Gestire i compiti della cucina

Bettarello, Mendozza - Progetto SAS Laboratorio 22/23

Informazioni generali

Nome caso d'uso: Gestire i compiti della cucina
Portata: Sistema
Livello: Obiettivo utente
Attore primario: Chef
Parti Interessate: Chef, personale di servizio, cuoco
Pre-condizioni: L'attore deve essere autenticato come Chef
Garanzie di successo o post-condizioni: Lo chef deve aver creato un foglio riepilogativo contenente tutti i compiti della cucina per soddisfare la creazione del menu del preciso servizio.

Scenario principale di successo

#	Attore	Sistema
1	Crea il foglio riepilogativo del servizio	Compila automaticamente il foglio del servizio in oggetto per far si' che vi si trovino tutte le preparazioni e tutte le ricette previste dal menu' associato
Puo' proseguire con il passo 2, alternativamente il caso d'uso puo' terminare		
2	**Opzionalmente** ordina per difficolta' e/o importanza la lista	Viene salvato l'ordine deciso dal cuoco
3	**Opzionalmente** si informa sui turni consultando il tabellone dedicato	Viene fornito il tabellone dei turni
4	Assegna un compito specificando ricetta o preparazione, il momento in cui eseguirlo ed il cuoco che lo eseguira' ed eventualmente delle porzioni	Viene creata l'associazione tra compito, elemento della lista, orario e cuoco. Viene aggiornata la situazione turni.
5	**Opzionalmente**, assegna un cuoco ad un determinato servizio	Viene creata l'associazione tra cuoco e servizio
Torna al passo 2 fino a quando non e' soddisfatto		

Estensione 1a

#	Attore	Sistema
1a.1	Apre una foglio riepilogativo pre-esistente	Viene restituito il foglio riepilogativo pre-esistente in modalita' modifica

Estensione 1b

#	Attore	Sistema
1b.1	Elimina un foglio riepilogativo	Conferma l'eliminazione del foglio
Termine del caso d'uso		

Eccezione 1a.1a

#	Attore	Sistema
1a.1a.1	Un servizio o un turno collegato al foglio è già in esecuzione	Restituisce messaggio di errore
Termine del caso d'uso		

Eccezione 1b.1a

#	Attore	Sistema
1b.1a.1	Un servizio o un turno collegato al foglio è già in esecuzione	Restituisce messaggio di errore
Termine del caso d'uso		

Eccezione 4.1a

#	Attore	Sistema
4.1a.1	Cuoco non disponibile in quella finestra di tempo	Viene mostrata la disponibilità del cuoco

Estensione 4a

#	Attore	Sistema
4a.1	Modifica un task esistente	Viene confermata la modifica

Eccezione 4a.1a

#	Attore	Sistema
4a.1a.1	Cuoco non disponibile in quella finestra di tempo	Viene mostrata la disponibilità del cuoco

Estensione 4b

#	Attore	Sistema
4b.1	Elimina un assegnamento esistente	Viene confermata l’eliminazione

Eccezione 5.1a

#	Attore	Sistema
5.1a.1	Cuoco non disponibile in quella finestra di tempo	Viene mostrata la disponibilità del cuoco

Diagramma di sequenza di sistema

Modello di dominio

Contratti

Pre-condizione generale: attore viene identificato con un'istanza *c* di Chef

1. creaFoglioRiepilogativo(s: servizio)

pre-condizioni:

- Esiste Servizio s
- Servizio s non è in esecuzione
- Servizio s non ha nessun foglio riepilogativo assegnato

post-condizioni:
- creata un'istanza *f* di Foglio riepilogativo
- assegnata istanza f a servizio s
- *c* **proprietario di** *f*

2. ordinaLista(ordinamentoDesiderato: ordinamentoAssegnamenti)

pre-condizioni:

- *c'è un foglio f aperto in modifica*

post-condizioni:
- assegnato il tipo di ordinamento al foglio f
- ordinata la lista di assegnamenti del foglio f secondo l'ordinamento richiesto

3. consultaTabellaTurni()

pre-condizioni:
Nessuna pre-condizione

post-condizioni:
Nessuna post-condizione

4. creaAssegnamento(a: Assegnamento)

pre-condizioni:

- *c'è un foglio f aperto in modifica*
- il cuoco assegnato al task è disponibile nella finestra di tempo dell'assegnamento

post-condizioni:
- creata un'istanza a di assegnamento
- assegnato l'assegnamento a al foglio riepilogativo f

5. assegnaCuocoServizio(s: servizio, c: Cuoco)

pre-condizioni:

- *il cuoco c è disponibile durante il servizio s*

post-condizioni:

- viene assegnato il cuoco c al servizio s come cuoco di servizio

Contratti delle estensioni

1a. apreFoglioRiepilogativo(s: servizio)

pre-condizioni:

- Esiste Sevizio s
- Servizio s non è in esecuzione
- Servizio s ha un foglio riepilogativo assegnato

post-condizioni:
- aperta in modifica l'istanza f, ovvero il foglio riepilogativo del servizio s

1b. eliminaFoglioRiepilogativo(s: servizio)

pre-condizioni:

- Esiste Sevizio s
- Servizio s non è in esecuzione

post-condizioni:
- eliminata l'istanza f di foglio riepilogativo, dove f è il foglio riepilogativo assegnato a s
- eliminato assegnamento del foglio f al servizio s

4a. modificaAssegnamento (*aM: Assegnamento*)

pre-condizioni:

- c'è un foglio f aperto in modifica
- l'assegnamento aM ha l'id di un assegnamento già assegnato ad f
- il cuoco assegnato al task è disponibile nella finestra di tempo dell'assegnamento

post-condizioni:
- aggiornata l'istanza di assegnamento corrispondente all'id di aM con aM

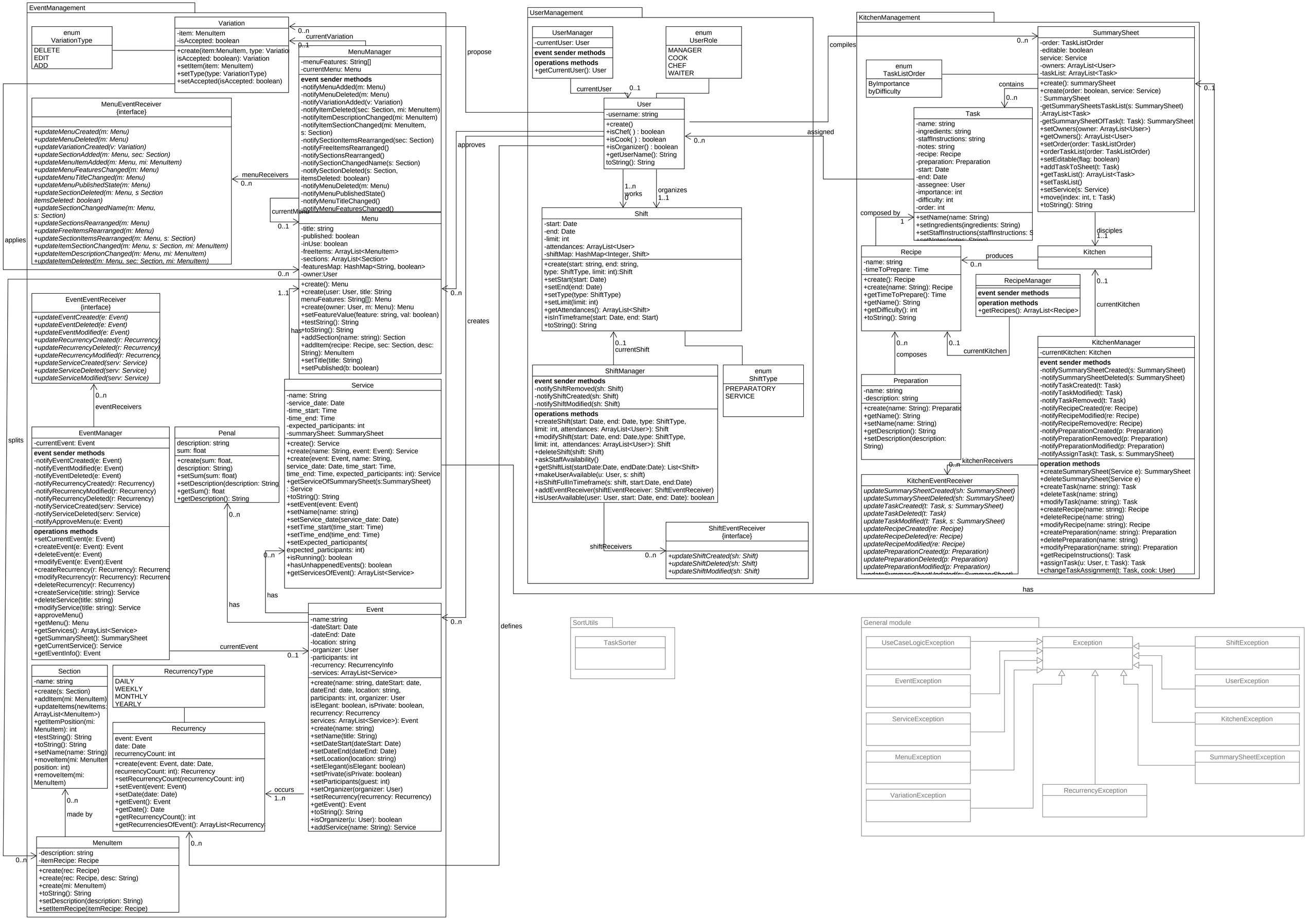
4b. eliminaAssegnamento (a: Assegnamento)

pre-condizioni:
Nessuna pre-condizione

post-condizioni:
- eliminata l'istanza a di assegnamento
- eliminato assegnamento del foglio f all'assegnamento a

Design Class Diagram

Design Class Diagram (DCD)



System Sequence Diagram

1

km: KitchenManager

s: Service

um: UserManager

createSummarySheet(s)

getCurrentUser()

user

alt

[!user.isChef()]

throw UserException.InsufficientPermissionException

[s.getSummarySheet()==null]

throw ServiceException("Service already exists")

summarySheet = new SummarySheet(TaskListOrder.ByDifficulty, s)

sumsheet: SummarySheet

notifySummarySheetCreated(summarySheet)

for each KitchenEventReceiver r of receivers

r: KitchenEventReceiver

updateSummarySheetUpdated(summarySheet)

currentSummarySheet = summarySheet

sumsheet

sumsheet

1a

km: KitchenManager

s: Service

um: UserManager

openSummarySheet(s)

getCurrentUser()

user

getSummarySheet()

sumsheet: Service

ss

alt

[user==null]

throw UserException("No user is logged in")

[!user.isChef()]

throw UserException("User is not a chef")

[ss==null]

throw ServiceException
("No summary sheet for this service")

[!ss.getOwners().contains(user)]

throw UserException
("User is not an owner of the service")

[s.getApproved_menu_id()!=null]

throw ServiceException
("Menu for this service is already approved")

currentSummarySheet = ss

ss

1b

km: KitchenManager

s: Service

um: UserManager

deleteSummarySheet(s)

getCurrentUser()

user

getSummarySheet()

ss: SummarySheet

ss

alt

[user==null]

throw UserExceprtion("No user is logged in")

[ss==null]

throw ServiceException.IsRunningException

[s.getApproved_menu_id()!=null]

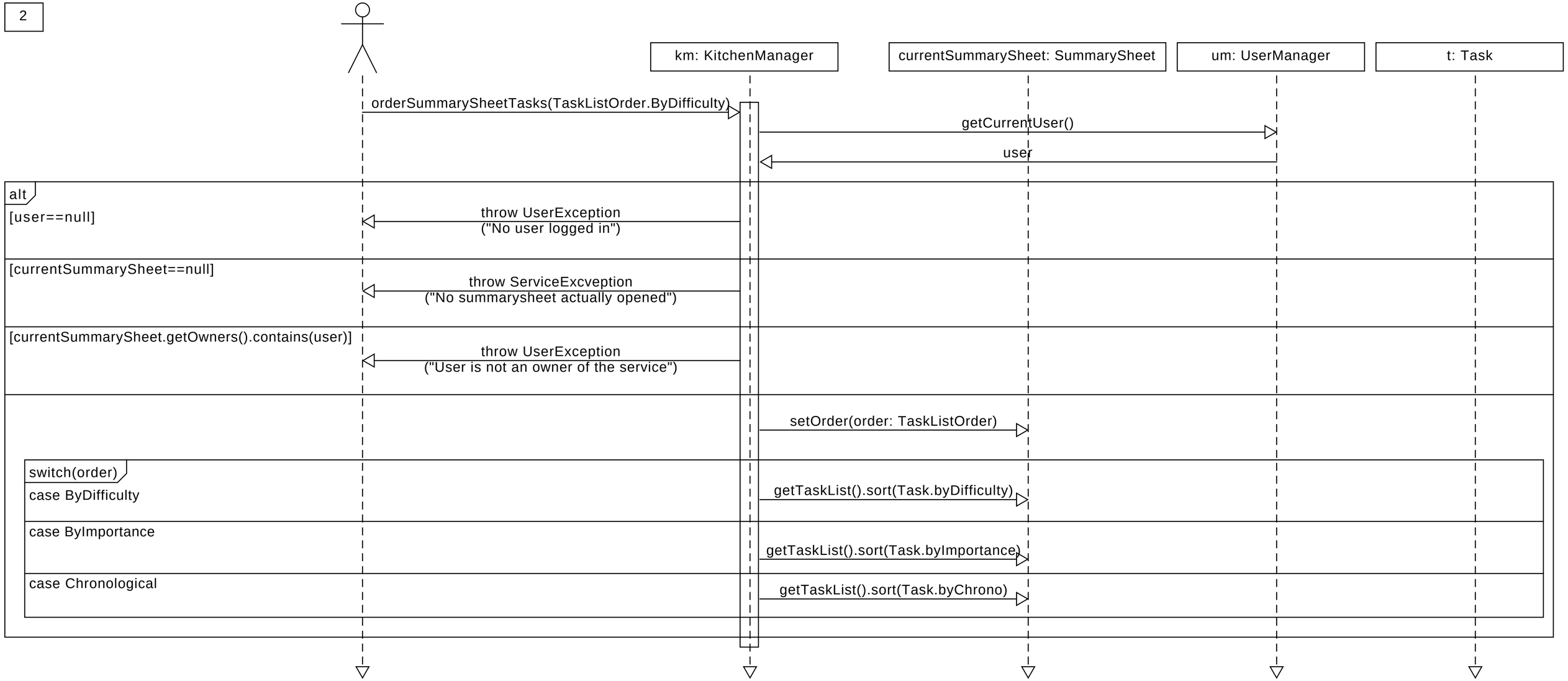
throw UserException.InsufficientPermissionException

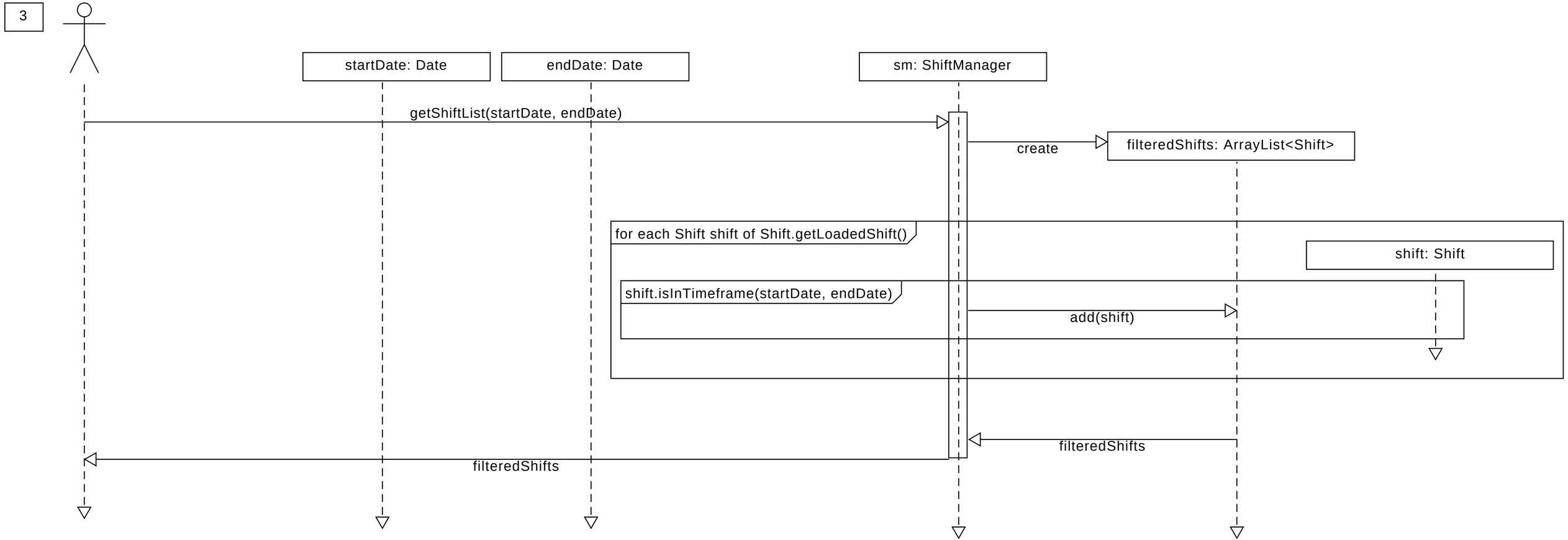
notifySummarySheetDeleted(ss)

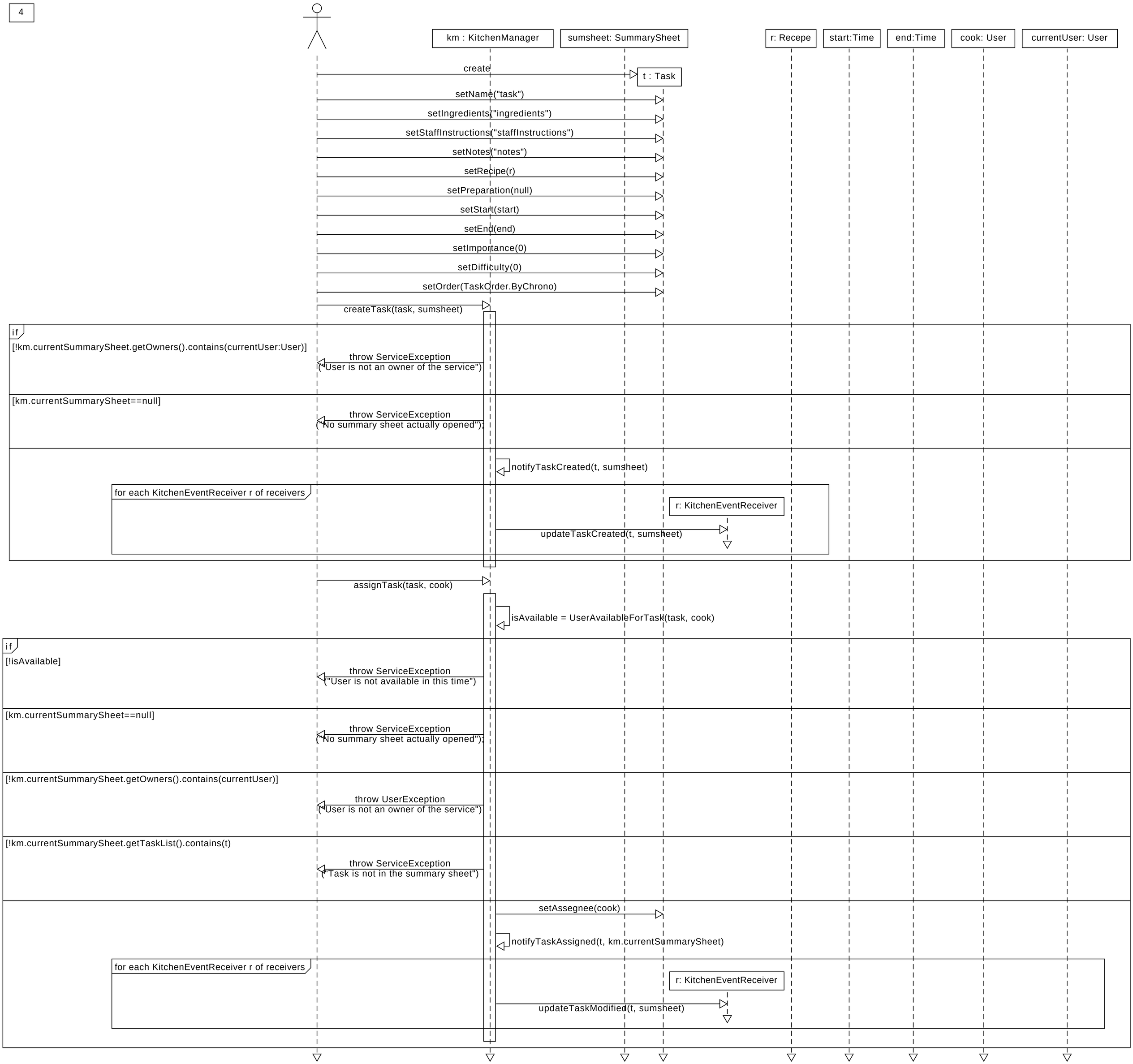
for each KitchenEventReceiver r of receivers

r: KitchenEventReceiver

updateSummarySheetUpdated(summarySheet)


2
km: KitchenManager
currentSummarySheet: SummarySheet
um: UserManager
t: Task
orderSummarySheetTasks(TaskListOrder.ByDifficulty)
getCurrentUser()
user
alt
[user==null]
throw UserException
("No user logged in")
[currentSummarySheet==null]
throw ServiceExcveption
("No summarysheet actually opened")
[currentSummarySheet.getOwners().contains(user)]
throw UserException
("User is not an owner of the service")
setOrder(order: TaskListOrder)
switch(order)
case ByDifficulty
getTaskList().sort(Task.byDifficulty)
case ByImportance
getTaskList().sort(Task.byImportance)
case Chronological
getTaskList().sort(Task.byChrono)


3
startDate: Date
endDate: Date
sm: ShiftManager
getShiftList(startDate, endDate)
create
filteredShifts: ArrayList<Shift>
for each Shift shift of Shift.getLoadedShift()
shift: Shift
shift.isInTimeframe(startDate, endDate)
add(shift)
filteredShifts
filteredShifts


km : KitchenManager
sumsheet: SummarySheet
r: Recepe
start:Time
end:Time
cook: User
currentUser: User
create
t : Task
setName("task")
setIngredients("ingredients")
setStaffInstructions("staffInstructions")
setNotes("notes")
setRecipe(r)
setPreparation(null)
setStart(start)
setEnd(end)
setImportance(0)
setDifficulty(0)
setOrder(TaskOrder.ByChrono)
createTask(task, sumsheet)
if
[!km.currentSummarySheet.getOwners().contains(currentUser:User)]
throw ServiceException
("User is not an owner of the service")
[km.currentSummarySheet==null]
throw ServiceException
("No summary sheet actually opened");
notifyTaskCreated(t, sumsheet)
for each KitchenEventReceiver r of receivers
r: KitchenEventReceiver
updateTaskCreated(t, sumsheet)
assignTask(task, cook)
isAvailable = UserAvailableForTask(task, cook)
if
[!isAvailable]
throw ServiceException
("User is not available in this time")
[km.currentSummarySheet==null]
throw ServiceException
("No summary sheet actually opened");
[!km.currentSummarySheet.getOwners().contains(currentUser)]
throw UserException
("User is not an owner of the service")
[!km.currentSummarySheet.getTaskList().contains(t)
throw ServiceException
("Task is not in the summary sheet")
setAssegnee(cook)
notifyTaskAssigned(t, km.currentSummarySheet)
for each KitchenEventReceiver r of receivers
r: KitchenEventReceiver
updateTaskModified(t, sumsheet)

4a

km : KitchenManager

sumsheet: SummarySheet

r: Recepe

start:Time

end:Time

cook: User

t : Task

getTaskList().get(0)

setName("task")

setIngredients("ingredients")

setStaffInstructions("staffInstructions")

setNotes("notes")

setRecipe(r)

setPreparation(null)

setStart(start)

setEnd(end)

setImportance(0)

setDifficulty(0)

setOrder(TaskOrder.ByChrono)

modifyTask(task)

isAvailable = UserAvailableForTask(task, cook)

if

[km.currentSummarySheet==null]

throw ServiceException
("User is not available in this time")

[!km.currentSummarySheet.getTaskList().contains(t)]

throw ServiceException
("Task is not in the summary sheet")

[!km.currentSummarySheet.getTaskList().contains(user)]

throw UserException
("User is not an owner of the service")

[!isAvailable]

throw ServiceException
("User is not available in this time")

notifyTaskModified(t, km.currentSummarySheet)

for each KitchenEventReceiver r of receivers

r: KitchenEventReceiver

updateTaskModified(t, km.currentSummarySheet)

4b

km : KitchenManager

sumsheet: SummarySheet

currentUser: User

t : Task

getTaskList().get(0)

deleteTask(t)

if

[km.currentSummarySheet==null]

throw ServiceException
("User is not available in this time")

[!km.currentSummarySheet.getTaskList().contains(t)]

throw ServiceException
("Task is not in the summary sheet")

[!km.currentSummarySheet.getTaskList().contains(user)]

throw UserException
("User is not an owner of the service")

nortifyTaskRemoved(t)

for each KitchenEventReceiver r of receivers

r: KitchenEventReceiver

updateTaskDeleted(t, sumsheet)

5

km : KitchenManager

km.currentService : Service

currentUser: User

cook: User

assignSupportCookToService(cook)

if

[currentUser==null]

throw ServiceException
("No user logged in")

[km.currentSummarySheet==null]

throw ServiceException
("No summary sheet actually opened")

[!km.isUserAvailableForService(currentService, c)]

throw ServiceException
("No summary sheet actually opened")

[!cook.isCook()]

throw ServiceException
("User is not a cook")

setSupportCook(cook)

notifyAssignSupportCook(cook, km.currentSummarySheet)

for each KitchenEventReceiver r of receivers

r: KitchenEventReceiver

updateSupportCookAssigned(km.currentSummarySheet, cook)