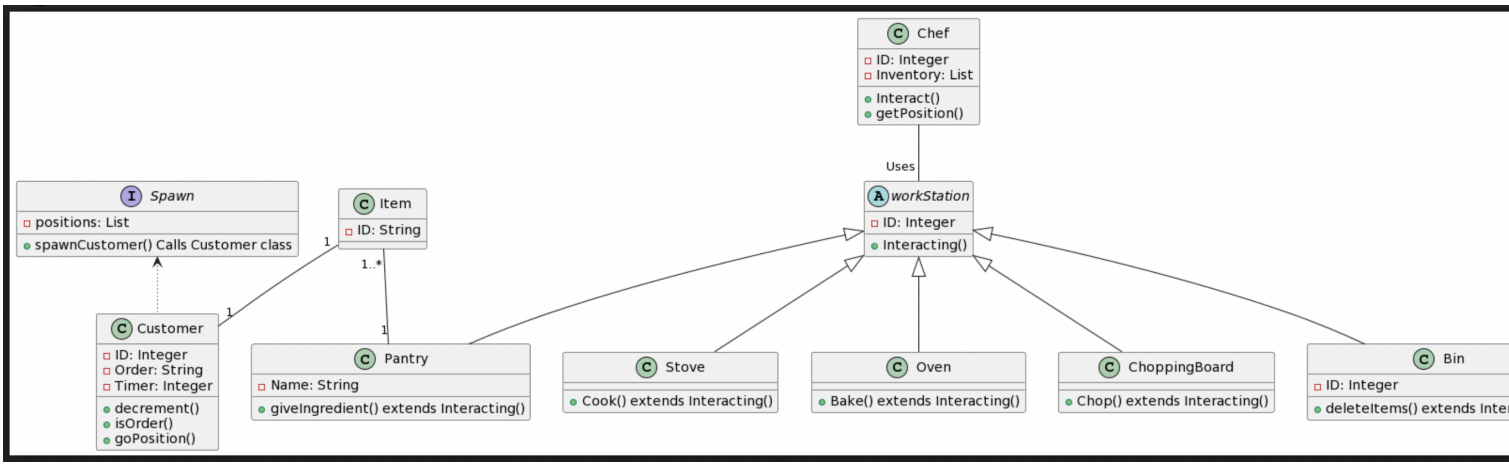
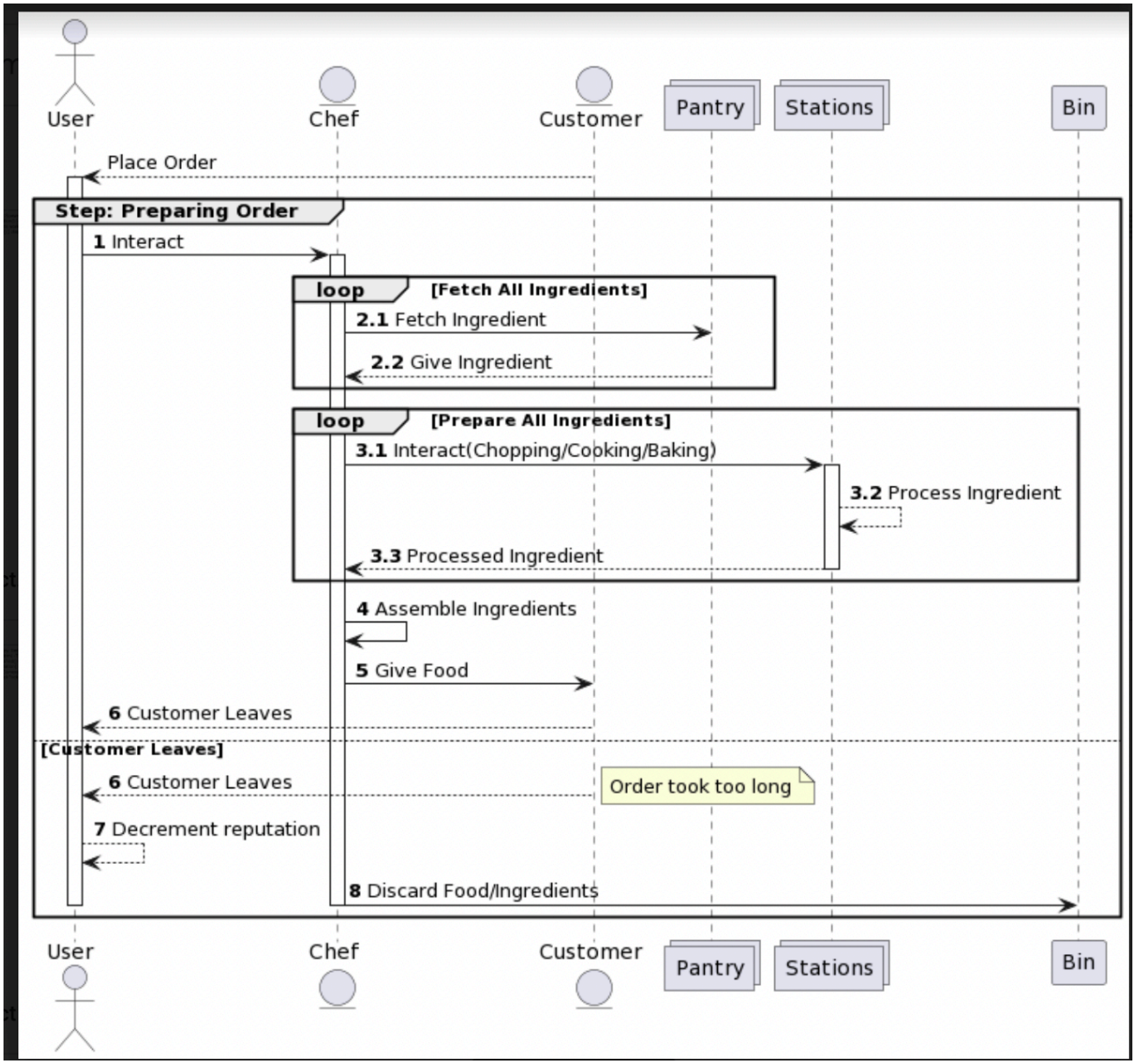
```
@startuml

interface Spawn{
	-positions: List

	+spawnCustomer() Calls Customer class
}
Spawn <-[Dotted]- Customer

class Chef{
	-ID: Integer
	-Inventory: List


	+Interact()
	+getPosition()
}

class Customer{
	-ID: Integer
	-Order: String
	-Timer: Integer

	+decrement()
	+isOrder()
	+goPosition()
}

Item"1" -- "1" Customer

class Item{
	-ID: String

}
Item "1..*" -- "1" Pantry

abstract class workStation{
	-ID: Integer

	+Interacting()
}
Chef -- "Uses" workStation


class Stove extends workStation{

	+Cook() extends Interacting()
}

class Oven extends workStation{

	+Bake() extends Interacting()
}

class ChoppingBoard extends workStation{

	+Chop() extends Interacting()
}

class Pantry extends workStation{
	-Name: String

	+giveIngredient() extends Interacting()
}

class Bin extends workStation{
	-ID: Integer

	+deleteItems() extends Interacting()
}

@enduml
```

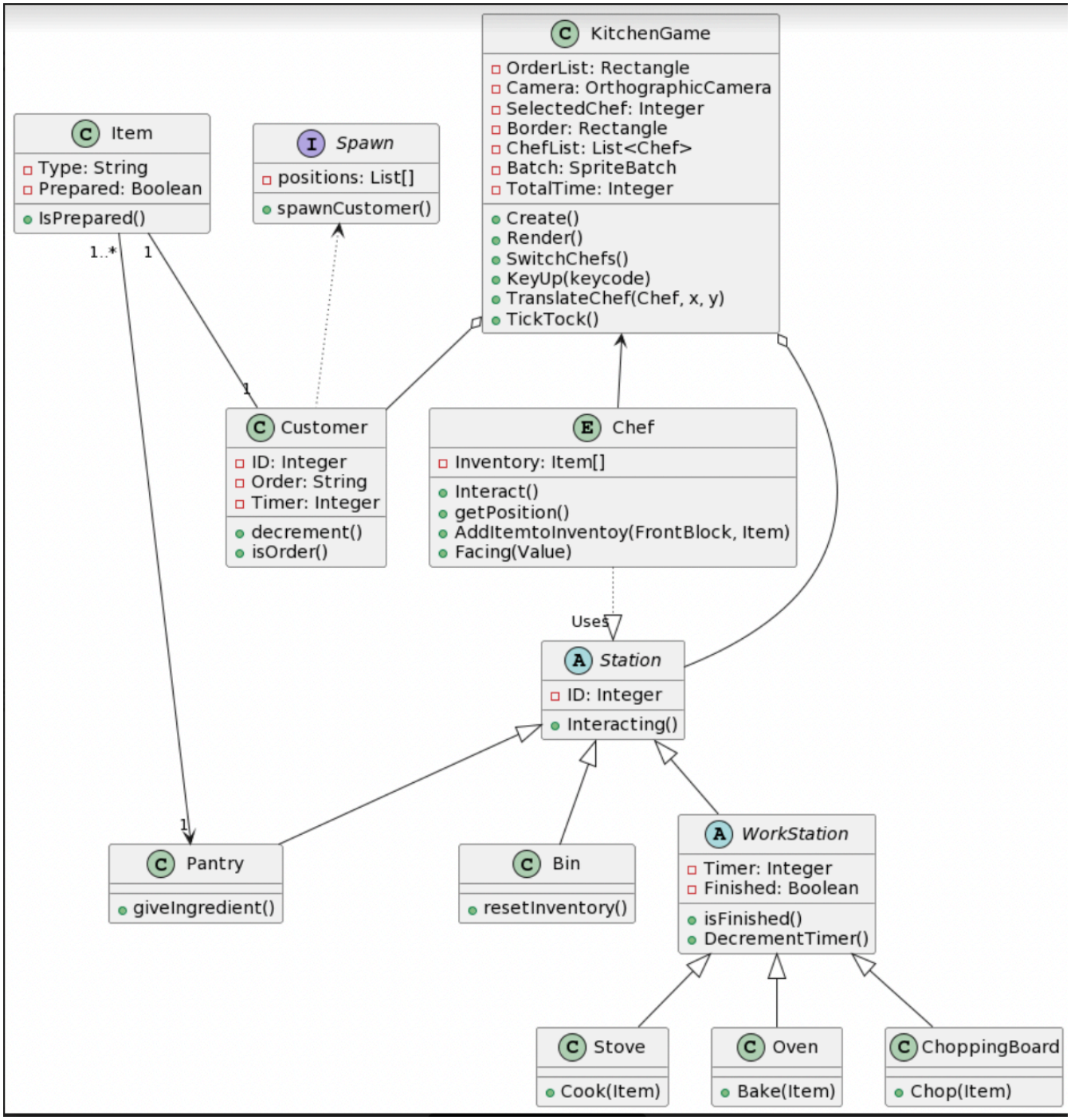

Chef
ID: Integer
Inventory: List
Interact()
getPosition()
Uses
Spawn
positions: List
spawnCustomer() Calls Customer class
Item
ID: String
workStation
ID: Integer
Interacting()
1
1..*
1
1
Customer
ID: Integer
Order: String
Timer: Integer
decrement()
isOrder()
goPosition()
Pantry
Name: String
giveIngredient() extends Interacting()
Stove
Cook() extends Interacting()
Oven
Bake() extends Interacting()
ChoppingBoard
Chop() extends Interacting()
Bin
ID: Integer
deleteItems() extends Inte


User
Chef
Customer
Pantry
Stations
Bin
Place Order
Step: Preparing Order
1 Interact
loop
[Fetch All Ingredients]
2.1 Fetch Ingredient
2.2 Give Ingredient
loop
[Prepare All Ingredients]
3.1 Interact(Chopping/Cooking/Baking)
3.2 Process Ingredient
3.3 Processed Ingredient
4 Assemble Ingredients
5 Give Food
6 Customer Leaves
[Customer Leaves]
6 Customer Leaves
Order took too long
7 Decrement reputation
8 Discard Food/Ingredients
User
Chef
Customer
Pantry
Stations
Bin

```
@startuml
actor User
entity Chef
entity Customer
collections Pantry
collections Stations
participant Bin

Customer --> User++: Place Order

group Step: Preparing Order
    autonumber
    User -> Chef ++: Interact

    'loop
    loop Fetch All Ingredients
    autonumber 2.1
    Chef-> Pantry: Fetch Ingredient
    Pantry --> Chef: Give Ingredient
    end

    'loop
    loop Prepare All Ingredients
    autonumber 3.1
    Chef-> Stations ++: Interact(Chopping/Cooking/Baking)
    Stations --> Stations: Process Ingredient
    Stations --> Chef --: Processed Ingredient
    end

    autonumber 4
    Chef -> Chef: Assemble Ingredients
    Chef-> Customer: Give Food
    Customer --> User: Customer Leaves
else Customer Leaves
    autonumber 6
    Customer --> User: Customer Leaves
    note right: Order took too long
    User --> User: Decrement reputation
    Chef -> Bin --: Discard Food/Ingredients
    User --
end



@enduml
```

```
@startuml

interface Spawn{
      -positions: List[]

      +spawnCustomer()

}

entity Chef{

      -Inventory: Item[]

      +Interact()
      +getPosition()
      +AddItemtoInventoy(FrontBlock, Item)
      +Facing(Value)
}

class Customer{
      -ID: Integer
      -Order: String
      -Timer: Integer

      +decrement()
      +isOrder()
}

class Item{
      -Type: String
      -Prepared: Boolean

      +IsPrepared()

}

abstract class Station{
      -ID: Integer

      +Interacting()
}

abstract class WorkStation extends Station{
      -Timer: Integer
      -Finished: Boolean

      +isFinished()
      +DecrementTimer()
}

Chef -- "Uses" Station
```

```
class Stove extends WorkStation{

      +Cook(Item)
}

class Oven extends WorkStation{

      +Bake(Item)
}

class ChoppingBoard extends WorkStation{


      +Chop(Item)
}

class Pantry extends Station{

      +giveIngredient()
}

class Bin extends Station{

      +resetInventory()
}

class KitchenGame{
      -OrderList: Rectangle
      -Camera: OrthographicCamera
      -SelectedChef: Integer
      -Border: Rectangle
      -ChefList: List<Chef>
      -Batch: SpriteBatch
      -TotalTime: Integer


      +Create()
      +Render()
      +SwitchChefs()
      +KeyUp(keycode)
      +TranslateChef(Chef, x, y)
      +TickTock()
}

KitchenGame <-- Chef
KitchenGame o-- Customer
KitchenGame o-- Station

Spawn <-[Dotted]- Customer
Item"1" -- "1" Customer
Item"1..*" --> "1" Pantry



@enduml
```


KitchenGame
OrderList: Rectangle
Camera: OrthographicCamera
SelectedChef: Integer
Border: Rectangle
ChefList: List<Chef>
Batch: SpriteBatch
TotalTime: Integer
Create()
Render()
SwitchChefs()
KeyUp(keycode)
TranslateChef(Chef, x, y)
TickTock()
Item
Type: String
Prepared: Boolean
IsPrepared()
Spawn
positions: List[]
spawnCustomer()
1..*
1
1
Customer
ID: Integer
Order: String
Timer: Integer
decrement()
isOrder()
Chef
Inventory: Item[]
Interact()
getPosition()
AddItemtoInventoy(FrontBlock, Item)
Facing(Value)
Uses
Station
ID: Integer
Interacting()
1
Pantry
giveIngredient()
Bin
resetInventory()
WorkStation
Timer: Integer
Finished: Boolean
isFinished()
DecrementTimer()
Stove
Cook(Item)
Oven
Bake(Item)
ChoppingBoard
Chop(Item)