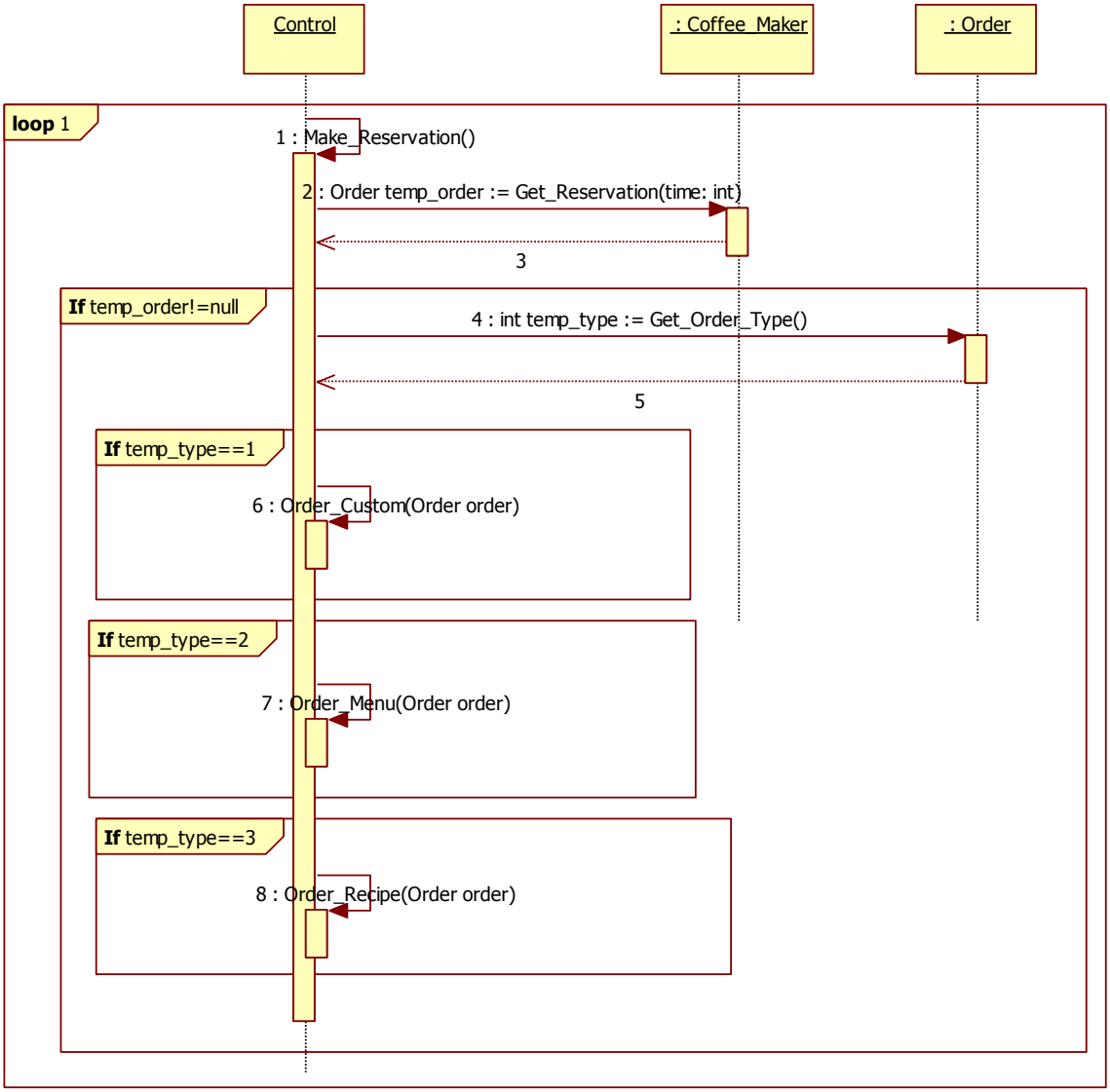
6. Order_Coffee



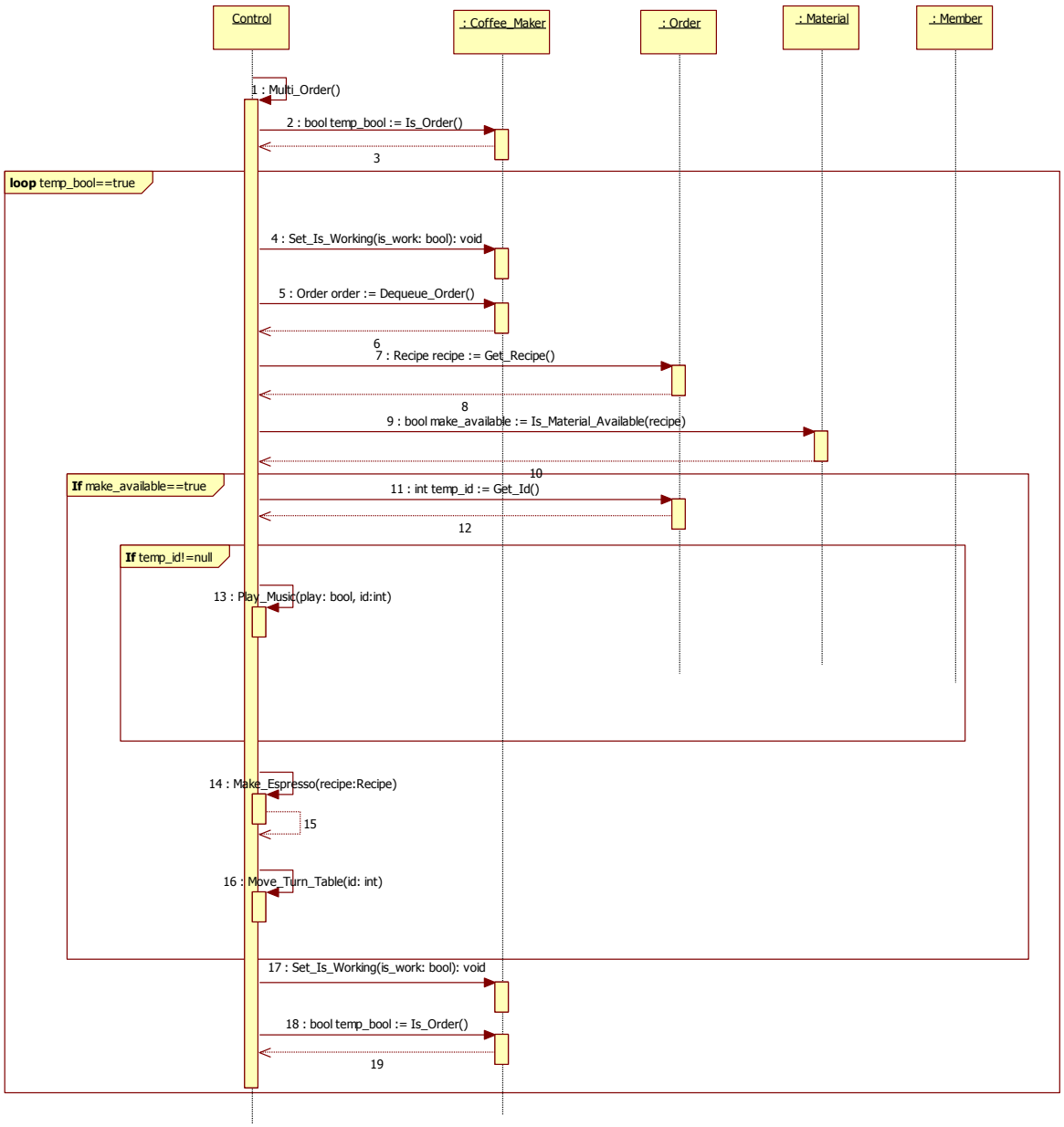
7. Receive_Order



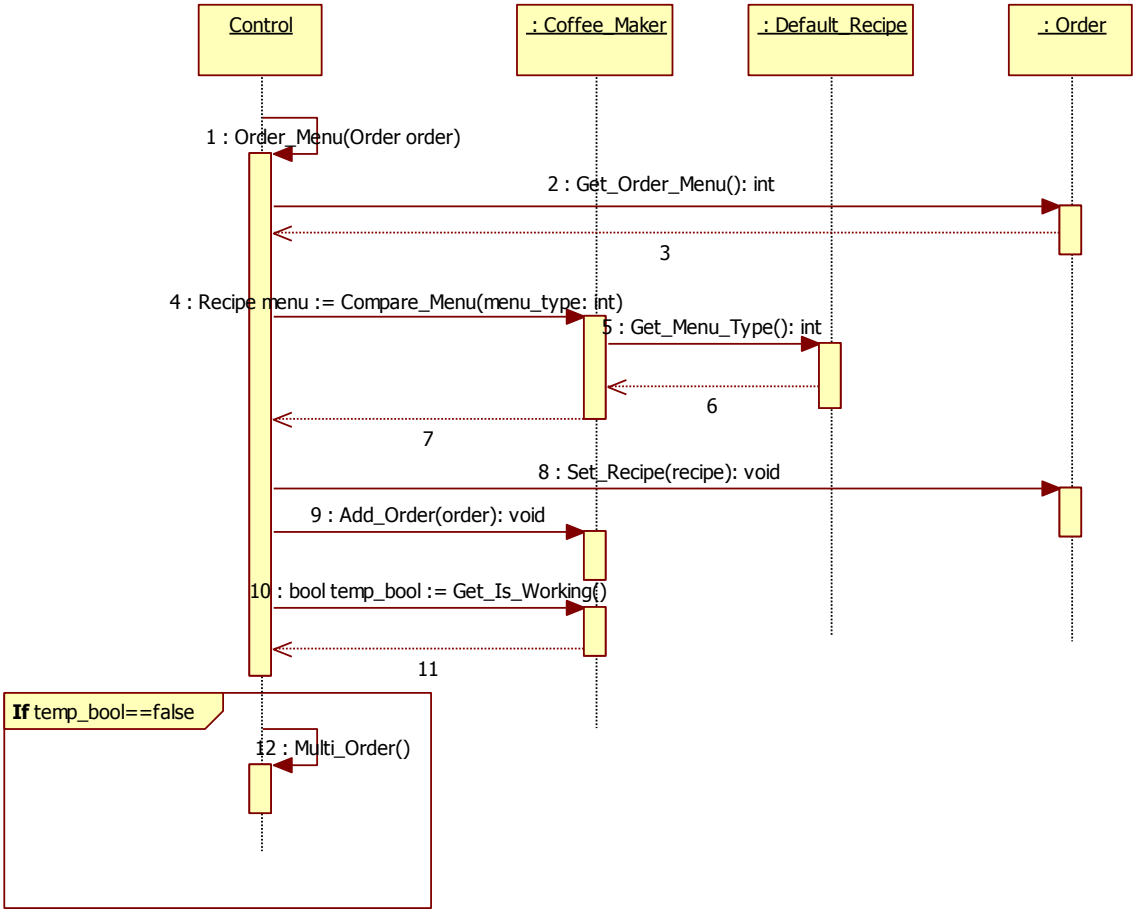
8. Check_Reservation



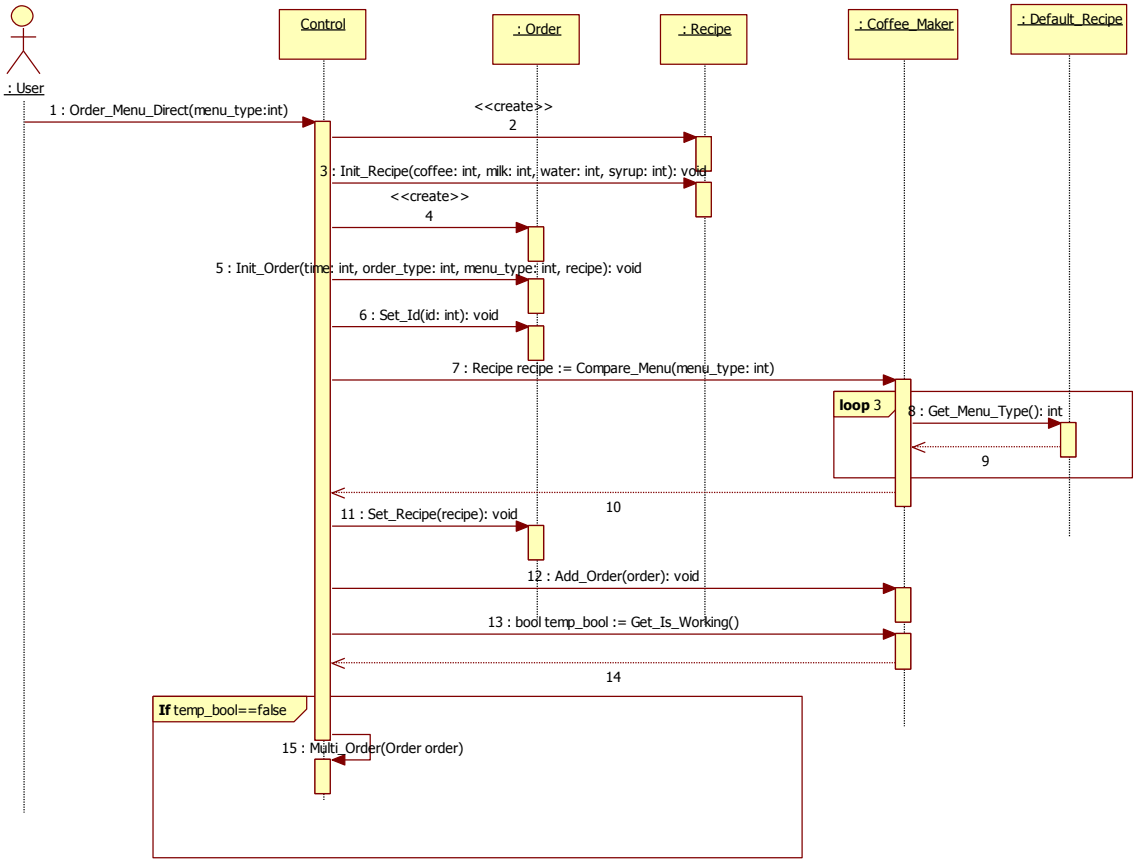
9. Multi_Order



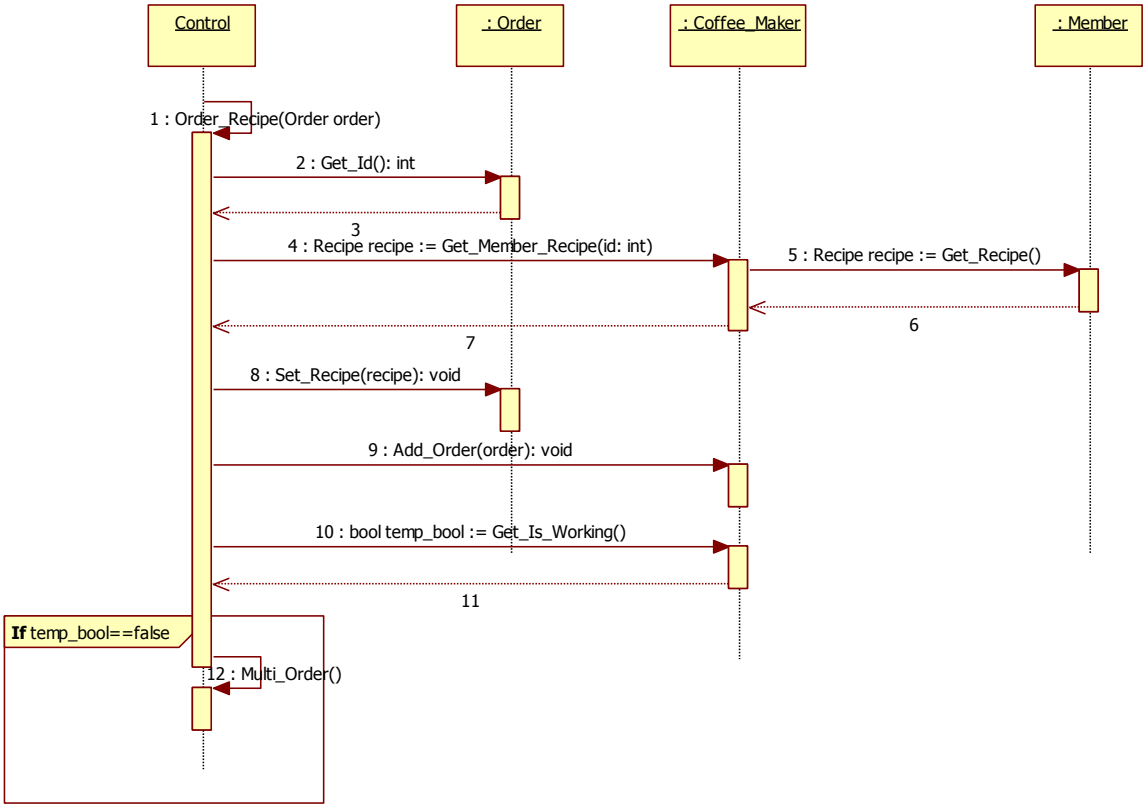
10. Order_Menu



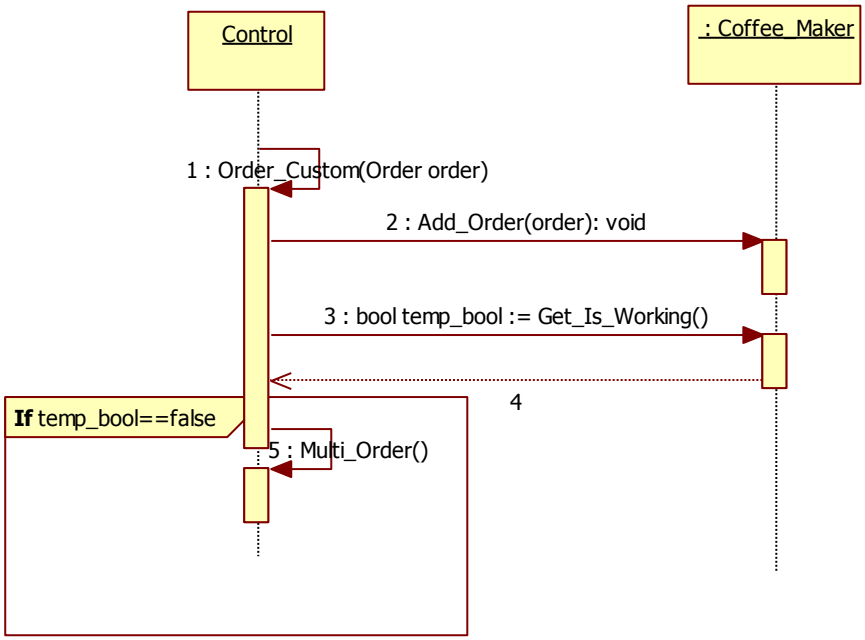
11. Order_Menu_Direct



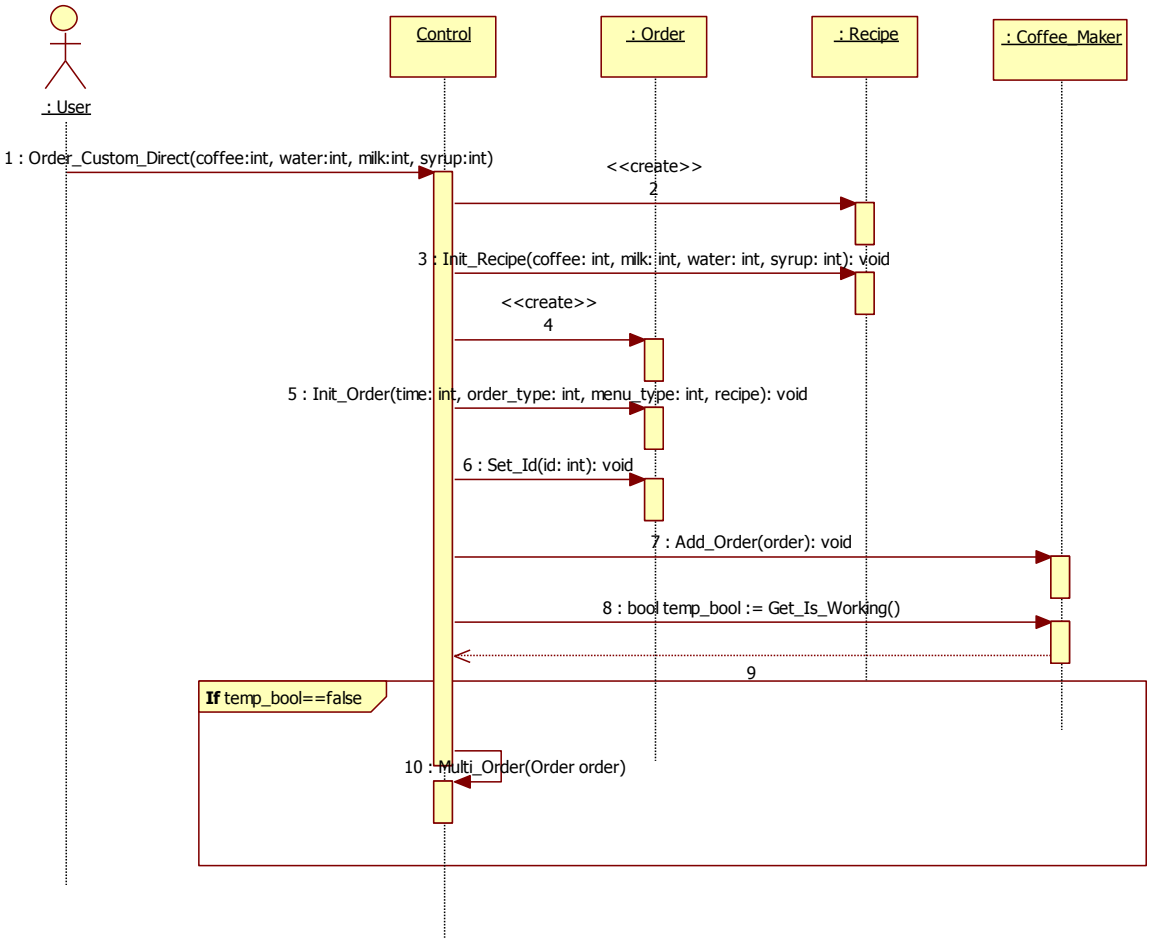
12. Order_Recipe



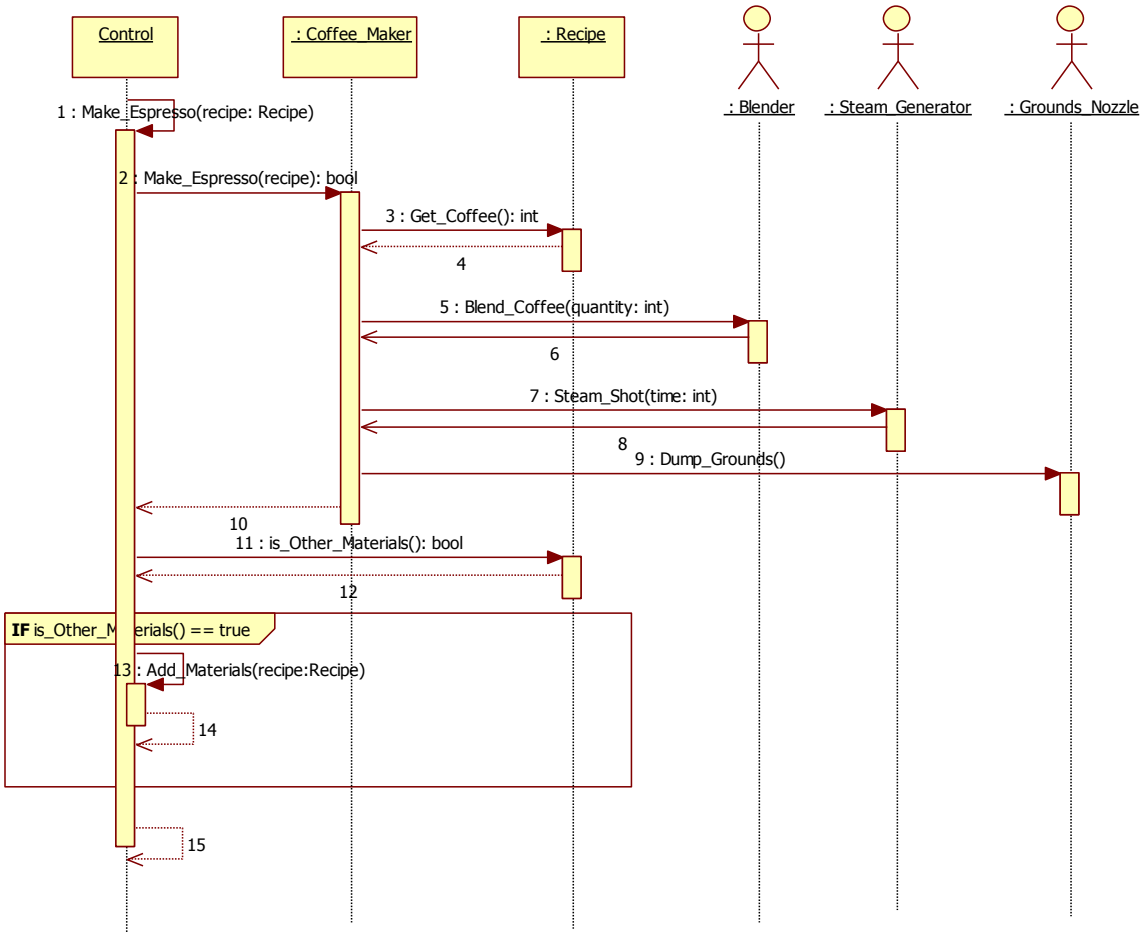
13. Order_Custom



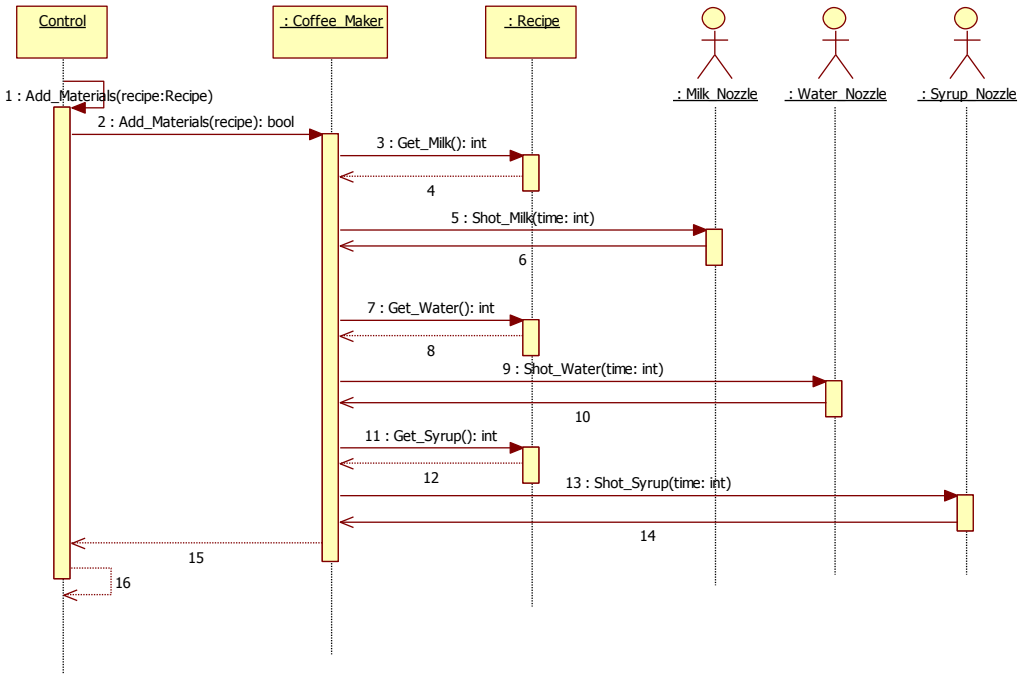
14. Order_Custom_Direct



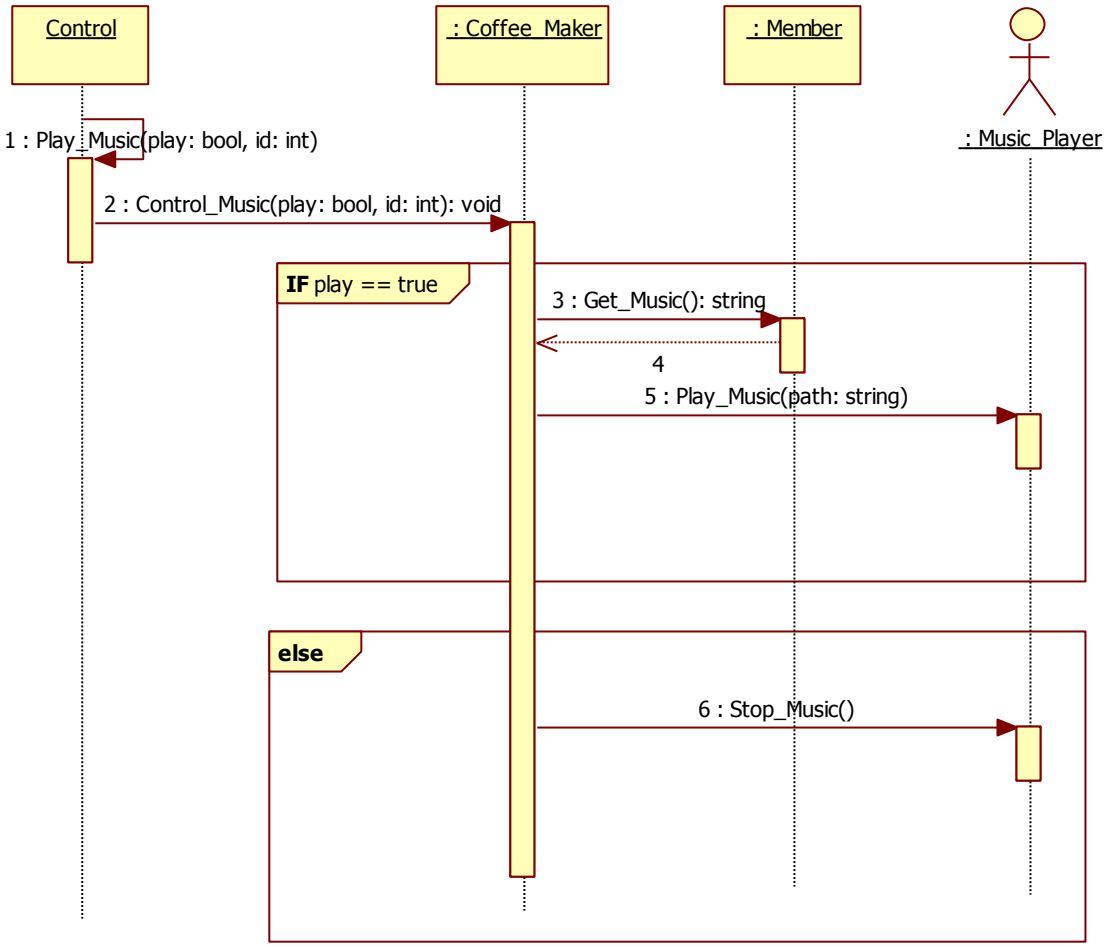
15. Make_Espresso



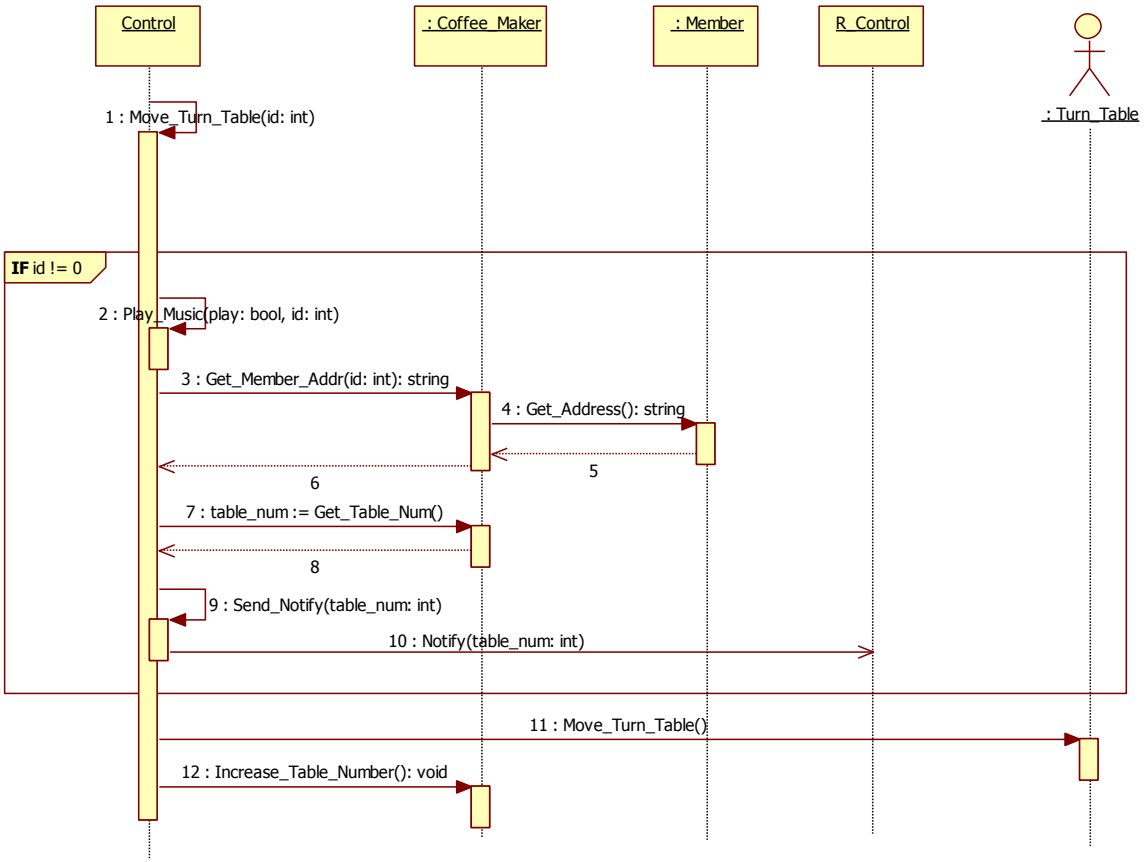
16. Add_Materials



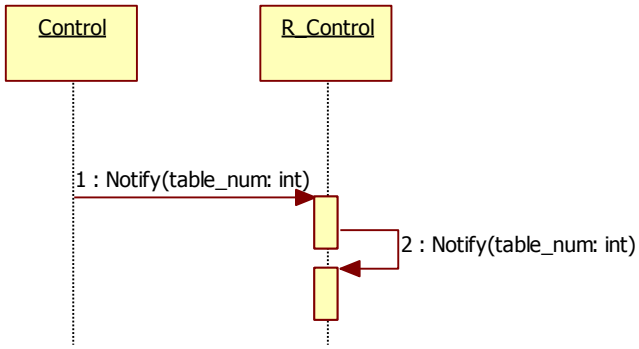
17. Play_Music



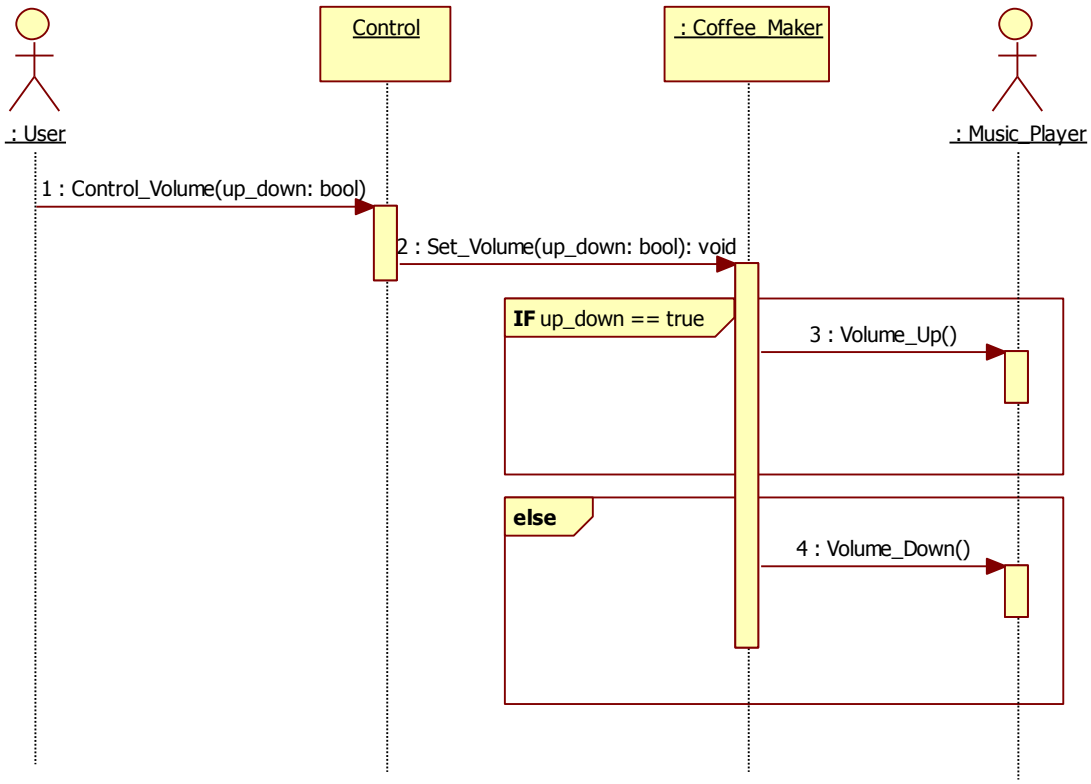
18. Move_Turn_Table



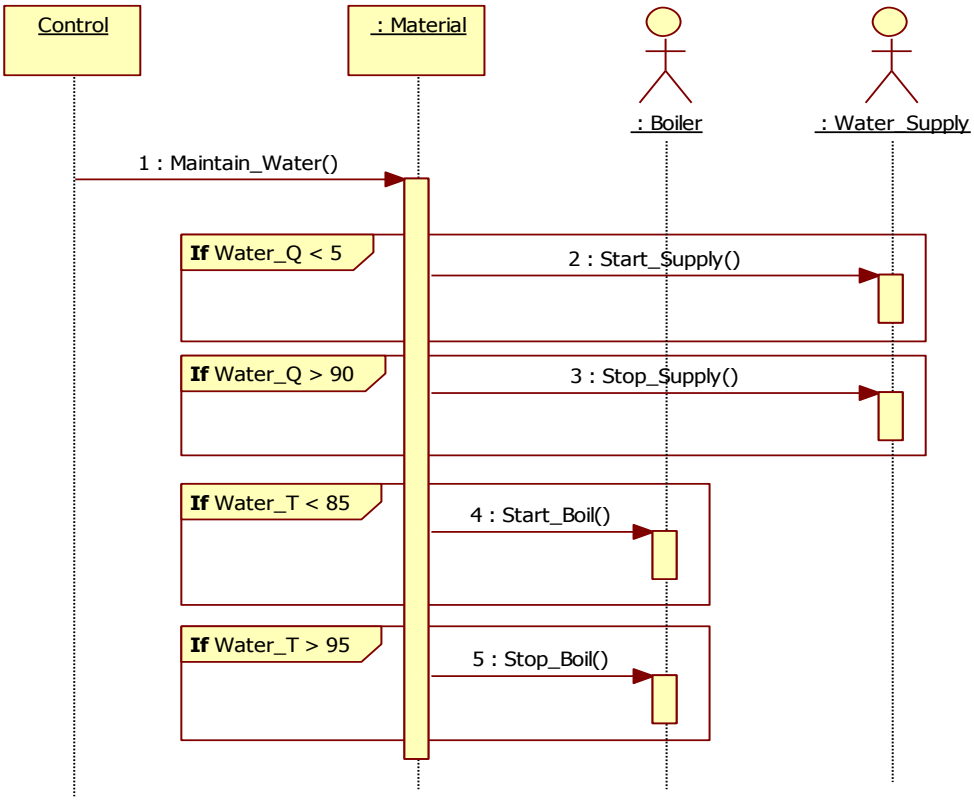
19. Notify



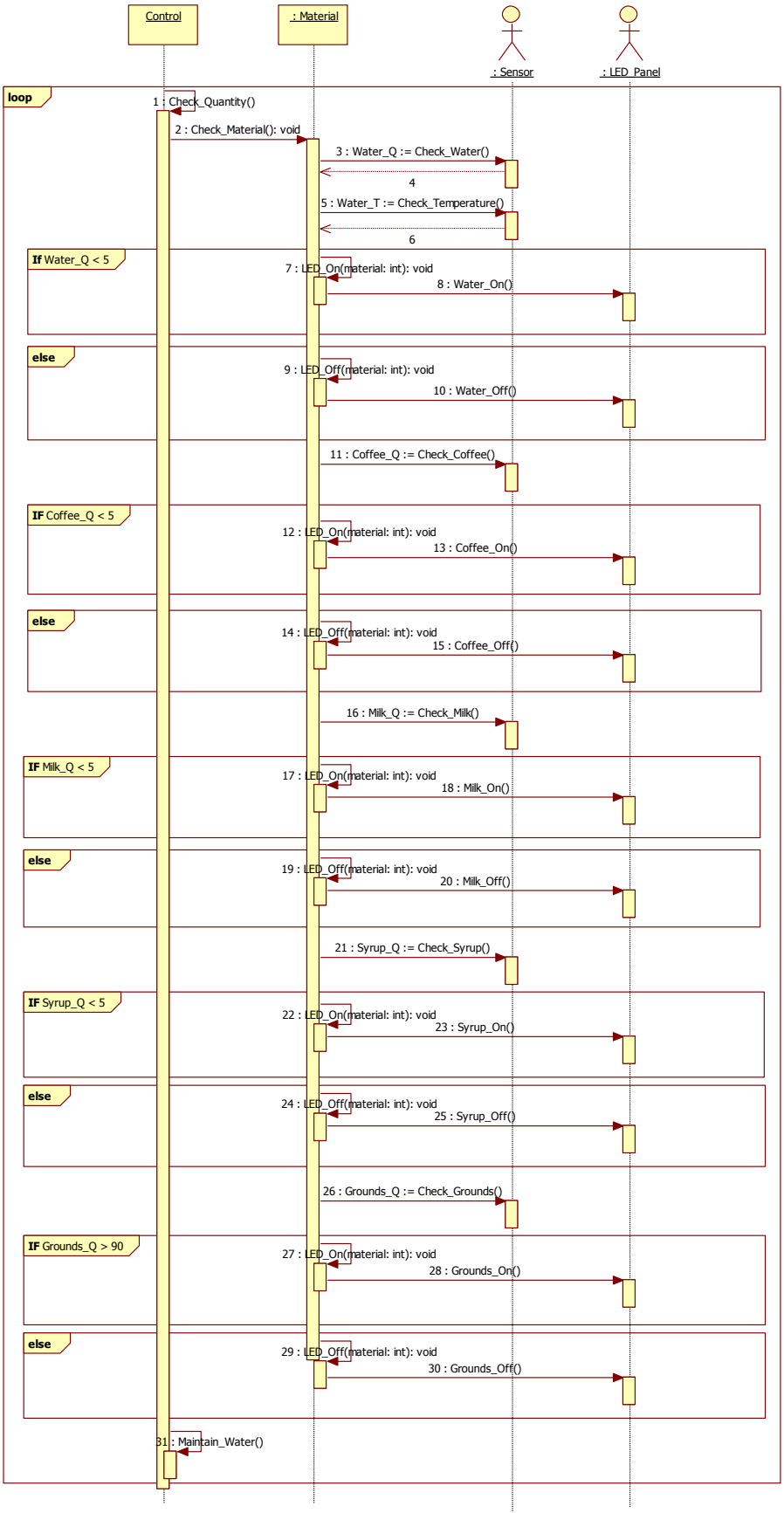
20. Control_Volume



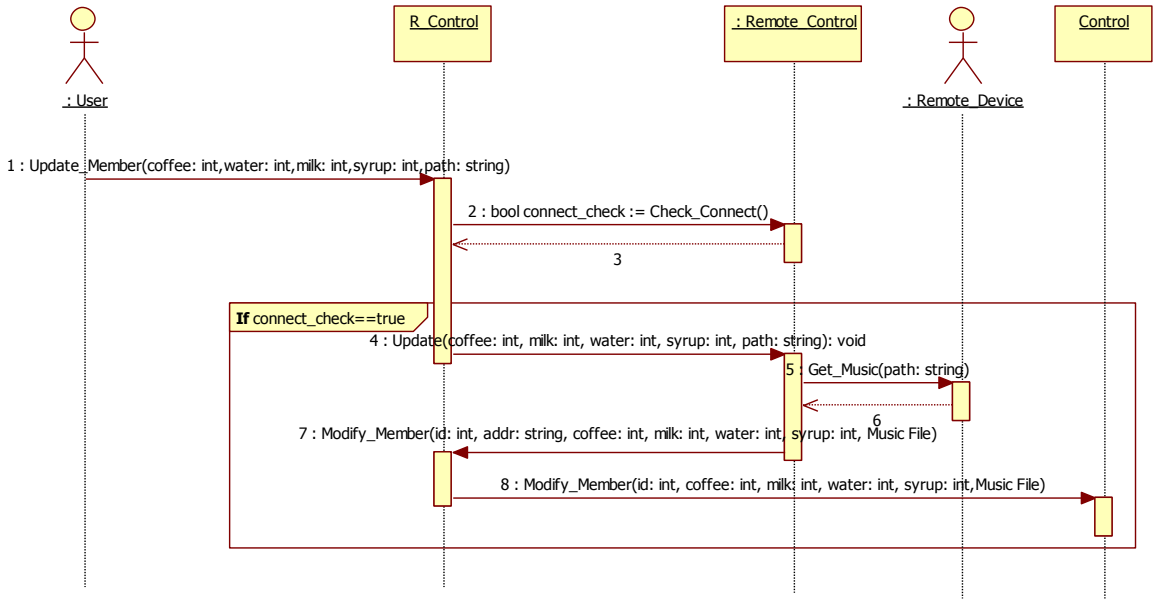
21. Maintain_Water



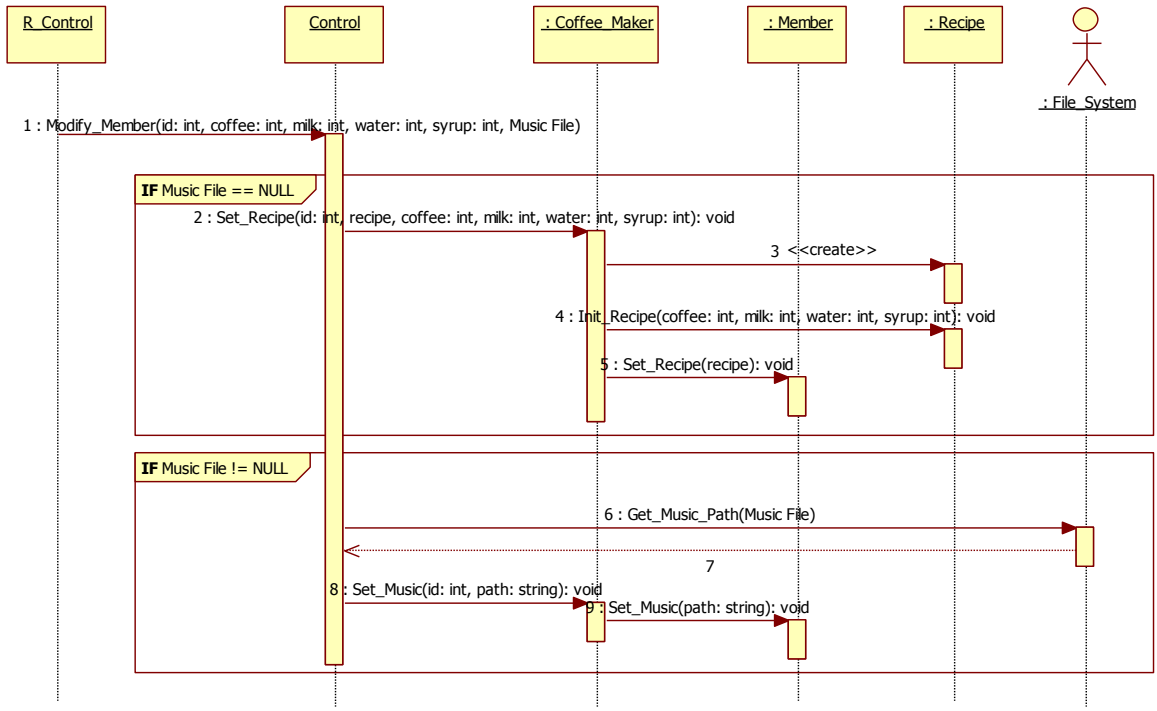
22. Check_Quantity



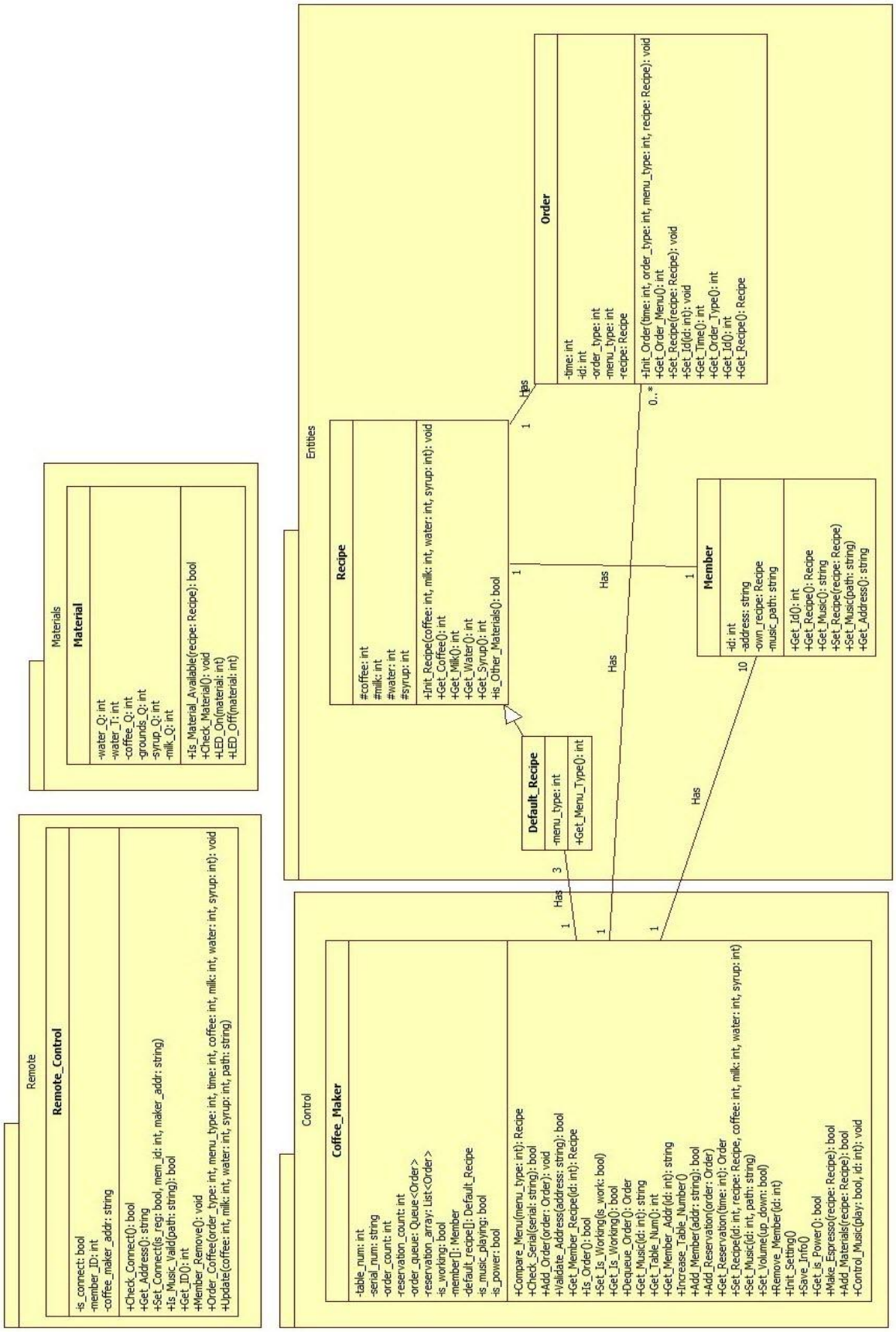
23. Update_Member



24. Modify_Member



Activity 2245. Define Design Class Diagram



◎Conclusion

OOAD의 좋았던점

- use case에서부터 시작 하기 때문에 만들어진 결과물이 사용자가 요구하는 기능을 모두 포함 하고 있다.

- 함수 각각의 기능들이 독립적이기 때문에 구현하는 사람 입장에서 전체 흐름을 파악하지 못해도 구현에 지장이 없다. 따라서 디자인하는 사람과 구현하는 사람을 분리할 수 있다.

- 두번째 cycle을 돌 때 시스템 요구사항의 변화가 있었음에도 불구하고, 첫번째 cycle에서 나온 결과물의 많은 부분을 그대로 사용할 수 있었다. 그래서 시스템을 유지보수할 때 시간과 비용을 절약할 수 있고, 점진적인 개발이 가능하다.

OOAD의 안좋았던점

- System 전체의 흐름을 파악하는 것이 어렵다.

- System boundary가 명확하지 못하면, 다음단계로의 진행이 매우 어렵다.

- 실시간으로 처리되어야 하는 부분을 표현하기가 힘들다.

- Diagram이 너무 많기 때문에, 어떤 것을 표현할 때 어떤 Diagram을 사용해야 할지 선택이 어렵고, 여러개의 Diagram을 그려야 하므로 시간이 많이 소모된다.

- 처음 OOAD를 사용하는 사람이 이해하기가 매우 어렵다.

이번 프로젝트를 통해 느낀점

- 개발할 분야의 Domain에 대한 사전지식이 매우 중요하다.

- 첫번째 cycle에서 미쳐 고려하지 못했던 사항들을 두번째 cycle에서 추가하는 것이 가능해서 보다 완벽하게 디자인 할 수 있었다.

◎ Appendix

◉ Generated code

- Coffee_Maker class

```
//
//
//   Generated by StarUML(tm) Java Add-In
//
//   @ Project : Untitled
//   @ File Name : Coffee_Maker.java
//   @ Date : 2011-05-05
//   @ Author :
//
//
```

```
public class Coffee_Maker {
        private int table_num;
        private string serial_num;
        private int order_count;
        private int reservation_count;
        private Queue<Order> order_queue;
        private List<Order> reservation_array;
        private bool is_working;
        private Member member[];
        private Default_Recipe default_recipe[];
        private bool is_music_playing;
        private bool is_power;
```

```
public Recipe Compare_Menu(int menu_type) {

}

public bool Check_Serial(string serial) {

}

public void Add_Order(Order order) {

}

public bool Validate_Address(string address) {

}

public Recipe Get_Member_Recipe(int id) {

}

public bool Is_Order() {

}

public void Set_Is_Working(bool is_work) {

}

public bool Get_Is_Working() {

}
```

```
public Order Dequeue_Order() {

}

public string Get_Music(int id) {

}

public int Get_Table_Num() {

}

public string Get_Member_Addr(int id) {

}

public void Increase_Table_Number() {

}

public bool Add_Member(string addr) {

}

public void Add_Reservation(Order order) {

}

public Order Get_Reservation(int time) {

}
```

```
        public void Set_Recipe(int id, Recipe recipe, int coffee, int milk, int water, int syrup) {

        }

        public void Set_Music(int id, string path) {

        }

        public void Set_Volume(bool up_down) {

        }

        public void Remove_Member(int id) {

        }

        public void Init_Setting() {

        }

        public void Save_Info() {

        }

        public bool Get_is_Power() {

        }

        public bool Make_Espresso(Recipe recipe) {

        }
```

```
        public bool Add_Materials(Recipe recipe) {

        }

        public void Control_Music(bool play, int id) {

        }
}
```

-Default_Recipe class

```
//
//
//   Generated by StarUML(tm) Java Add-In
//
//   @ Project : Untitled
//   @ File Name : Default_Recipe.java
//   @ Date : 2011-05-05
//   @ Author :
//
//
```

```
public class Default_Recipe extends Recipe {
        private int menu_type;
        public Coffee_Maker 1;
        public int Get_Menu_Type() {

        }
```

```
}
-Material class
//
//
//   Generated by StarUML(tm) Java Add-In
//
//   @ Project : Untitled
//   @ File Name : Material.java
//   @ Date : 2011-05-05
//   @ Author :
//
//




public class Material {
        private int water_Q;
        private int water_T;
        private int coffee_Q;
        private int grounds_Q;
        private int syrup_Q;
        private int milk_Q;
        public bool Is_Material_Available(Recipe recipe) {

        }

        public void Check_Material() {

        }

        public void LED_On(int material) {
```

```
        }

        public void LED_Off(int material) {

        }
}
```

-Member class

```
//
//
//   Generated by StarUML(tm) Java Add-In
//
//   @ Project : Untitled
//   @ File Name : Member.java
//   @ Date : 2011-05-05
//   @ Author :
//
//




public class Member {
        private int id;
        private string address;
        private Recipe own_recipe;
        private string music_path;
        public int Get_Id() {

        }
```

```
        public Recipe Get_Recipe() {

        }

        public string Get_Music() {

        }

        public void Set_Recipe(Recipe recipe) {

        }

        public void Set_Music(string path) {

        }

        public string Get_Address() {

        }
}
```

-Order class

```
//
//
//   Generated by StarUML(tm) Java Add-In
//
//   @ Project : Untitled
//   @ File Name : Order.java
//   @ Date : 2011-05-05
//   @ Author :
//
//
```

```
public class Order {
	private int time;
	private int id;
	private int order_type;
	private int menu_type;
	private Recipe recipe;
	public void Init_Order(int time, int order_type, int menu_type, Recipe recipe) {

	}

	public int Get_Order_Menu() {

	}

	public void Set_Recipe(Recipe recipe) {

	}

	public void Set_Id(int id) {

	}

	public int Get_Time() {

	}

	public int Get_Order_Type() {
```

```
        }

        public int Get_Id() {

        }

        public Recipe Get_Recipe() {

        }
}
```

-Recipe class

```
//
//
//   Generated by StarUML(tm) Java Add-In
//
//   @ Project : Untitled
//   @ File Name : Recipe.java
//   @ Date : 2011-05-05
//   @ Author :
//
//




public class Recipe {
        protected int coffee;
        protected int milk;
        protected int water;
        protected int syrup;
```

```
        public void Init_Recipe(int coffee, int milk, int water, int syrup) {

        }

        public int Get_Coffee() {

        }

        public int Get_Milk() {

        }

        public int Get_Water() {

        }

        public int Get_Syrup() {

        }

        public bool is_Other_Materials() {

        }
}
```

-Remote_Control class

```
//
//
//   Generated by StarUML(tm) Java Add-In
//
//   @ Project : Untitled
//   @ File Name : Remote_Control.java
```

```
//   @ Date : 2011-05-05
//   @ Author :
//
//




public class Remote_Control {
        private bool is_connect;
        private int member_ID;
        private string coffee_maker_addr;
        public bool Check_Connect() {

        }

        public string Get_Address() {

        }

        public void Set_Connect(bool is_reg, int mem_id, string maker_addr) {

        }

        public bool Is_Music_Vaild(string path) {

        }

        public int Get_ID() {

        }
```

```
        public void Member_Remove() {

        }

        public void Order_Coffee(int order_type, int menu_type, int time, int
coffee, int milk, int water, int syrup) {

        }

        public void Update(int coffee, int milk, int water, int syrup, string path)
{

        }
}
```