

# Requirements:

## Team 14

Joshua Beaven ([jb2825@york.ac.uk](mailto:jb2825@york.ac.uk))

Calum Feenan ([cf1218@york.ac.uk](mailto:cf1218@york.ac.uk))

Harry Hillman ([ht512@york.ac.uk](mailto:ht512@york.ac.uk))

Jonathon Snelgrove ([js3336@york.ac.uk](mailto:js3336@york.ac.uk))

Omar Alhosani ([oa560@york.ac.uk](mailto:oa560@york.ac.uk))

Rayan Butt ([rb1770@york.ac.uk](mailto:rb1770@york.ac.uk))

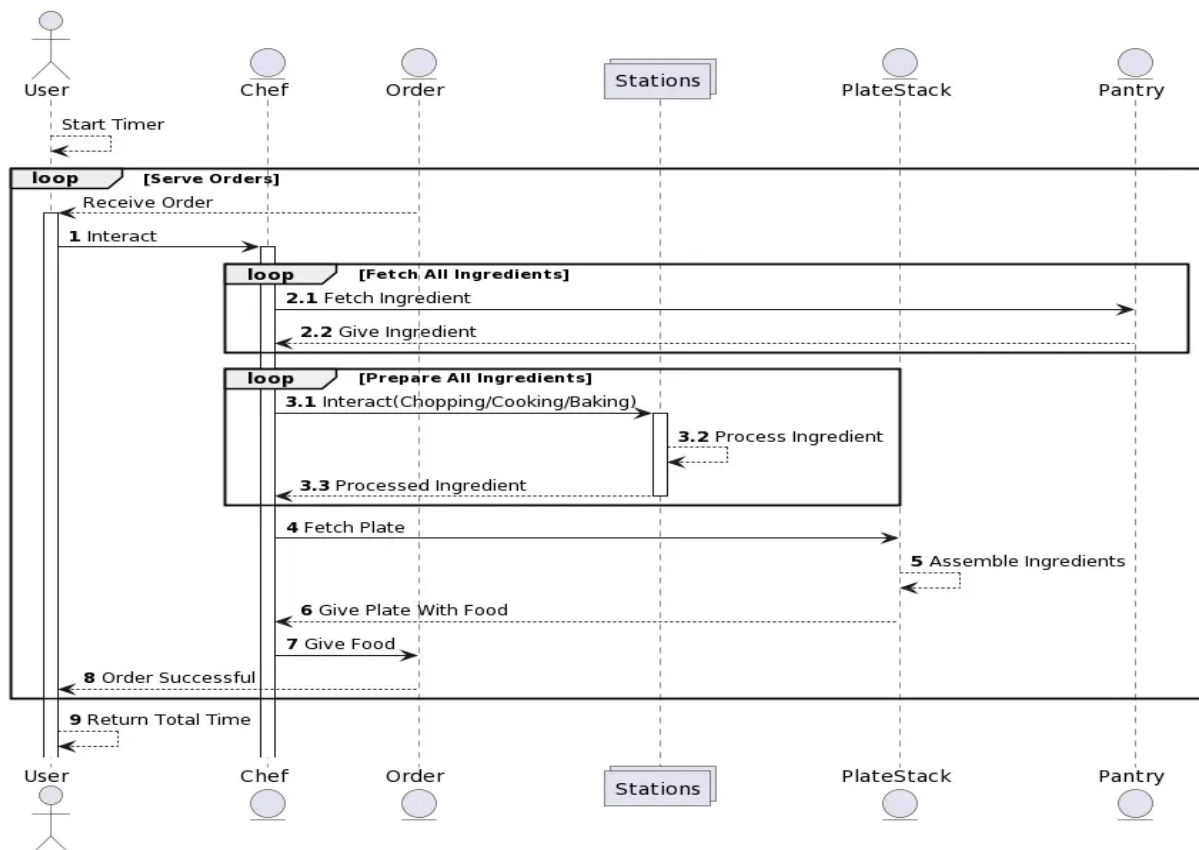
## Part A: Structural and Behavioural Diagrams

Since taking an object-oriented system (OOS) approach, as the project is being coded in Java, the Unified Modelling Language (UML) was used for the architectural representation of the product as a language specialised to visually represent OOS. Including its flexible and comprehensive properties, UML has a base foundation for modelling the more basic parts of object-oriented software.

Both structural and behavioural architecture were made using PlantUML, an open-source tool that turns plain text into different types of UML diagrams. PlantUML is supported by different software development environments like IntelliJ and Visual Studio Code.

To visualise the software's flow and behaviour a sequence diagram was used to illustrate a series of sequential steps over time, representing the actual gameplay. Using sequence diagrams reduces confusion and provides a logical representation of the game's intended software.

### Behavioural Diagram



PlantUML provides icons and a proper visual representation of the software, thus giving future teams a better understanding of the code and reducing the amount of time wasted in the process of transitioning to this project.

```

classDiagram
    class MenuType {
        START GAME
        PAUSE
    }
    class StationType {
        GRILL
        OVEN
        CUTTING_BOARD
        BIN
        DELIVERY_POINT
        PLATE_STACK
        PANTRY
    }
    class Order {
        Item: ArrayList<Integer>
        x: Integer
        medium: Random
        MaxItems: Integer
        LISI: Integer
        LISIARR: Integer
        isCompleted: Boolean
        noTimer: Boolean
        ItemList: int[]
        timeWaited: Integer
        OrderBackGroundPath: String
        successTexture: Texture
        failTexture: Texture
        textOrderForeground: Texture
        orderTexture: Texture
        OrderScreenWidth: Integer
        OrderScreenHeight: Integer
        OrderTimeElapsed: Boolean noTimer
        compareTo(int IC)
        initialiseMeals()
        genicCompleted()
        setisCompleted(boolean completed)
        getHeight()
        getWidth()
        combineTextures(Texture background, Texture foreground)
        displayRespPlate_MeatType deliveredMeal()
        getTexture()
        getTimeWaited()
        incrementTimeWaited()
        getTimeArrived()
    }
    class Chef {
        tilePosition: Point
        cameraPosition: Point
        width: Integer
        height: Integer
        chefTextures: ArrayList<Texture>
        direction: Facing
        inventory: Inventory
        chefNumber: Integer
        create()
        render()
        dispose()
        initialiseTileGrid()
        setTileAdjacency(int n, int m)
        SwitchChef()
        collisionCheck(Chef PlayerChef)
        VerifyInteract(Chef PlayerChef)
        StateInteract(Chef Player)
        AddStationProcessString imagePath, int x, int y
        AddStationProgress(int x, int y)
        RemoveStationProcess(int x, int y)
        RemoveStationProgress(int x, int y)
        UpdateInventoryTextures()
        KeyUpOrKeycode()
        AnyTypedChar character()
        TranslateChef(Chef, x, y)
        TickTock()
    }
    class Tile {
        tileStation: Station
        position: Point
        containsCounter: Boolean
        somethingAboveMe: Boolean
        somethingRightOfMe: Boolean
        somethingLeftOfMe: Boolean
        tile(int x, int y)
        Tile(int x, int y, boolean containsCounter)
        ContainsStation()
        getTileStation()
        setTileStation(station value)
        getContainsPosition()
        setContainsPosition(int x, int y)
        getGridId()
        setGridId(int x)
        getGridY()
        setGridY(int y)
        CanMove(Chef Facing direction)
        CanInteract(Chef Facing direction)
        DoesExist(Point tilepos)
        AnythingThere(Chef Facing direction)
        setAboveMe(boolean value)
        setRightOfMe(boolean value)
        setUnderMe(boolean value)
        setLeftOfMe(boolean value)
        setStation()
        setStation(tileStation)
    }
    class Inventory {
        inventoryFull: Boolean
        inventoryEmpty: Boolean
        items: ArrayList<Item>
        Inventory()
        add(Item item)
        removeFromInventory()
        removeItem(index)
        removeItem(item)
        size()
        isEmpty()
        isNotFull()
        validateInventory()
        containsInventoryType()
    }
    class Bin {
        BinID: Integer
        stationType: StationType
        Bin()
        Interact(Inventory inventory)
        equals(StationType type)
        incrementProgress()
    }
    class DeliveryPoint {
        DeliveryPointID: Integer
        stationType: StationType
        meal: Plate
        DeliveryPoint()
        Interact(Inventory inventory)
        equals(StationType type)
        confirmDelivery()
        getDeliveredMealType()
        getProgress()
        incrementProgress()
    }
    class Grill {
        GrillID: Integer
        isProcessing: Boolean
        contents: Ingredient
        stationType: StationType
        progress: Integer
        Grill()
        isSlowDelivered(item item)
        Interact(Inventory items)
        setResults(Inventory items)
        getProcessing()
        getContents()
        equals(StationType type)
        getProgress()
        incrementProgress()
    }
    class Oven {
        OvenID: Integer
        Interact(Inventory items)
        isProcessing: Boolean
        contents: Ingredient
        stationType: StationType
        getProgress()
        getContents()
        equals(StationType type)
        incrementProgress()
    }
    class CuttingBoard {
        CuttingBoardID: Integer
        isProcessing: Boolean
        contents: Ingredient
        stationType: StationType
        progress: Integer
        CuttingBoard()
        isSlowDelivered(item item)
        Interact(Inventory items)
        setResults(Inventory items)
        getProcessing()
        getContents()
        equals(StationType type)
        getProgress()
        incrementProgress()
    }
    class PlateStack {
        PlateStackID: Integer
        stationType: StationType
        PlateStack()
        Interact(Inventory items)
        equals(StationType type)
        containsMeal(Inventory items)
        getProgress()
        incrementProgress()
    }
    class Pantry {
        pantryType: Ingredient.IngredientType
        PantryID: Integer
        stationType: StationType
        Pantry()
        Interact(Inventory inventory)
        getProgress()
        incrementProgress()
    }
    class Ingredient {
        isCooked: Boolean
        isChopped: Boolean
        isPrepared: Boolean
        isCookedAndChopped: Boolean
        isPreparedAndChopped: Boolean
        checkIfPrepared()
        isChopped()
        getIngredientType()
        getMealType()
    }
    class Facing {
        UP
        RIGHT
        DOWN
        LEFT
        value: Integer
        map: HashMap<>
        Facing(int value): Facing
        valueOf(int Facing): Facing
        getValue()
    }
    class MealType {
        HAMBURGER
        SALAD
        value: Integer
        map: HashMap<>
        VALUES: List<MealType>
        SIZE: Integer
        RANDOM: Random
        MealType(int value)
        valuesOf(int mealType)
        getValue()
        SIZE()
        randomMeal()
    }
    MenuType "1" --> "1" StationType
    StationType "1" --> "1" Order
    StationType "1" --> "1" Chef
    StationType "1" --> "1" Tile
    StationType "1" --> "1" Inventory
    StationType "1" --> "1" Bin
    StationType "1" --> "1" DeliveryPoint
    StationType "1" --> "1" Grill
    StationType "1" --> "1" Oven
    StationType "1" --> "1" CuttingBoard
    StationType "1" --> "1" PlateStack
    StationType "1" --> "1" Pantry
    StationType "1" --> "1" Ingredient
    Chef "1" --> "1" Order
    Chef "1" --> "1" Tile
    Chef "1" --> "1" Inventory
    Chef "1" --> "1" Bin
    Chef "1" --> "1" DeliveryPoint
    Chef "1" --> "1" Grill
    Chef "1" --> "1" Oven
    Chef "1" --> "1" CuttingBoard
    Chef "1" --> "1" PlateStack
    Chef "1" --> "1" Pantry
    Chef "1" --> "1" Ingredient
    Tile "1" --> "1" Order
    Tile "1" --> "1" Inventory
    Tile "1" --> "1" Bin
    Tile "1" --> "1" DeliveryPoint
    Tile "1" --> "1" Grill
    Tile "1" --> "1" Oven
    Tile "1" --> "1" CuttingBoard
    Tile "1" --> "1" PlateStack
    Tile "1" --> "1" Pantry
    Tile "1" --> "1" Ingredient
    Inventory "1" --> "1" Order
    Inventory "1" --> "1" Bin
    Inventory "1" --> "1" DeliveryPoint
    Inventory "1" --> "1" Grill
    Inventory "1" --> "1" Oven
    Inventory "1" --> "1" CuttingBoard
    Inventory "1" --> "1" PlateStack
    Inventory "1" --> "1" Pantry
    Inventory "1" --> "1" Ingredient
    Bin "1" --> "1" Order
    Bin "1" --> "1" Inventory
    Bin "1" --> "1" DeliveryPoint
    Bin "1" --> "1" Grill
    Bin "1" --> "1" Oven
    Bin "1" --> "1" CuttingBoard
    Bin "1" --> "1" PlateStack
    Bin "1" --> "1" Pantry
    Bin "1" --> "1" Ingredient
    DeliveryPoint "1" --> "1" Order
    DeliveryPoint "1" --> "1" Inventory
    DeliveryPoint "1" --> "1" Bin
    DeliveryPoint "1" --> "1" Grill
    DeliveryPoint "1" --> "1" Oven
    DeliveryPoint "1" --> "1" CuttingBoard
    DeliveryPoint "1" --> "1" PlateStack
    DeliveryPoint "1" --> "1" Pantry
    DeliveryPoint "1" --> "1" Ingredient
    Grill "1" --> "1" Order
    Grill "1" --> "1" Inventory
    Grill "1" --> "1" Bin
    Grill "1" --> "1" DeliveryPoint
    Grill "1" --> "1" Oven
    Grill "1" --> "1" CuttingBoard
    Grill "1" --> "1" PlateStack
    Grill "1" --> "1" Pantry
    Grill "1" --> "1" Ingredient
    Oven "1" --> "1" Order
    Oven "1" --> "1" Inventory
    Oven "1" --> "1" Bin
    Oven "1" --> "1" DeliveryPoint
    Oven "1" --> "1" Grill
    Oven "1" --> "1" CuttingBoard
    Oven "1" --> "1" PlateStack
    Oven "1" --> "1" Pantry
    Oven "1" --> "1" Ingredient
    CuttingBoard "1" --> "1" Order
    CuttingBoard "1" --> "1" Inventory
    CuttingBoard "1" --> "1" Bin
    CuttingBoard "1" --> "1" DeliveryPoint
    CuttingBoard "1" --> "1" Grill
    CuttingBoard "1" --> "1" Oven
    CuttingBoard "1" --> "1" PlateStack
    CuttingBoard "1" --> "1" Pantry
    CuttingBoard "1" --> "1" Ingredient
    PlateStack "1" --> "1" Order
    PlateStack "1" --> "1" Inventory
    PlateStack "1" --> "1" Bin
    PlateStack "1" --> "1" DeliveryPoint
    PlateStack "1" --> "1" Grill
    PlateStack "1" --> "1" Oven
    PlateStack "1" --> "1" CuttingBoard
    PlateStack "1" --> "1" Pantry
    PlateStack "1" --> "1" Ingredient
    Pantry "1" --> "1" Order
    Pantry "1" --> "1" Inventory
    Pantry "1" --> "1" Bin
    Pantry "1" --> "1" DeliveryPoint
    Pantry "1" --> "1" Grill
    Pantry "1" --> "1" Oven
    Pantry "1" --> "1" CuttingBoard
    Pantry "1" --> "1" PlateStack
    Pantry "1" --> "1" Ingredient
    Ingredient "1" --> "1" Order
    Ingredient "1" --> "1" Inventory
    Ingredient "1" --> "1" Bin
    Ingredient "1" --> "1" DeliveryPoint
    Ingredient "1" --> "1" Grill
    Ingredient "1" --> "1" Oven
    Ingredient "1" --> "1" CuttingBoard
    Ingredient "1" --> "1" PlateStack
    Ingredient "1" --> "1" Pantry
    Ingredient "1" --> "1" Ingredient
    Facing "1" --> "1" Order
    Facing "1" --> "1" Inventory
    Facing "1" --> "1" Bin
    Facing "1" --> "1" DeliveryPoint
    Facing "1" --> "1" Grill
    Facing "1" --> "1" Oven
    Facing "1" --> "1" CuttingBoard
    Facing "1" --> "1" PlateStack
    Facing "1" --> "1" Pantry
    Facing "1" --> "1" Ingredient
    MealType "1" --> "1" Order
    MealType "1" --> "1" Inventory
    MealType "1" --> "1" Bin
    MealType "1" --> "1" DeliveryPoint
    MealType "1" --> "1" Grill
    MealType "1" --> "1" Oven
    MealType "1" --> "1" CuttingBoard
    MealType "1" --> "1" PlateStack
    MealType "1" --> "1" Pantry
    MealType "1" --> "1" Ingredient
  
```

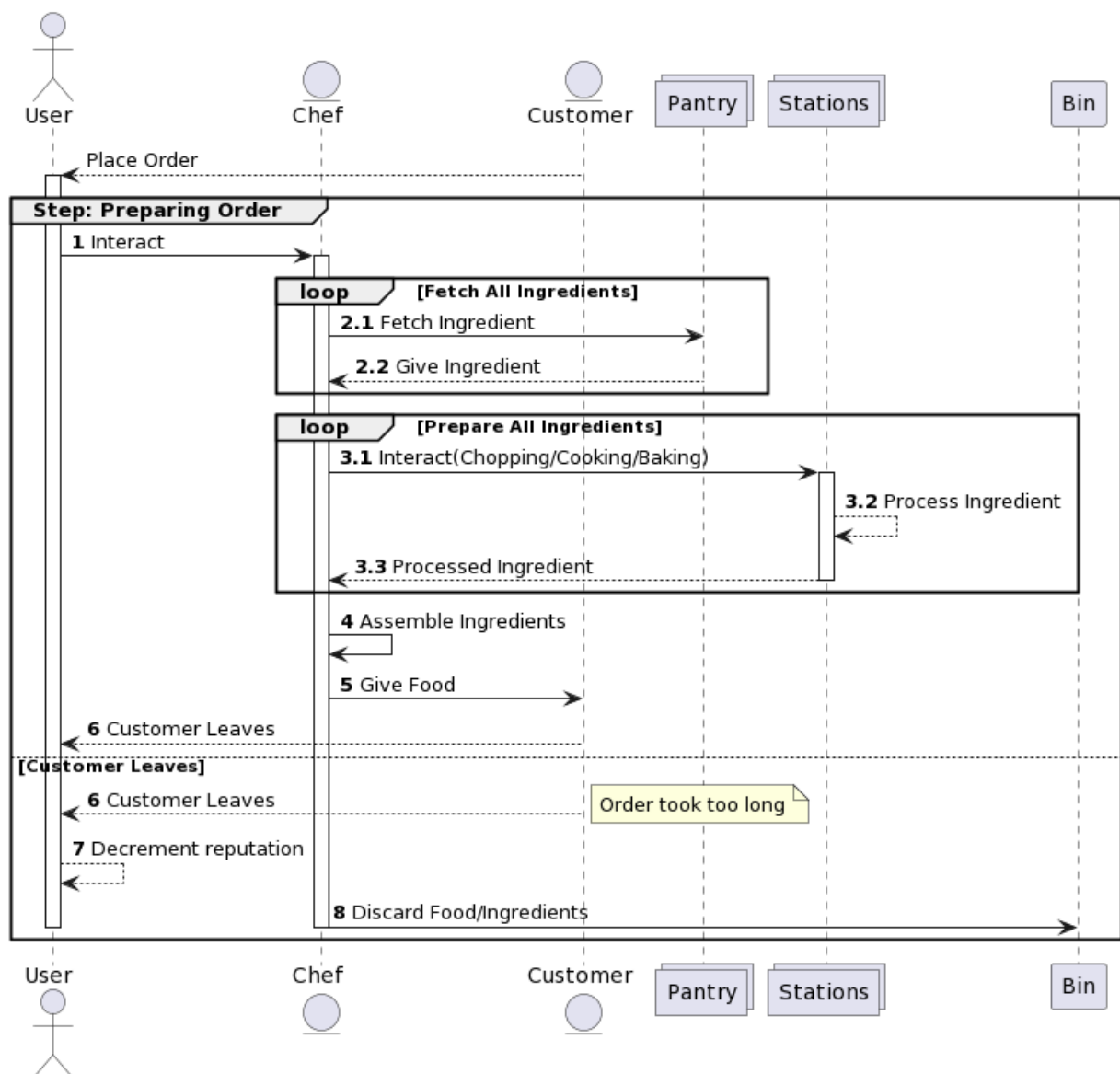
## Part B: Systemic Justification of Architecture

During the development phase of the architecture, the behavioural and structural diagrams were developed based on the Requirements section. This was carried out by:

- Focusing on satisfying all user requirements first.
- Discarding planned architectural developments that would hinder progress.
- Discarding architecture that is out of bound of user requirements.

The first iteration of the behavioural diagram did not receive any drastic changes to its structure, however, elements like “PlateStack” were introduced to facilitate problems like the congestion of items in the chef’s inventory or the method of serving the finished food items. The failure condition was removed as this version of the game does not have a failure condition, and the Step “Preparing Order” is replaced with the loop “Serve Orders”.

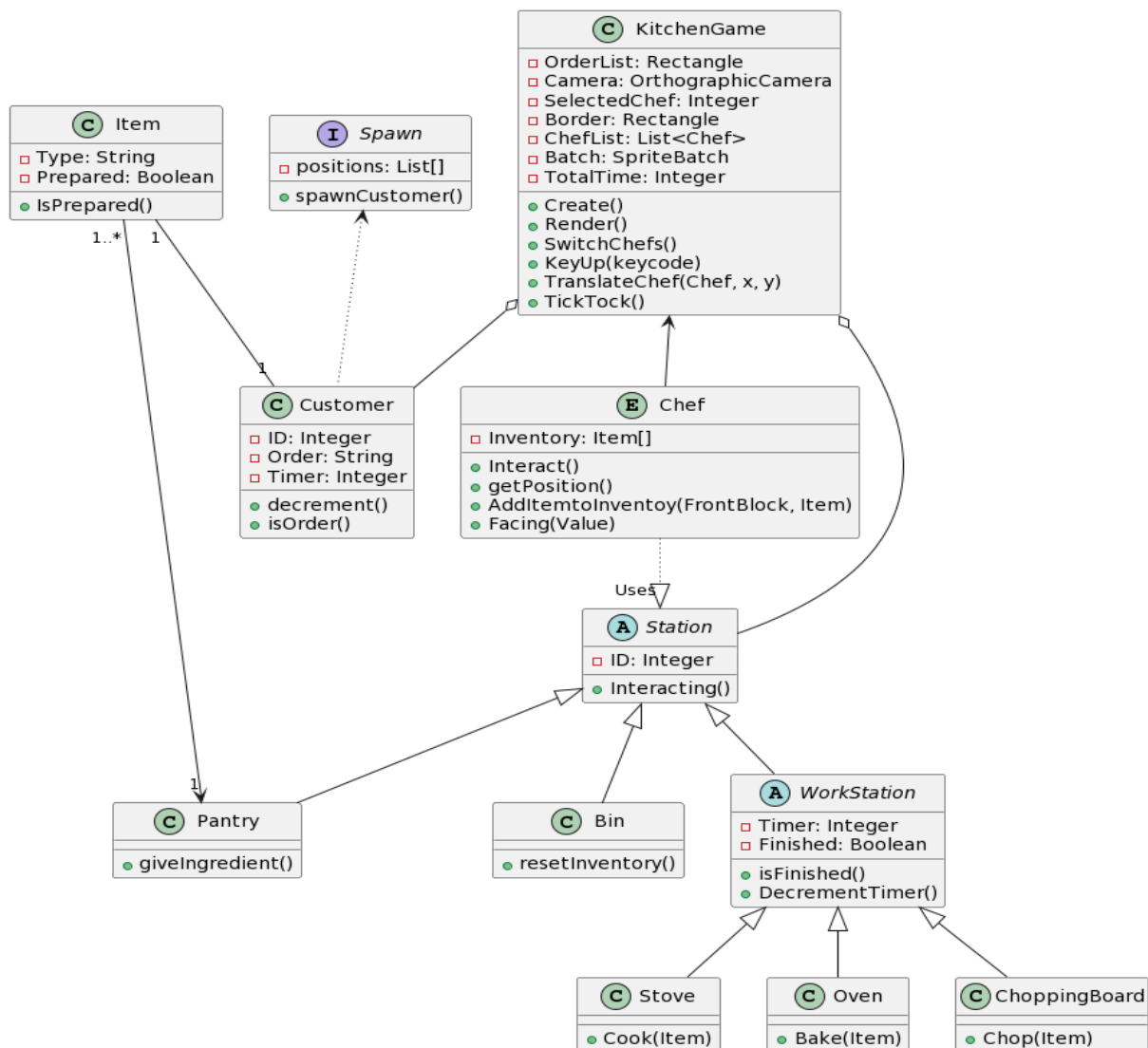
### Behavioural early iteration



In earlier iterations of the structural diagram, elements such as “Customer” and “Spawn” were introduced to develop the basis of what the ordering and reputation systems were meant to become. However, in future iterations the idea that a “customer” must be created and introduced along the “spawn” interface was deemed unnecessary, causing them to be discarded for the “order” entity that provided a simpler and more direct approach.

Elements like the abstract classes “Workstation” and “Station” would eventually be replaced by an interface “Station” to facilitate future teams’ implementation of the monetary system which allows players to earn and spend money on workstations.

### Structure early iteration



Java Class	Requirements	Relation
<b>Bin</b>	UR_ClearInventory	As the chef throws what they're holding into the bin, they empty their inventory stack.
<b>Chef</b>	UR_CookMove UR_ChefControl	Class to describe the current chefs position, movement, inventory and interactions.
<b>CuttingBoard</b>	FR_Recipe	Class used to change an ingredient, such as "Cheese" to "Chopped Cheese" from the top of the Chefs inventory stack.
<b>DeliveryPoint</b>	UR_Counter, FR_CustomerCounter	Class used as a space for the order given and the order from the Chefs inventory to each other.
<b>Grill</b>	FR_Recipe	Child class of Station to change an ingredient.
<b>Ingredient</b>	UR_Recipe, UR_Pantry, UR_CookPantry, FR_GettingIngredients	Parent class for each ingredient used in a recipe.
<b>Inventory</b>	UR_ClearInventory, UR_Pantry, UR_ClearInventory, FR_GettingIngredients	Parent Class to represent Ingredient/Plate and what the chef is carrying.
<b>Item</b>	FR_GettingIngredients	A parent class to Inventory.
<b>KitchenGame14</b>	UR_Gamedisplay NFR_FrameRate NFR_NoCrashing	The main class - used for creating and rendering objects.
<b>Order</b>	UR_CustomerOrder	Class which randomly generates an order from a customer, to be completed by the player/chefs.
<b>Oven</b>	UR_FlipPatties	A child class of Station, this can be used to cook an ingredient.
<b>Pantry</b>	UR_Pantry, FR_GettingIngredients	Child class of Station. Used as a place to collect an ingredient from.
<b>Plate</b>	UR_Plate	Used to show that an order is completed and being carried by the chef.
<b>PlateStack</b>	UR_Plate	A Class which converts the chef inventory - which contains the valid and necessary ingredients of a recipe - to a single item of a "Plate" which has the order on it.
<b>Station</b>	UR_Interact	A Parent class for Pantry, Oven, Grill.
<b>Tile</b>	UR_CookMove UR_Contollers	A class used to represent a section in the "game grid". Each element is a tile, whether it be a Countertop, Floor a tile which has a function such as Pantry.
<b>Timer</b>	UR_PlaygameTime, FR_CustomerTimeWaiting	A class used to represent time using the amount of frames which have passed. Also used throughout the program to add time delays to cooking processes, to add an elapsed time timer.