

# Gestire i compiti della cucina

Bettarello, Mendoza - 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
<i>Puo' proseguire con il passo 2, alternatively il caso d'uso puo' terminare</i>		
2	<b>Opzionalmente</b> ordina per difficolta' e/o importanza la lista	Viene salvato l'ordine deciso dal cuoco
3	<b>Opzionalmente</b> 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 eseguirà ed eventualmente delle porzioni	Viene creata l'associazione tra compito, elemento della lista, orario e cuoco. Viene aggiornata la situazione turni.
5	<b>Opzionalmente</b> , assegna un cuoco ad un determinato servizio	Viene creata l'associazione tra cuoco e servizio
<i>Torna al passo 2 fino a quando non e' soddisfatto</i>		

#### Estensione 1a

#	Attore	Sistema
1a.1	Apri un 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 Servizio 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 Servizio 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)**

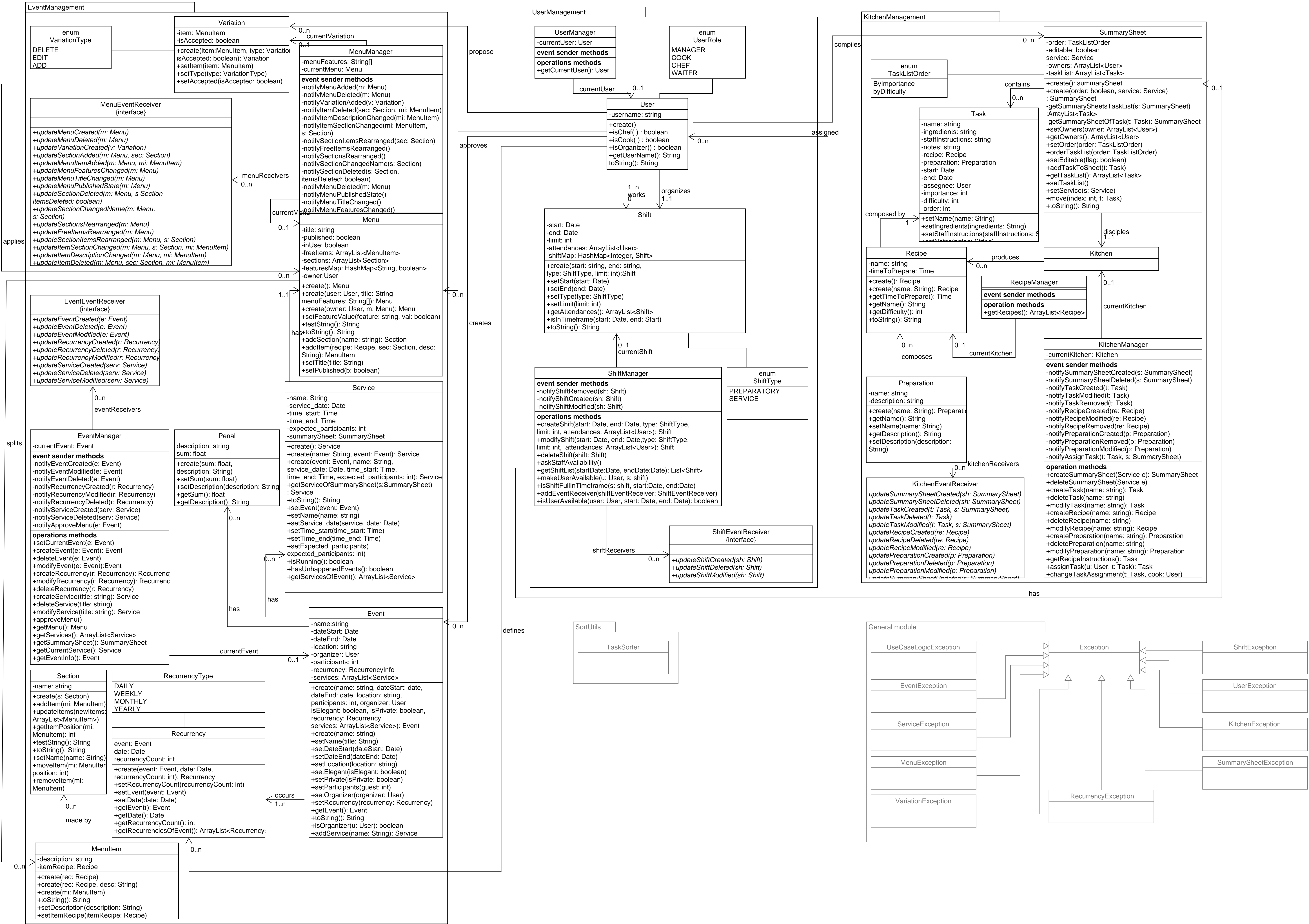
This diagram illustrates the design of a system, showing the relationships between various classes, interfaces, and enumerations. The diagram is organized into several packages: EventManagement, UserManagement, KitchenManagement, and General module.

**EventManagement Package:**

- enum VariationType**: DELETE, EDIT, ADD.
- Variation**:
  - item: MenuItem
  - isAccepted: boolean
  - +create(item: MenuItem, type: VariationType): Variation
  - +isAccepted(): boolean
  - +setItem(item: MenuItem)
  - +setType(type: VariationType)
  - +setAccepted(isAccepted: boolean)
- MenuEventManager** (interface):
  - +updateMenuCreated(m: Menu)
  - +updateMenuDeleted(m: Menu)
  - +updateVariationCreated(v: Variation)
  - +updateSectionAdded(m: Menu, sec: Section)
  - +updateMenuItemAdded(m: Menu, mi: MenuItem)
  - +updateMenuFeaturesChanged(m: Menu)
  - +updateMenuTitleChanged(m: Menu)
  - +updateMenuPublishedState(m: Menu)
  - +updateSectionDeleted(m: Menu, s: Section, itemsDeleted: boolean)
  - +updateSectionChangedName(m: Menu, s: Section)
  - +updateSectionsRearranged(m: Menu)
  - +updateFreeItemsRearranged(m: Menu)
  - +updateSectionItemsRearranged(m: Menu, s: Section)
  - +updateItemSectionChanged(m: Menu, s: Section, mi: MenuItem)
  - +updateItemDescriptionChanged(m: Menu, mi: MenuItem)
  - +updateItemDeleted(m: Menu, sec: Section, mi: MenuItem)
- MenuEventReceiver** (interface):
  - +updateMenuCreated(m: Menu)
  - +updateMenuDeleted(m: Menu)
  - +updateVariationCreated(v: Variation)
  - +updateSectionAdded(m: Menu, sec: Section)
  - +updateMenuItemAdded(m: Menu, mi: MenuItem)
  - +updateMenuFeaturesChanged(m: Menu)
  - +updateMenuTitleChanged(m: Menu)
  - +updateMenuPublishedState(m: Menu)
  - +updateSectionDeleted(m: Menu, s: Section, itemsDeleted: boolean)
  - +updateSectionChangedName(m: Menu, s: Section)
  - +updateSectionsRearranged(m: Menu)
  - +updateFreeItemsRearranged(m: Menu)
  - +updateSectionItemsRearranged(m: Menu, s: Section)
  - +updateItemSectionChanged(m: Menu, s: Section, mi: MenuItem)
  - +updateItemDescriptionChanged(m: Menu, mi: MenuItem)
  - +updateItemDeleted(m: Menu, sec: Section, mi: MenuItem)
- MenuManager**:
  - menuFeatures: String[]
  - currentMenu: Menu
  - event sender methods**:
    - notifyMenuAdded(m: Menu)
    - notifyMenuDeleted(m: Menu)
    - notifyVariationAdded(v: Variation)
    - notifyItemDeleted(sec: Section, mi: MenuItem)
    - notifyItemDescriptionChanged(mi: MenuItem, s: Section)
    - notifyItemSectionChanged(mi: MenuItem, s: Section)
    - notifySectionItemsRearranged(sec: Section)
    - notifyFreeItemsRearranged()
    - notifySectionsRearranged()
    - notifySectionChangedName(s: Section)
    - notifySectionDeleted(s: Section, itemsDeleted: boolean)
    - notifyMenuDeleted(m: Menu)
    - notifyMenuPublishedState()
    - notifyMenuTitleChanged()
    - notifyMenuFeaturesChanged()
- Menu**:
  - title: string
  - published: boolean
  - inUse: boolean
  - freeItems: ArrayList<MenuItem>
  - sections: ArrayList<Section>
  - featuresMap: HashMap<String, boolean>
  - owner: User
  - operations methods**:
    - +create(): Menu
    - +create(user: User, title: String, menuFeatures: String[]): Menu
    - +create(owner: User, m: Menu): Menu
    - +setFeatureValue(feature: string, val: boolean)
    - +testString(): String
    - +toString(): String
    - +addSection(name: string): Section
    - +addItem(recipe: Recipe, sec: Section, desc: String): MenuItem
    - +setTitle(title: String)
    - +setPublished(b: boolean)
- MenuItem**:
  - description: string
  - itemRecipe: Recipe
  - operations methods**:
    - +create(rec: Recipe)
    - +create(rec: Recipe, desc: String)
    - +create(mi: MenuItem)
    - +toString(): String
    - +setDescription(description: String)
    - +setItemRecipe(itemRecipe: Recipe)
- EventManager**:
  - currentEvent: Event
  - event sender methods**:
    - notifyEventCreated(e: Event)
    - notifyEventDeleted(e: Event)
    - notifyEventModified(e: Event)
    - notifyEventDeleted(e: Event)
    - notifyRecurrencyCreated(r: Recurrency)
    - notifyRecurrencyModified(r: Recurrency)
    - notifyRecurrencyDeleted(r: Recurrency)
    - notifyServiceCreated(serv: Service)
    - notifyServiceDeleted(serv: Service)
    - notifyApproveMenu(e: Event)
  - operations methods**:
    - +setCurrentEvent(e: Event)
    - +createEvent(e: Event): Event
    - +deleteEvent(e: Event)
    - +modifyEvent(e: Event): Event
    - +createRecurrency(r: Recurrency): Recurrency
    - +modifyRecurrency(r: Recurrency): Recurrency
    - +deleteRecurrency(r: Recurrency)
    - +createService(title: string): Service
    - +deleteService(title: string)
    - +modifyService(title: string): Service
    - +approveMenu()
    - +getMenu(): Menu
    - +getServices(): ArrayList<Service>
    - +getSummarySheet(): SummarySheet
    - +getCurrentService(): Service
    - +getEventInfo(): Event
- EventEventReceiver** (interface):
  - +updateEventCreated(e: Event)
  - +updateEventDeleted(e: Event)
  - +updateEventModified(e: Event)
  - +updateRecurrencyCreated(r: Recurrency)
  - +updateRecurrencyDeleted(r: Recurrency)
  - +updateRecurrencyModified(r: Recurrency)
  - +updateServiceCreated(serv: Service)
  - +updateServiceDeleted(serv: Service)
  - +updateServiceModified(serv: Service)
- Penal**:
  - description: string
  - sum: float
  - operations methods**:
    - +create(sum: float, description: String)
    - +setSum(sum: float)
    - +setDescription(description: String)
    - +getSum(): float
    - +getDescription(): String
- Service**:
  - name: String
  - service\_date: Date
  - time\_start: Time
  - time\_end: Time
  - expected\_participants: int
  - summarySheet: SummarySheet
  - operations methods**:
    - +create(): Service
    - +create(name: String, event: Event): Service
    - +create(event: Event, name: String, service\_date: Date, time\_start: Time, time\_end: Time, expected\_participants: int): Service
    - +getServiceOfSummarySheet(s: SummarySheet): Service
    - +toString(): String
    - +setEvent(event: Event)
    - +setName(name: string)
    - +setService\_date(service\_date: Date)
    - +setTime\_start(time\_start: Time)
    - +setTime\_end(time\_end: Time)
    - +setExpected\_participants(expected\_participants: int)
    - +isRunning(): boolean
    - +hasUnhappenedEvents(): boolean
    - +getServicesOfEvent(): ArrayList<Service>
- Event**:
  - name: string
  - dateStart: Date
  - dateEnd: Date
  - location: string
  - organizer: User
  - participants: int
  - recurrency: RecurrencyInfo
  - services: ArrayList<Service>
  - operations methods**:
    - +create(name: string, dateStart: date, dateEnd: date, location: string, participants: int, organizer: User, isElegant: boolean, isPrivate: boolean, recurrency: Recurrency, services: ArrayList<Service>): Event
    - +create(name: string)
    - +setName(title: String)
    - +setDateStart(dateStart: Date)
    - +setDateEnd(dateEnd: Date)
    - +setLocation(location: string)
    - +setElegant(isElegant: boolean)
    - +setPrivate(isPrivate: boolean)
    - +setParticipants(guest: int)
    - +setOrganizer(organizer: User)
    - +setRecurrency(recurrency: Recurrency)
    - +getEvent(): Event
    - +getDate(): Date
    - +getRecurrencyCount(): int
    - +getRecurrenciesOfEvent(): ArrayList<Recurrency
- RecurrencyType**: DAILY, WEEKLY, MONTHLY, YEARLY.
- Recurrency**:
  - event: Event
  - date: Date
  - recurrencyCount: int
  - operations methods**:
    - +create(event: Event, date: Date, recurrencyCount: int): Recurrency
    - +setRecurrencyCount(recurrencyCount: int)
    - +setEvent(event: Event)
    - +setDate(date: Date)
    - +getEvent(): Event
    - +getDate(): Date
    - +getRecurrencyCount(): int
    - +getRecurrenciesOfEvent(): ArrayList<Recurrency
- MenuManager** (interface):
  - +updateMenuCreated(m: Menu)
  - +updateMenuDeleted(m: Menu)
  - +updateVariationCreated(v: Variation)
  - +updateSectionAdded(m: Menu, sec: Section)
  - +updateMenuItemAdded(m: Menu, mi: MenuItem)
  - +updateMenuFeaturesChanged(m: Menu)
  - +updateMenuTitleChanged(m: Menu)
  - +updateMenuPublishedState(m: Menu)
  - +updateSectionDeleted(m: Menu, s: Section, itemsDeleted: boolean)
  - +updateSectionChangedName(m: Menu, s: Section)
  - +updateSectionsRearranged(m: Menu)
  - +updateFreeItemsRearranged(m: Menu)
  - +updateSectionItemsRearranged(m: Menu, s: Section)
  - +updateItemSectionChanged(m: Menu, s: Section, mi: MenuItem)
  - +updateItemDescriptionChanged(m: Menu, mi: MenuItem)
  - +updateItemDeleted(m: Menu, sec: Section, mi: MenuItem)
- MenuEventReceiver** (interface):
  - +updateMenuCreated(m: Menu)
  - +updateMenuDeleted(m: Menu)
  - +updateVariationCreated(v: Variation)
  - +updateSectionAdded(m: Menu, sec: Section)
  - +updateMenuItemAdded(m: Menu, mi: MenuItem)
  - +updateMenuFeaturesChanged(m: Menu)
  - +updateMenuTitleChanged(m: Menu)
  - +updateMenuPublishedState(m: Menu)
  - +updateSectionDeleted(m: Menu, s: Section, itemsDeleted: boolean)
  - +updateSectionChangedName(m: Menu, s: Section)
  - +updateSectionsRearranged(m: Menu)
  - +updateFreeItemsRearranged(m: Menu)
  - +updateSectionItemsRearranged(m: Menu, s: Section)
  - +updateItemSectionChanged(m: Menu, s: Section, mi: MenuItem)
  - +updateItemDescriptionChanged(m: Menu, mi: MenuItem)
  - +updateItemDeleted(m: Menu, sec: Section, mi: MenuItem)
- MenuManager**:
  - menuFeatures: String[]
  - currentMenu: Menu
  - event sender methods**:
    - notifyMenuAdded(m: Menu)
    - notifyMenuDeleted(m: Menu)
    - notifyVariationAdded(v: Variation)
    - notifyItemDeleted(sec: Section, mi: MenuItem)
    - notifyItemDescriptionChanged(mi: MenuItem, s: Section)
    - notifyItemSectionChanged(mi: MenuItem, s: Section)
    - notifySectionItemsRearranged(sec: Section)
    - notifyFreeItemsRearranged()
    - notifySectionsRearranged()
    - notifySectionChangedName(s: Section)
    - notifySectionDeleted(s: Section, itemsDeleted: boolean)
    - notifyMenuDeleted(m: Menu)
    - notifyMenuPublishedState()
    - notifyMenuTitleChanged()
    - notifyMenuFeaturesChanged()
- Menu**:
  - title: string
  - published: boolean
  - inUse: boolean
  - freeItems: ArrayList<MenuItem>
  - sections: ArrayList<Section>
  - featuresMap: HashMap<String, boolean>
  - owner: User
  - operations methods**:
    - +create(): Menu
    - +create(user: User, title: String, menuFeatures: String[]): Menu
    - +create(owner: User, m: Menu): Menu
    - +setFeatureValue(feature: string, val: boolean)
    - +testString(): String
    - +toString(): String
    - +addSection(name: string): Section
    - +addItem(recipe: Recipe, sec: Section, desc: String): MenuItem
    - +setTitle(title: String)
    - +setPublished(b: boolean)
- MenuItem**:
  - description: string
  - itemRecipe: Recipe
  - operations methods**:
    - +create(rec: Recipe)
    - +create(rec: Recipe, desc: String)
    - +create(mi: MenuItem)
    - +toString(): String
    - +setDescription(description: String)
    - +setItemRecipe(itemRecipe: Recipe)

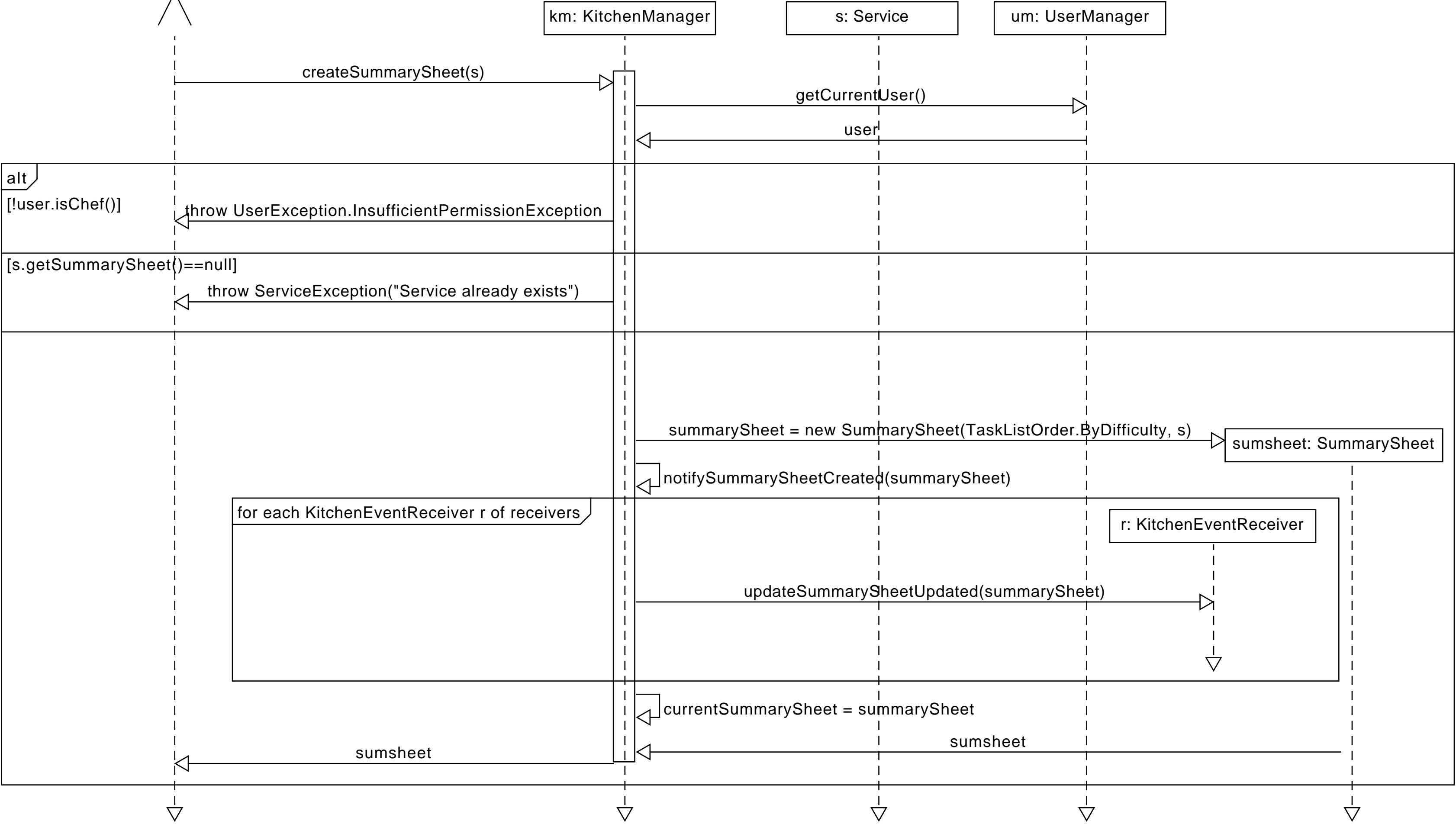
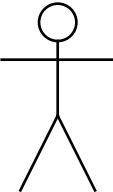
**UserManagement Package:**

- UserManager**:
  - currentUser: User
  - event sender methods**:
    - notifyUserCreated(u: User)
    - notifyUserDeleted(u: User)
    - notifyUserModified(u: User)
  - operations methods**:
    - +getCurrentUser(): User
- User**:
  - username: string
  - operations methods**:
    - +create()
    - +isChef(): boolean
    - +isCook(): boolean
    - +isOrganizer(): boolean
    - +getUserRole(): UserRole
    - +toString(): String
- Shift**:
  - start: Date
  - end: Date
  - limit: int
  - attendances: ArrayList<User>
  - shiftMap: HashMap<Integer, Shift>
  - operations methods**:
    - +create(start: string, end: string, type: ShiftType, limit: int): Shift
    - +setStart(start: Date)
    - +setEnd(end: Date)
    - +setType(type: ShiftType)
    - +setLimit(limit: int)
    - +getAttendances(): ArrayList<Shift>



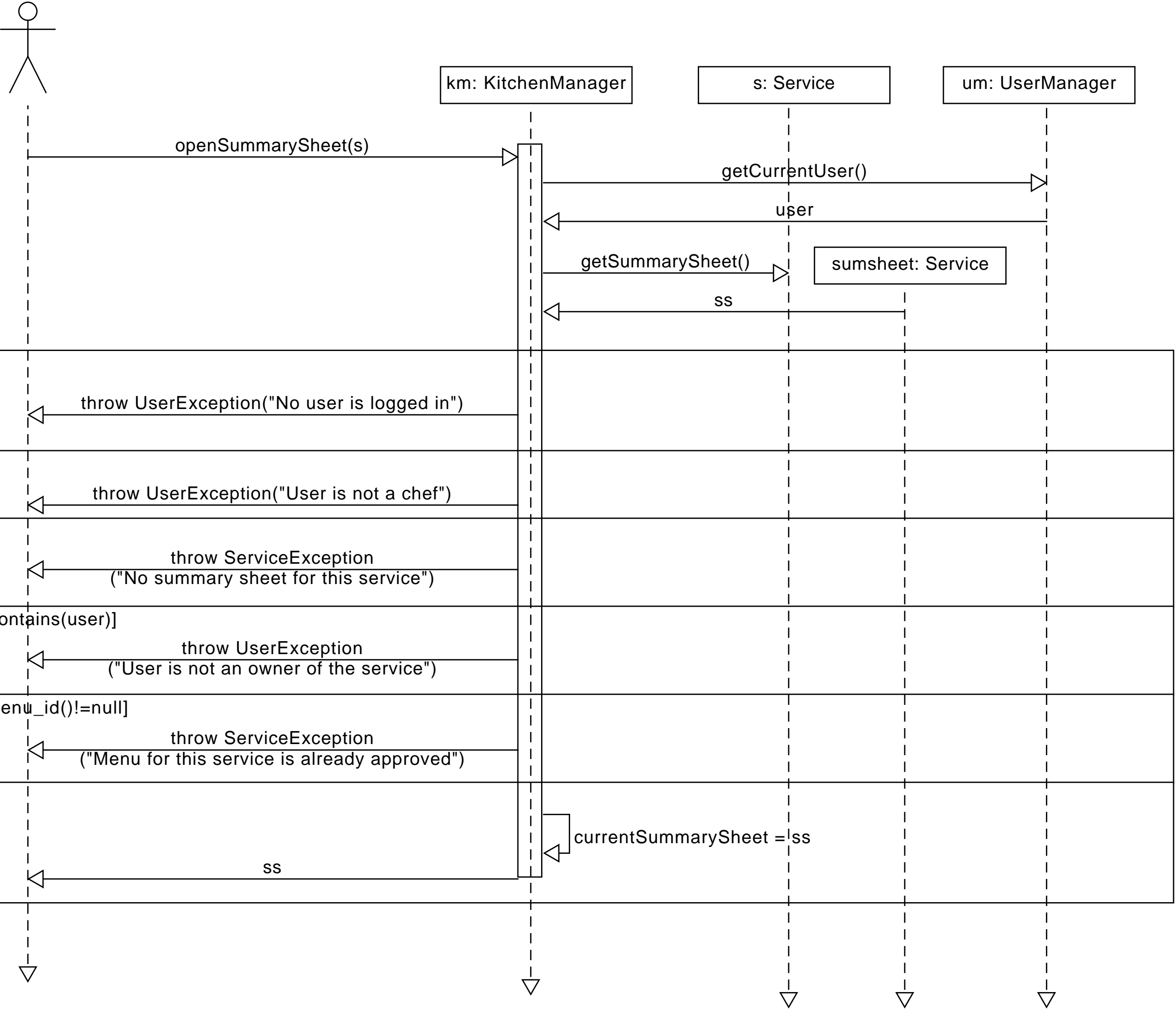
# System Sequence Diagram



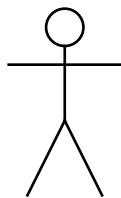




1a



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]

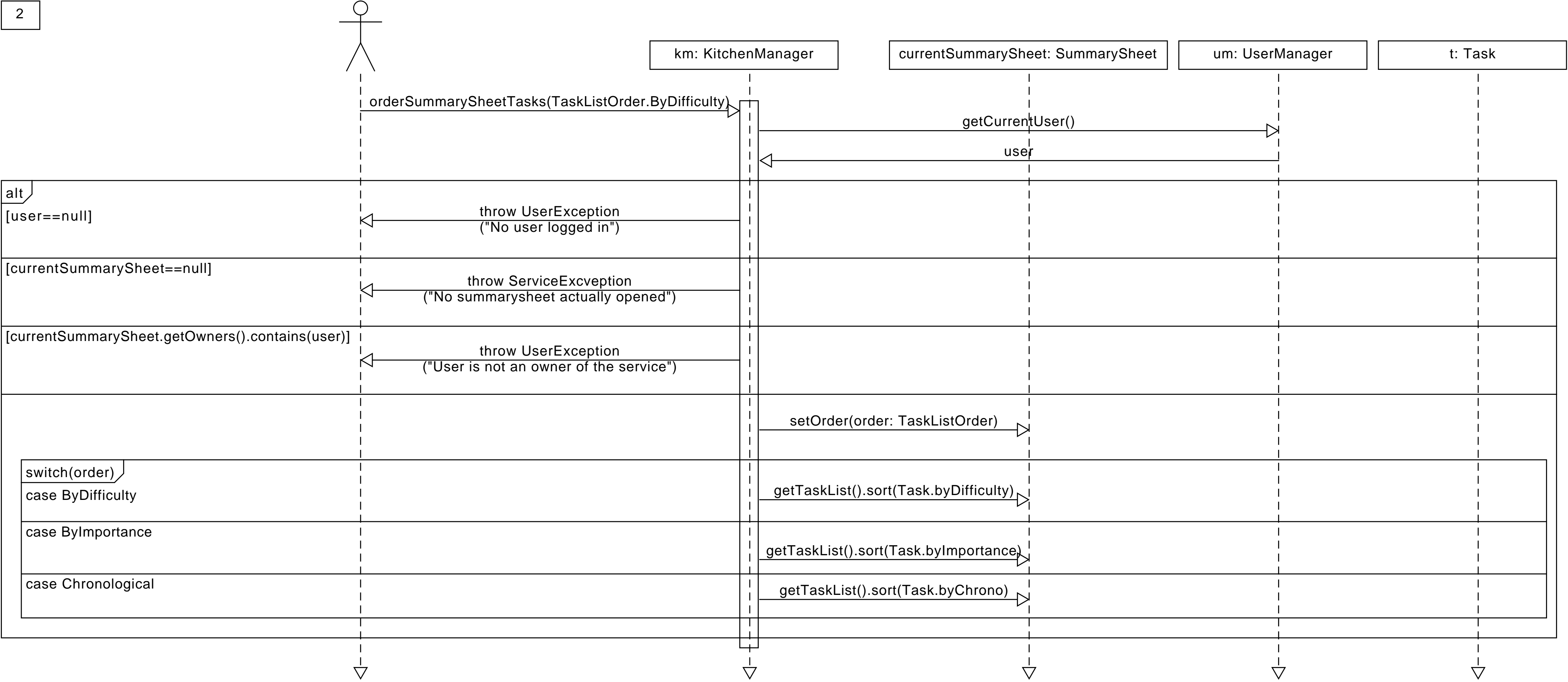
throw UserException.InsufficientPermissionException

notifySummarySheetDeleted(ss)

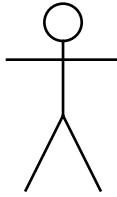
for each KitchenEventReceiver r of receivers

r: KitchenEventReceiver

updateSummarySheetUpdated(summarySheet)



3



startDate: Date

endDate: Date

sm: ShiftManager

filteredShifts: ArrayList<Shift>

shift: Shift

getShiftList(startDate, endDate)

create

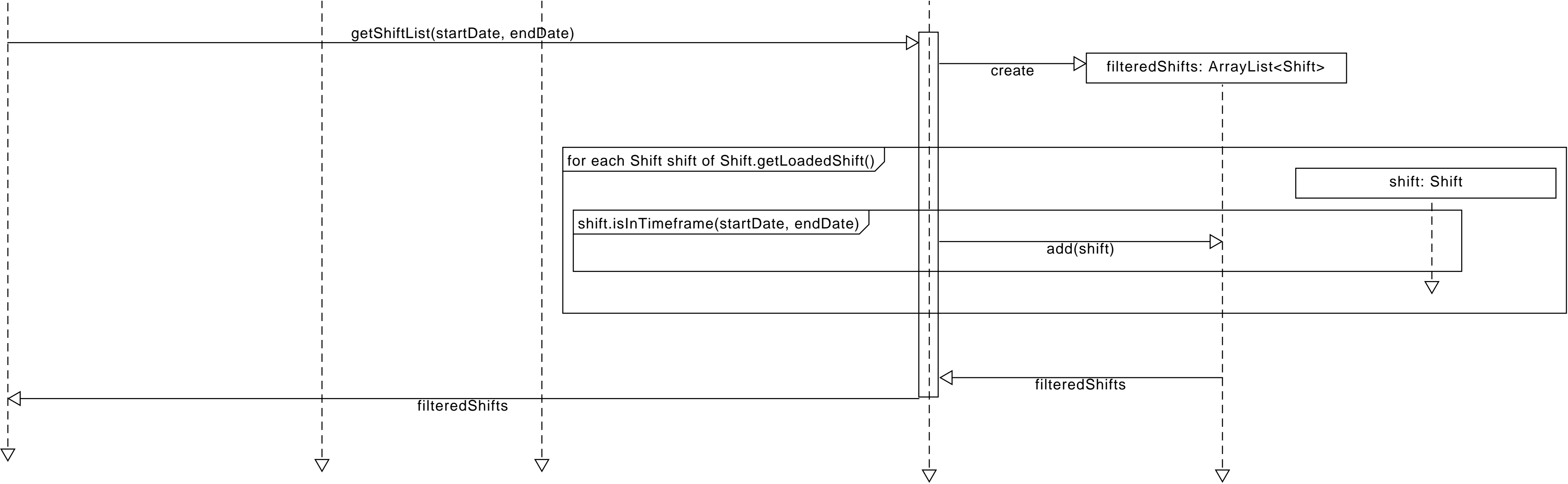
for each Shift shift of Shift.getLoadedShift()

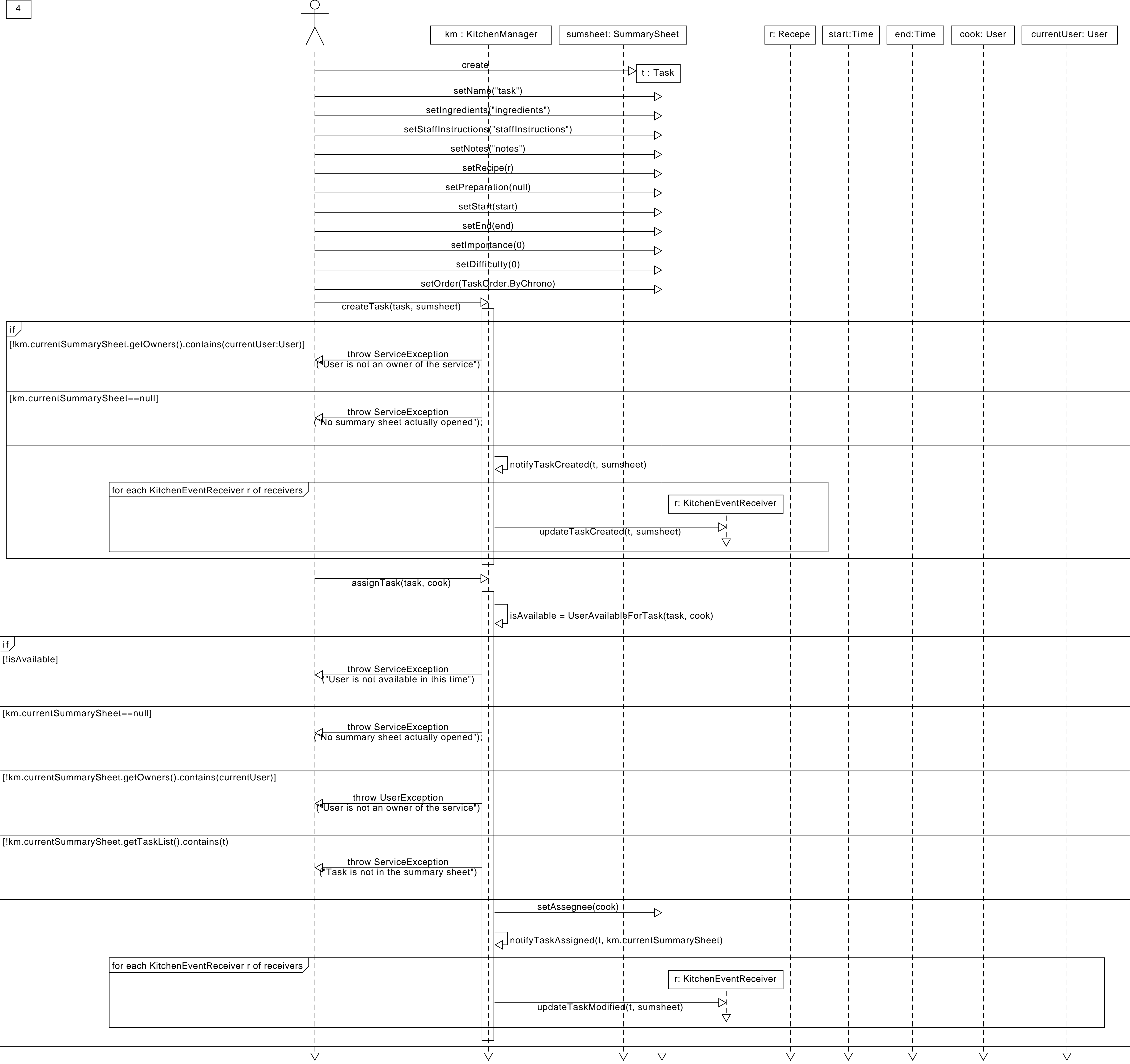
shift.isInTimeframe(startDate, endDate)

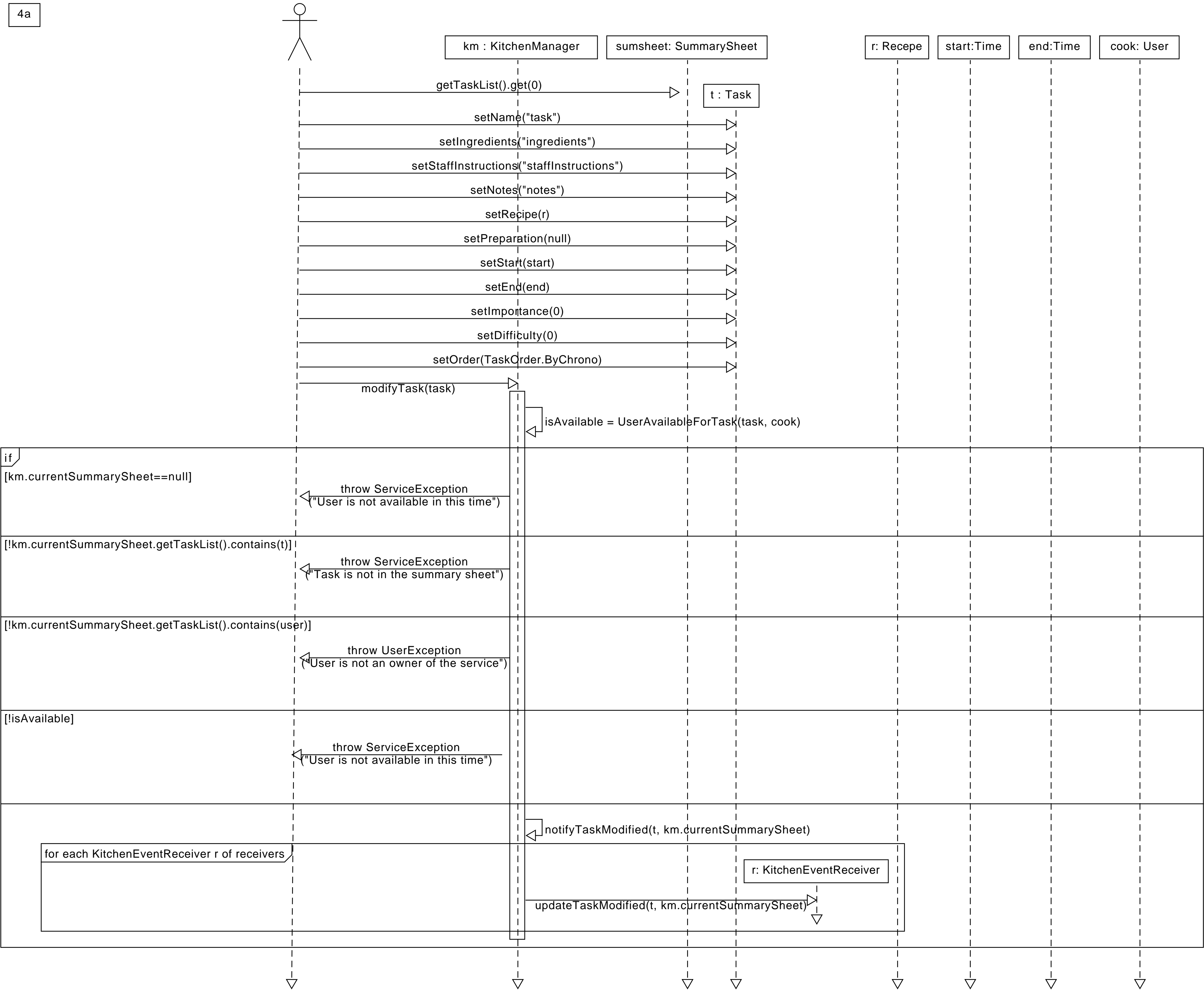
add(shift)

filteredShifts

filteredShifts







4 b

