Lecture 12: Matlab APP

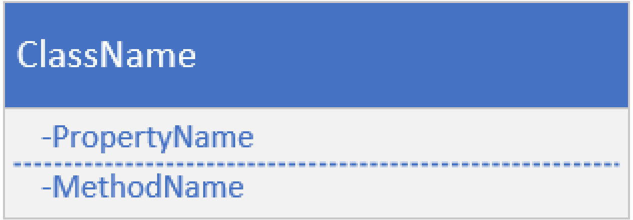
Announcement

- First customer consultation
  - Wed Apr 7th and Mon Apr 12th
  - Online via Tencent Meeting (5min for each team)
  - “Executive Summary” due on Mon Apr 5th (<=3 pg of ppt)

	Elevator	Vending	Train
Wed Apr 7th (1pm-1:50pm)	Team 1-11	Team 23-Team 33	Team 12-22
Wed Apr 7th (1:50pm-2:40pm)	Team 12-22	Team 34-Team 43	Team 1-11
Mon Apr 12th (1pm-1:50pm)	Team 23-Team 33	Team 1-11	Team 34-Team 44
Mon Apr 12th (1:50pm-2:40pm)	Team 34-Team 44	Team 12-22	Team 23-Team 33

Class definition in Matlab

```
classdef (ClassAttributes) ClassName < SuperClass1 & SuperClass2
    properties (PropertyAttributes)
        ...
    end
    methods (MethodAttributes)
        ...
    end
    events (EventAttributes)
        EventName
    end
end
```



Class Attributes

- Abstract
  - If specified as true, this class is an abstract class (cannot be instantiated).
  - classdef (Abstract = true) ClassName

- Sealed
  - If true, this class cannot be subclassed.

Value Class vs. Handle Class

- Value Class
- Each assignment creates a new copy of the object

```
classdef NumValue
  properties
    Number = 1
  end
end
```

- a = NumValue;
- b=a;
- a.Number = 7;
- b.Number
  - ans=1

- Handle Class
- Upon construction a reference to the object is created

```
classdef NumHandle < handle
  properties
    Number = 1
  end
end
```

- a = NumHandle;
- b=a;
- a.Number = 7;
- b.Number
  - ans=7

Value Class vs. Handle Class (cont.)

- When object passed into a function
  - Value object: a new copy of the object is created inside function workspace
  - Handle object: a copy of the handle (reference) is created instead of the object

- Deleting a handle object
  - Delete(NumHandle)

Object equality

Value object

- Can only evaluate whether value of the objects are the same

- a = NumValue;
- b = NumValue;
- isequal(a,b)
- ans=1

Handle object

- Can check whether they are the same object as well as their value equality

- a = NumHandle;
- b = a;
- a == b (same object?)
  - ans=1;
- isequal(a,b) (same value?)
  - ans=1;

a = NumHandle;

b = NumHandle;

a == b

ans=0;

isequal(a,b)

ans=1;

Class Members Access

- public — Unrestricted access
- protected — Access from methods in class or subclasses
- private — Access by class methods only (not from subclasses)
- List classes (and their subclasses) have access to this member
  - (Access = {?ClassName1,?ClassName2,...})

Property Attributes

- Read and write access
  - GetAccess
  - SetAccess
    - properties(GetAccess = 'public', SetAccess = 'private')
    - % public read access, but private write access.
    - end
    - SetAccess = immutable: set during construction, cannot be changed afterwards
- Constant

```
properties(Constant = true)
    DAYS_PER_YEAR =  365;
end
```

- Dependent
  - depend on other values
  - calculated only when needed.
  - i.e. area of a square depends on the width property

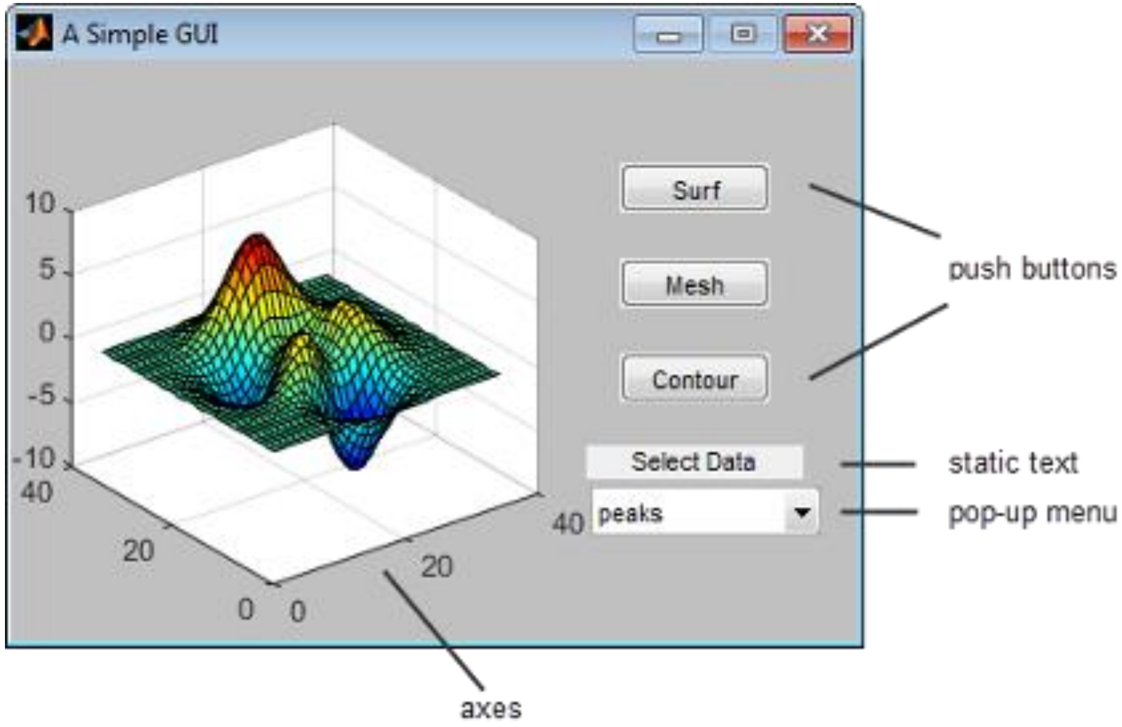
Class Constructor Method

- There is a default class constructor without input arguments
- We can define class constructor that overrides the default one
- Method with the same name as the class name

```
classdef ConstructorDesign < BaseClass1
methods
    function obj = ConstructorDesign(a,b,c)
end
end
```

Graphic User Interface

- Consists of handle objects
  - figure
  - axes
  - uitable
  - uicontrol
- Each of them have unique properties and methods
- You can view and change these by calling
  - inspect(h)



figure

- h=figure(prop1,’prop1value’,…);

- Make one of the figures active
  - figure(h1)

- Get the handle of the current figure
  - h=gcf;

figure properties

- Units
  - ‘pixels’
  - ‘normalized’: from (0,0) lower left to (1,1) upper right
- Position
  - [left bottom width height]
- Name
- Parent
  - root is the top level
- Children
  - n*1 graphic handle
  - Ordered according to level first, and then stacking order

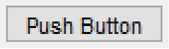
Figure callback events

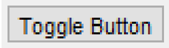
- ButtonDownFcn
  - When clicking the mouse button while the pointer is located over or near the object.
- KeypressFcn
  - When a key is pressed
- CreatFcn
  - When creating an object
- DeleteFcn
  - When deleting an object.

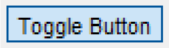
Callback functions

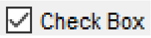
- h=figure(‘ButtonDownFcn’,@testButtonDown)
- function testButtonDown(src,event)
  - src: the UI component that triggered the callback
  - event: event data. i.e. key pressed
- Provide additional input arguments to the callback function
  - h=figure(‘ButtonDownFcn’,{@testButtonDown,arg1,arg2…})
  - function testButtonDown(src,event, arg1,arg2…)

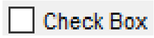
uicontrol

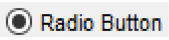
- ctrlhdl=uicontrol(fighdl,’style’,ctrlstyle,’prop1’,prop1value…)
- pushbutton 

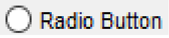
- togglebutton 

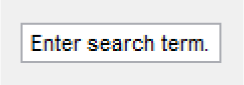


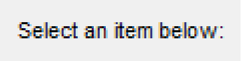
- checkbox 

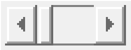
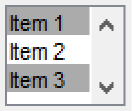


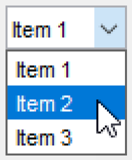
- radiobutton 



- edit 

- text 

- slider 
- listbox 

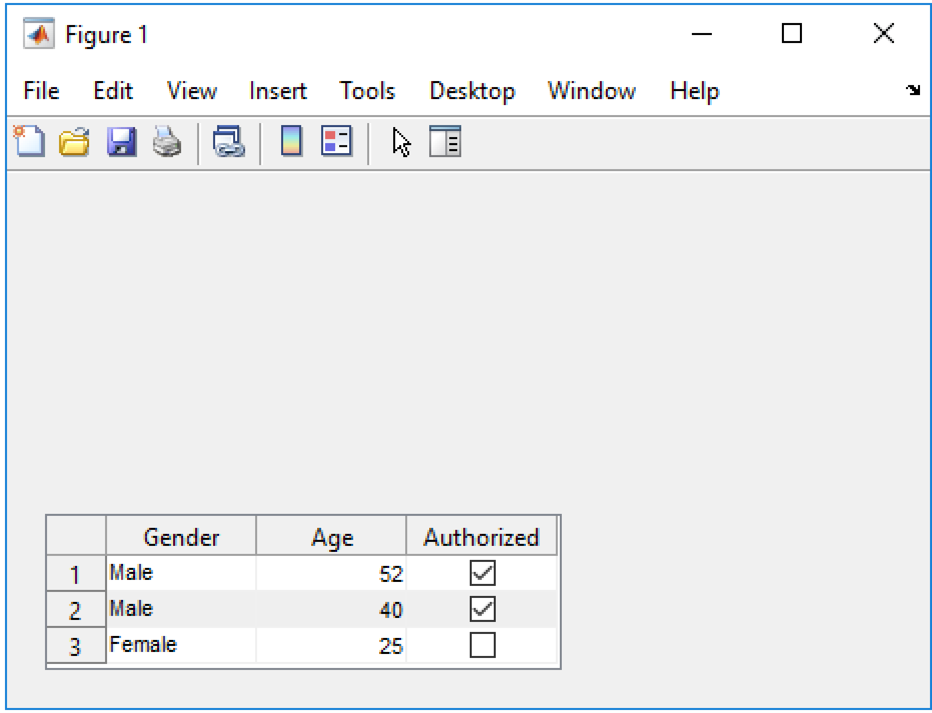
- popupmenu 

Common uicontrol properties

- Value
  - Checked/unchecked, slider position, listbox active index, etc
- String
  - Displayed string
  - Cell array of strings for listbox and popupmenu
  - i.e. {‘item1’;’item2’;’item3’}

uitable

- h=uitable(parent,’prop1’,prop1value…)
- data
  - A cell matrix
  - {'Male',52,true;'Male',40,true;'Female',25,false};
- ColumnName
  - 1*n Cell array
  - {'Gender','Age','Authorized’};



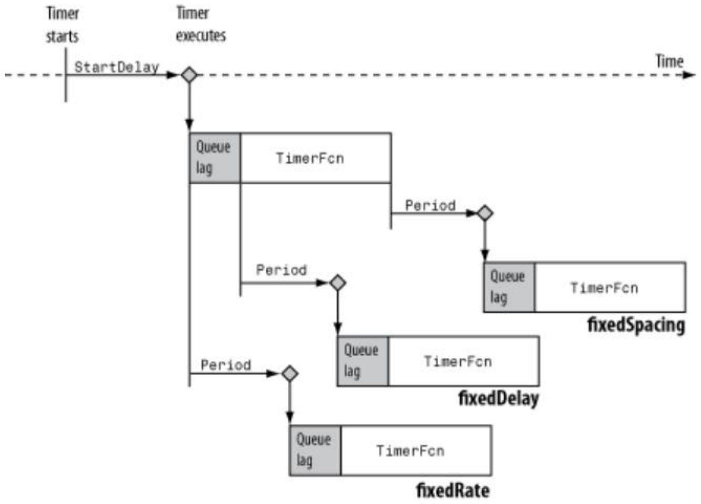
	Gender	Age	Authorized
1	Male	52	☑
2	Male	40	☑
3	Female	25	□

uitable callback events

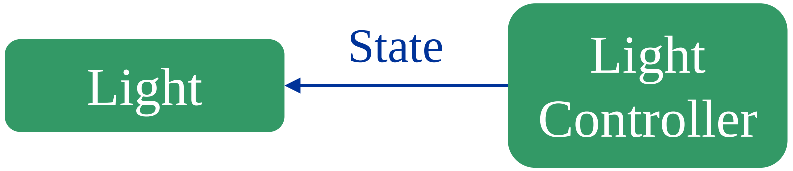
- CellSelectionCallback
  - CellSelectionChangeData as input argument
    - Indices: row and column indices of the cell the user edited
- CellEditCallback
  - CellEditData as input argument
    - Indices:
    - PreviousData
    - NewData

Timer Class

- t = timer;
- Properties
  - ‘ExecutionMode’
  - ‘Period’: Time between timer functions
  - ‘TimerFcn’: Function handle
- t.TimerFcn=@callback;
- Function callback(hObj,src,event)



Demo: Traffic Light



Example: Information system for restaurants

- The owner of restaurant A would like to improve service efficiency

Customer

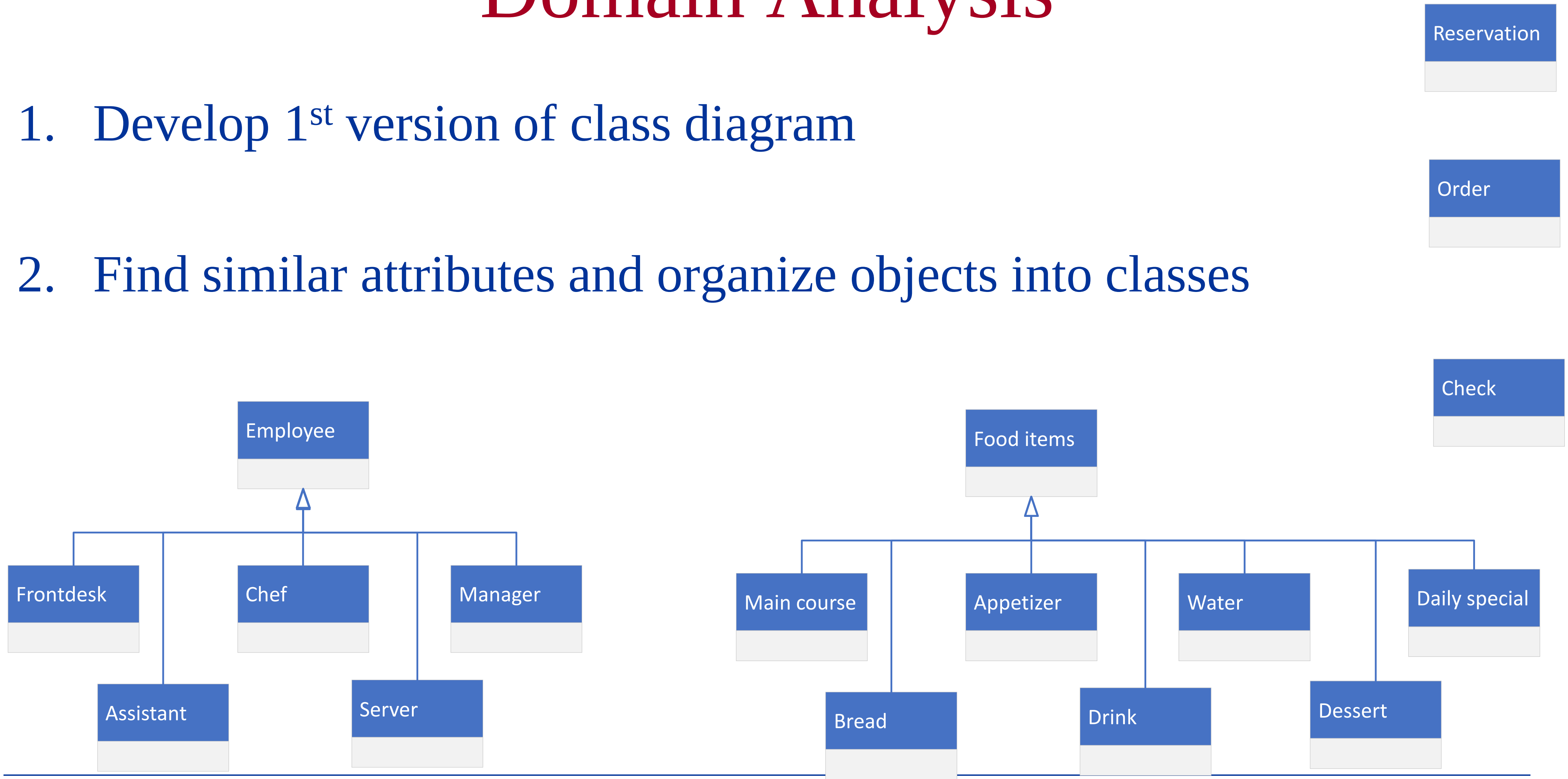
Server

Chef

Domain Analysis

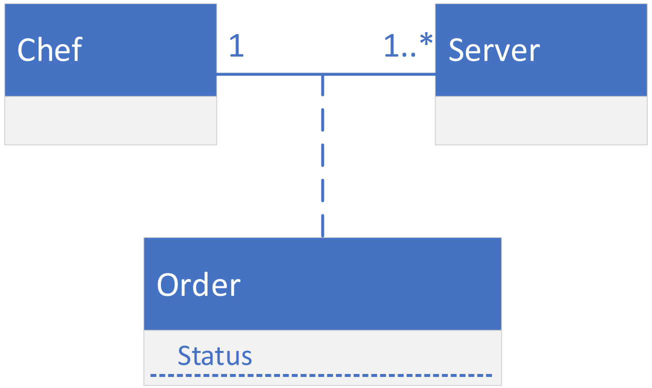
1. Develop 1st version of class diagram

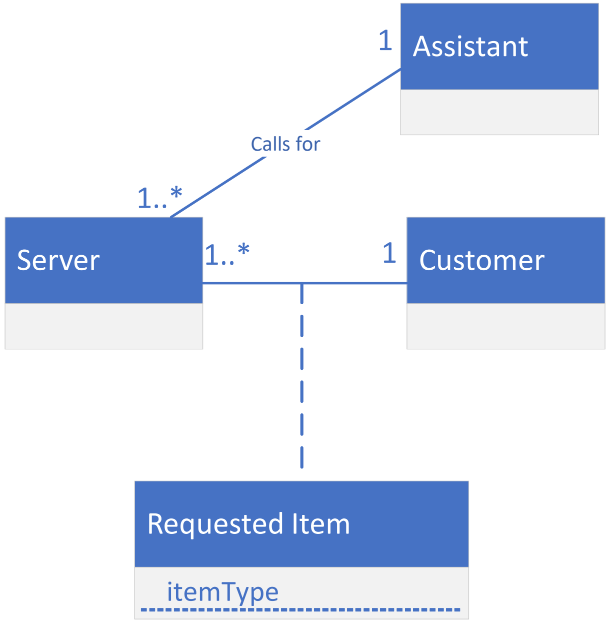
2. Find similar attributes and organize objects into classes

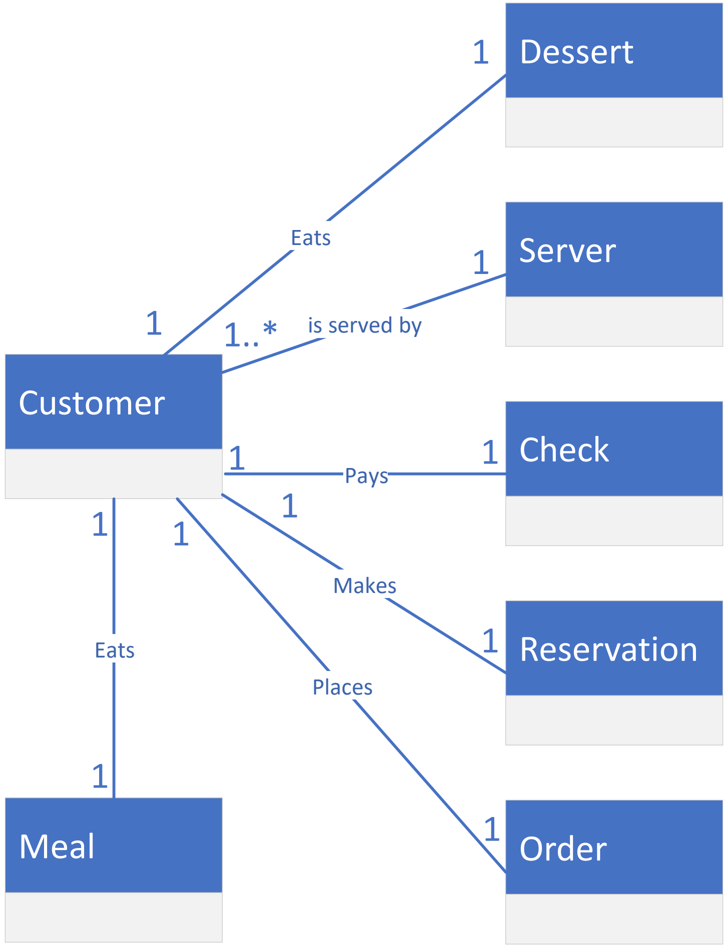


Domain Analysis (cont.)

3. Further understand the domain
   - Find associations

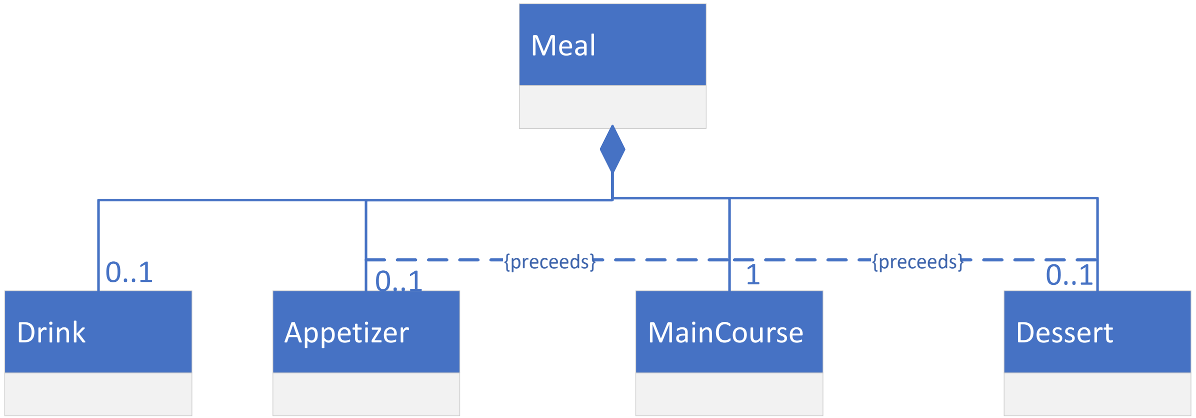


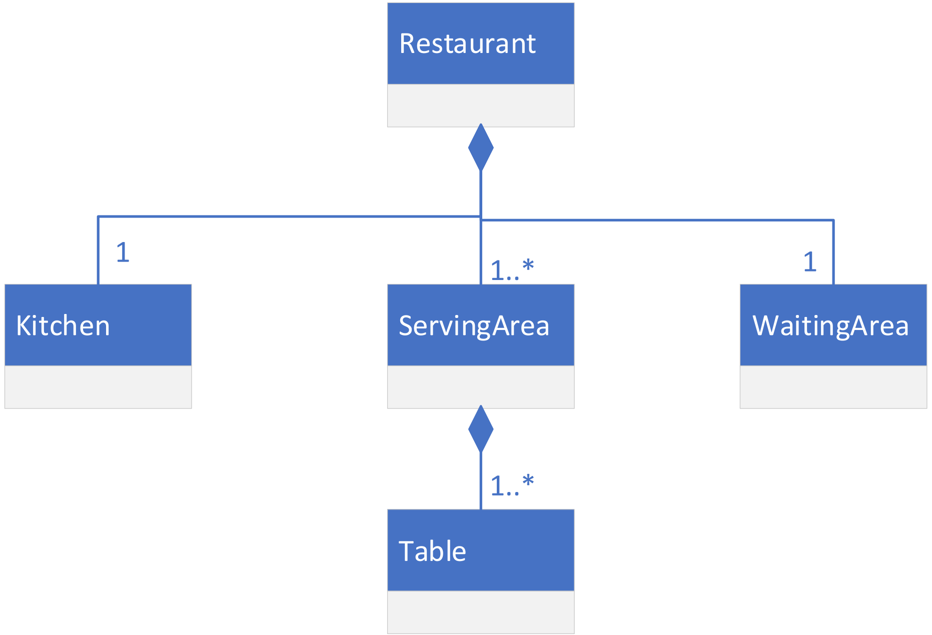




Domain Analysis (cont.)

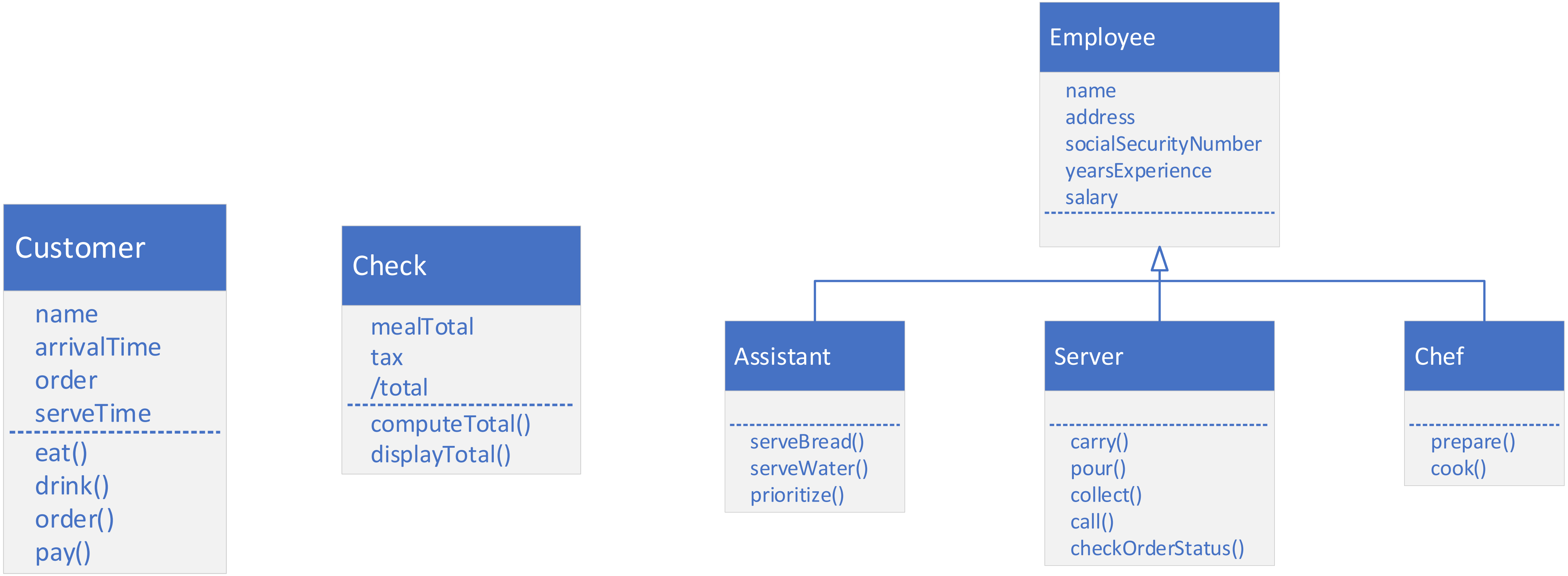
4. Find aggregations and compositions



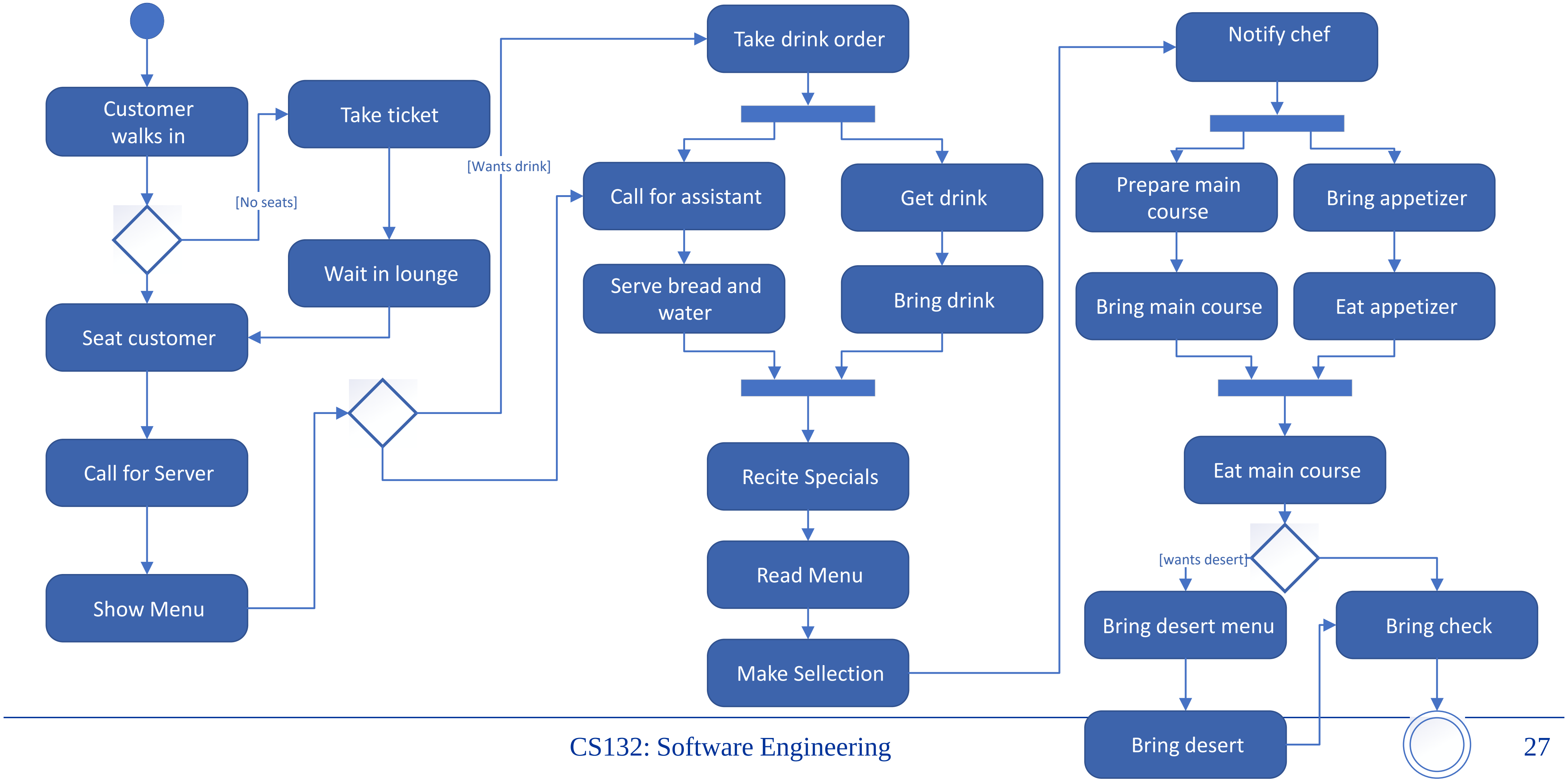


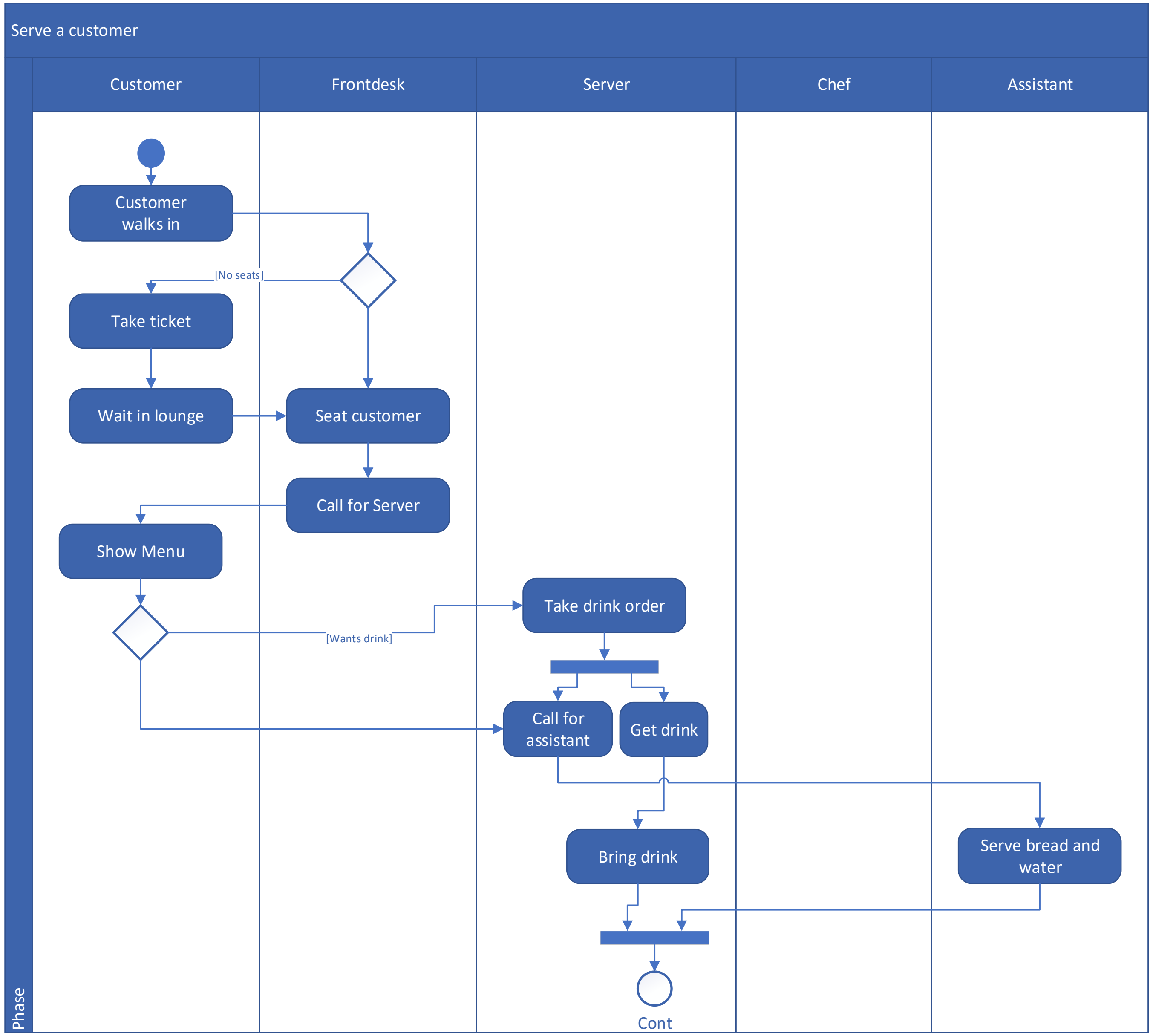
Domain Analysis (cont.)

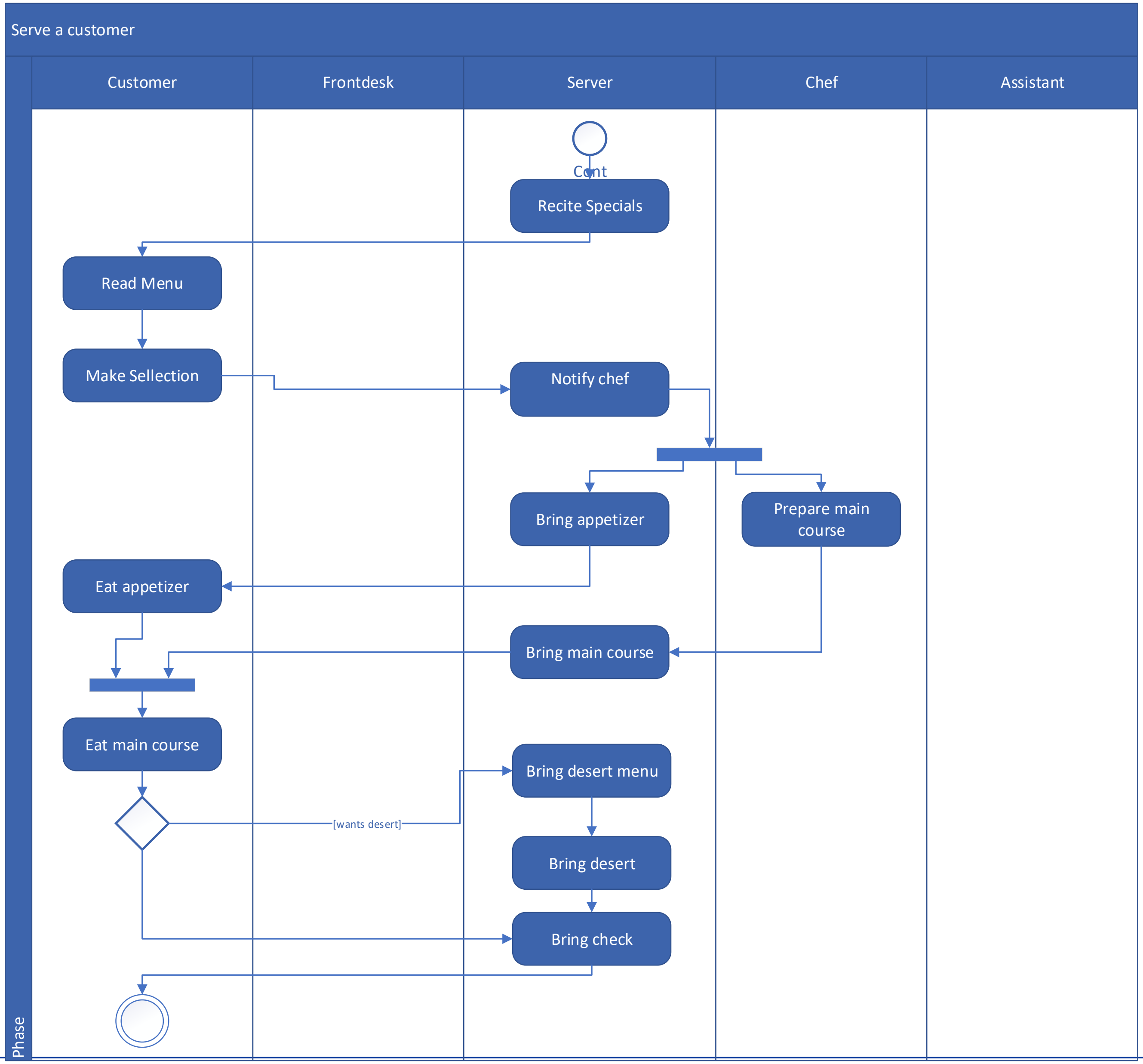
5. Enrich information in classes



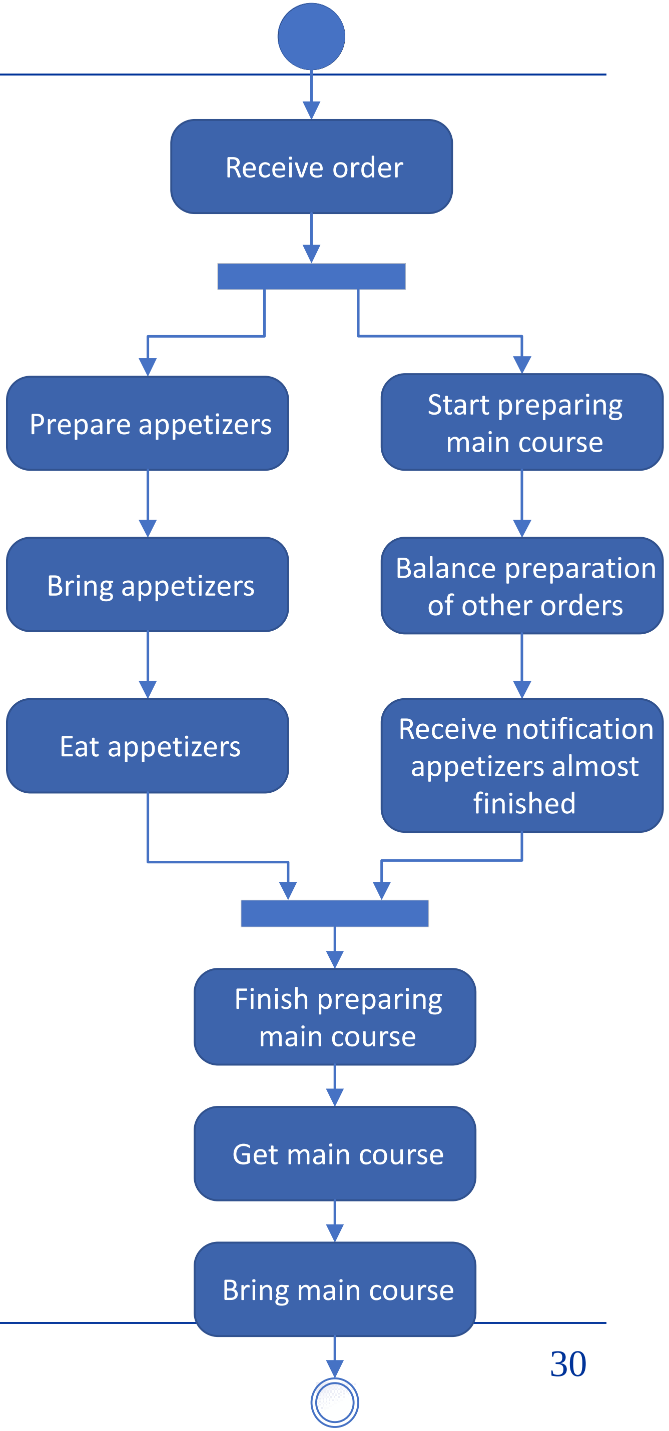
Discover Domain Procedures

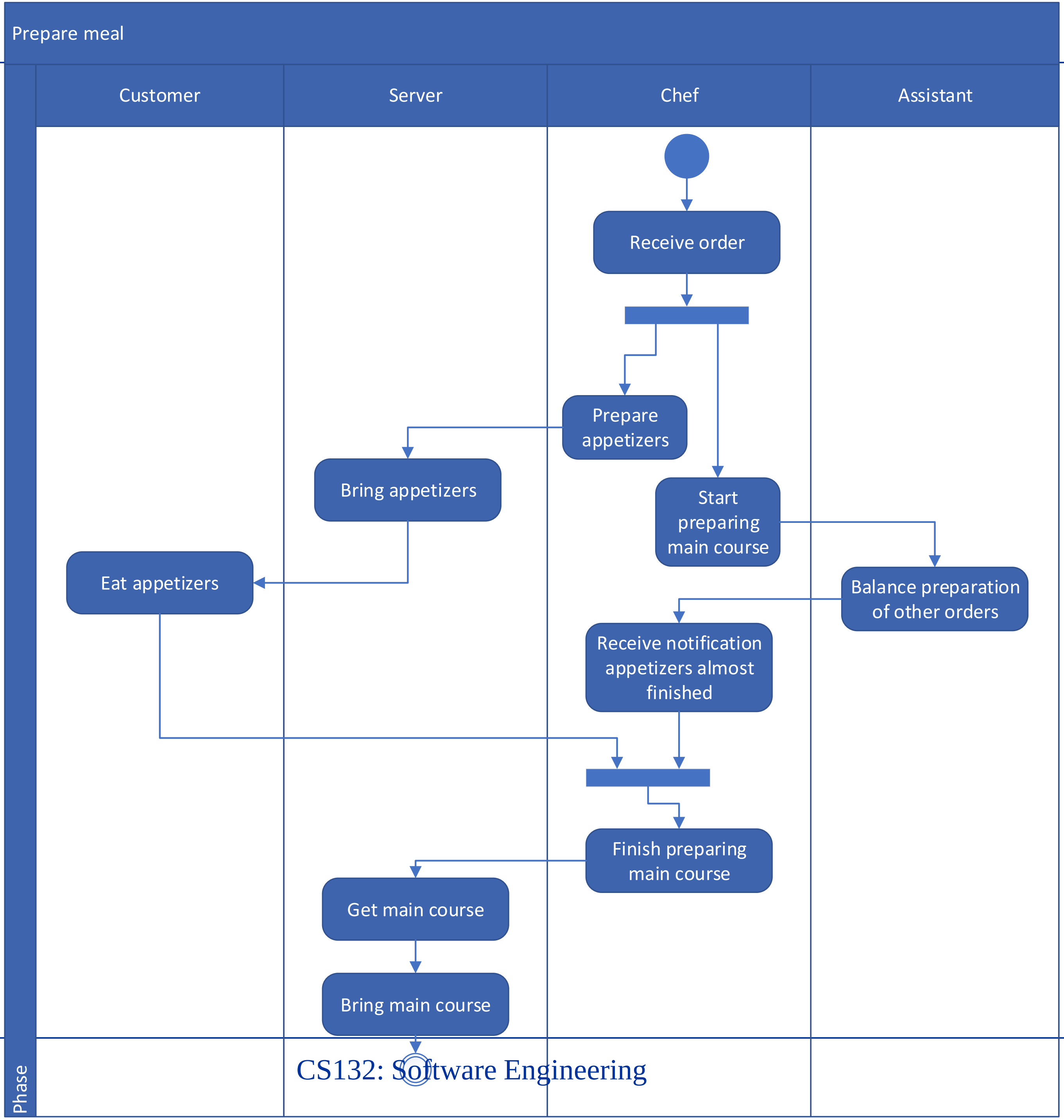



Serve a customer
Customer
Frontdesk
Server
Chef
Assistant
Phase
Customer walks in
[No seats]
Take ticket
Wait in lounge
Seat customer
Call for Server
Show Menu
[Wants drink]
Take drink order
Call for assistant
Get drink
Bring drink
Serve bread and water
Cont


Serve a customer
Customer
Frontdesk
Server
Chef
Assistant
Phase
Cont
Recite Specials
Read Menu
Make Sellection
Notify chef
Bring appetizer
Prepare main course
Eat appetizer
Bring main course
Eat main course
Bring desert menu
[wants desert]
Bring desert
Bring check

- Prepare food


Receive order
Prepare appetizers
Start preparing main course
Bring appetizers
Balance preparation of other orders
Eat appetizers
Receive notification appetizers almost finished
Finish preparing main course
Get main course
Bring main course

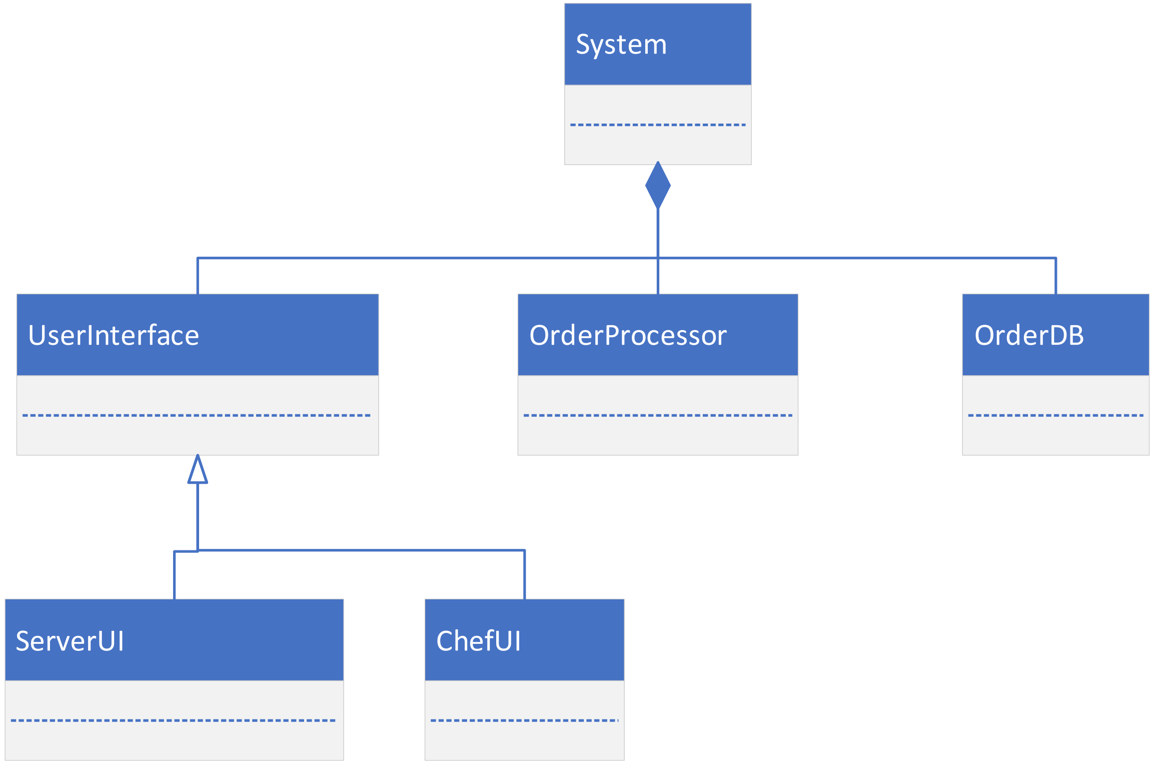

Prepare meal
Customer
Server
Chef
Assistant
Receive order
Prepare appetizers
Bring appetizers
Start preparing main course
Eat appetizers
Balance preparation of other orders
Receive notification appetizers almost finished
Finish preparing main course
Get main course
Bring main course
Phase

Discover system requirements

- Joint Application Development (JAD) session
  - Restaurant owner
    - Understands the overall objectives of the system
  - Server
    - Actual user of the system
  - System analyst
    - From solution’s perspective: propose potential system architecture
  - Modeler
    - From problem’s perspective: abstract potential use cases
  - Coordinator
    - Keep the conversations on track

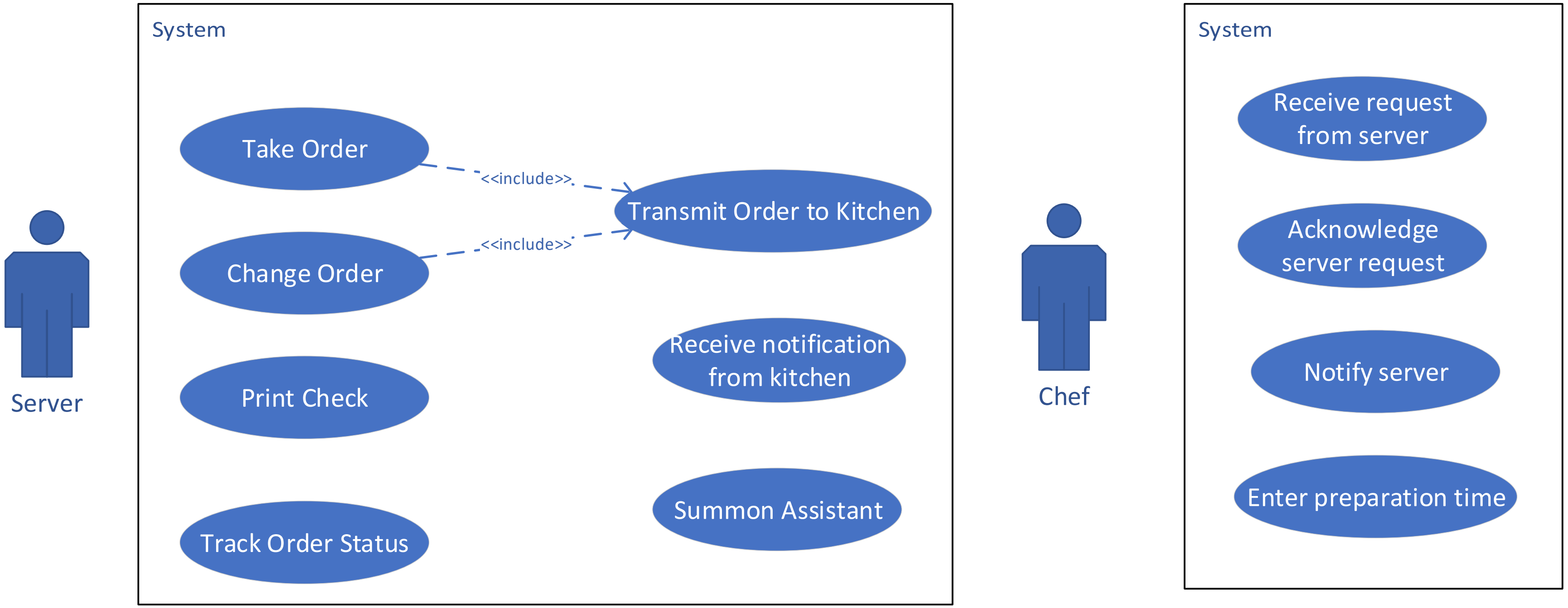
Discover system requirements (cont.)

- Requirements for intelligent restaurant system
  - Primary: Save the server’s travel time between kitchen and serving area
  - Secondary: Improve serving quality and efficiency
- Proposed solution
  - An order database that keeps track of order information
  - An order processor that handles order generation/modification
  - User interface for both the chef and the server



Discover system requirements (cont.)

- System requirements as use cases

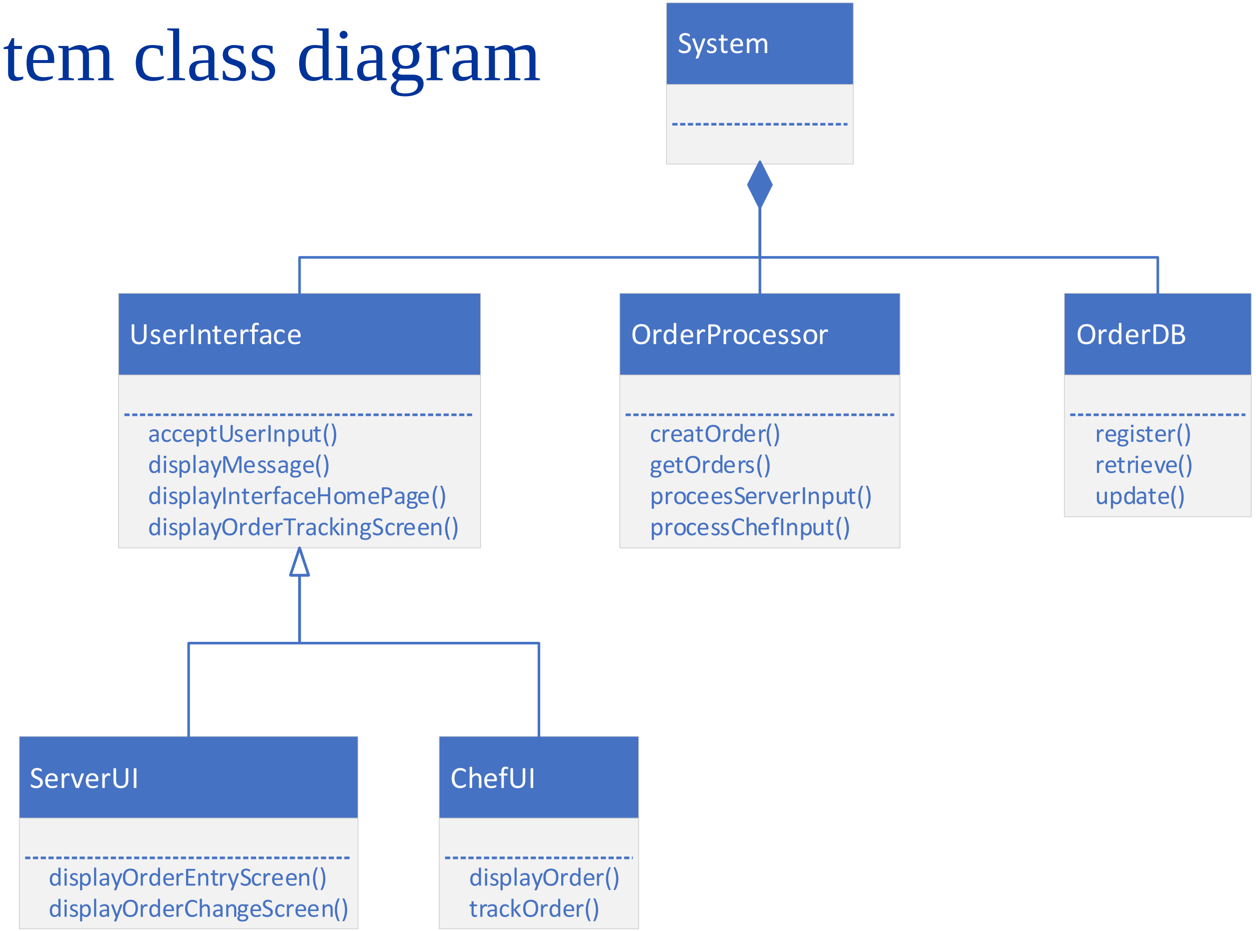


Discover system requirements (cont.)

- Expanding use cases in another JAD meeting
- Use case “Take an order”
  - Description: Server inputs order data in his/her terminal and transmit the order to the kitchen.
  - Precondition: Customer has read the menu and made selections
  - Postcondition: Order has been input into the system
  - Standard procedure
    1. Server activate the order entry screen on his/her terminal
    2. Server input the order information on the screen
    3. System send the order to the chef UI

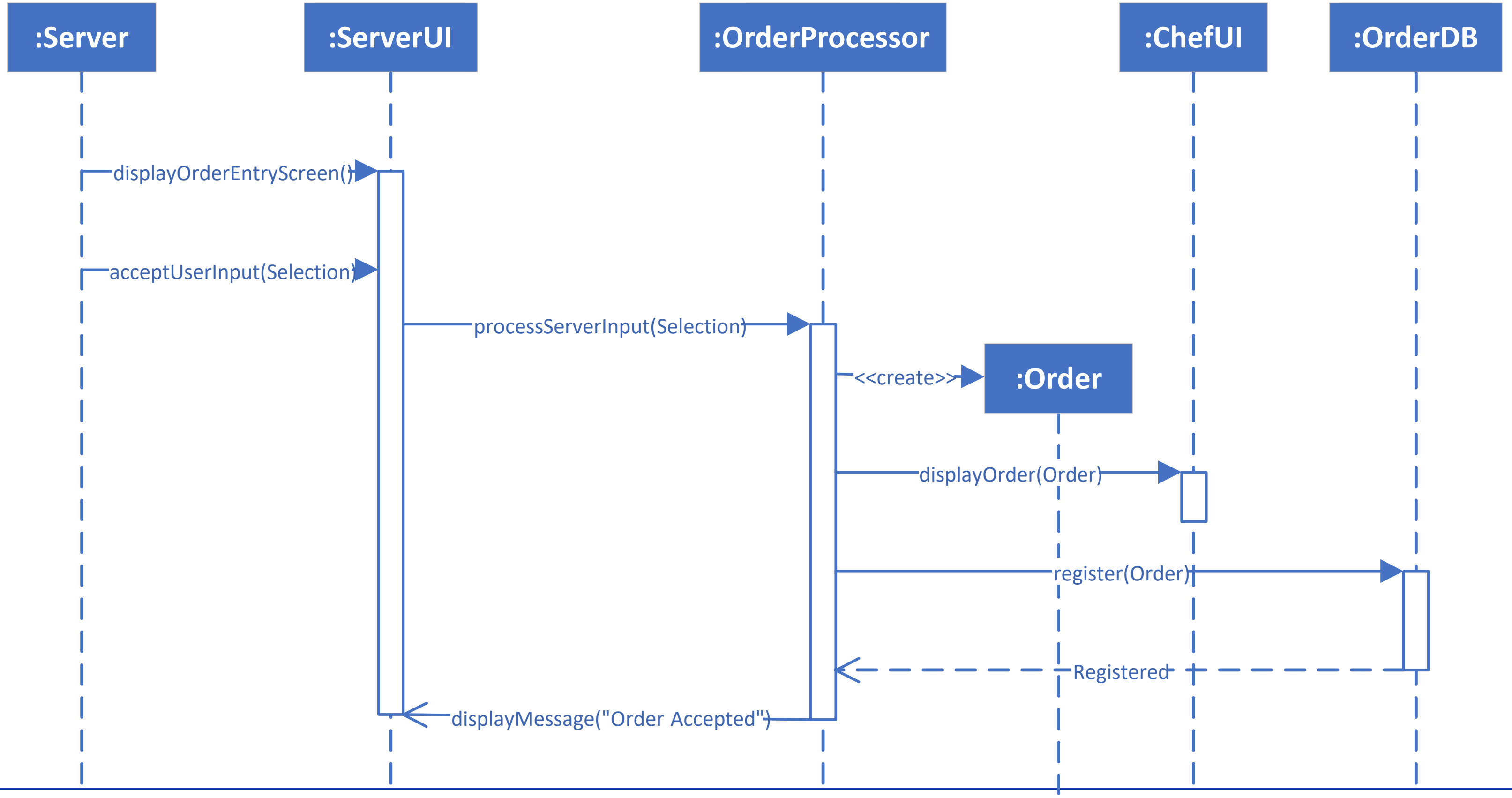
Discover system requirements (cont.)

- Enriched system class diagram



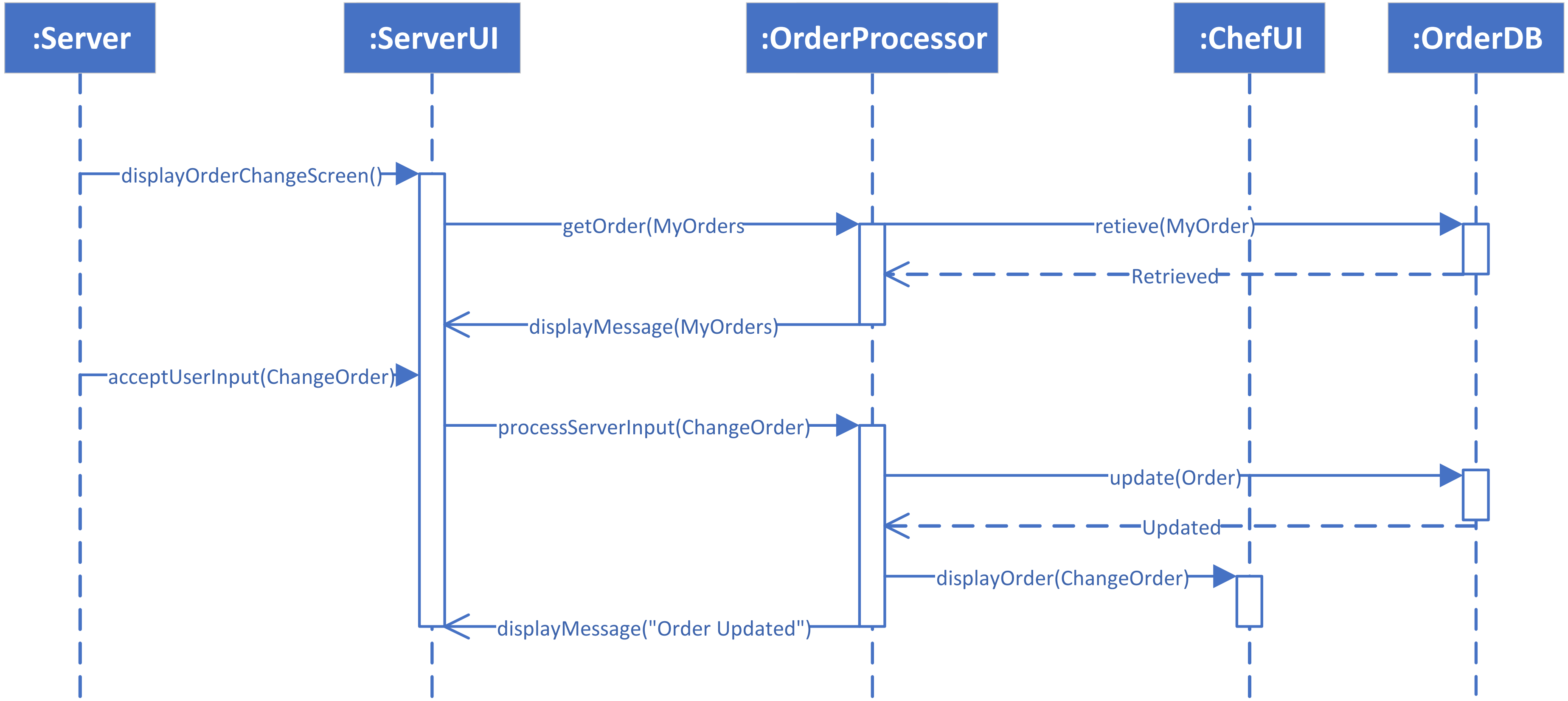
Identify interactions

- Use case “Take an order”


:Server
:ServerUI
:OrderProcessor
:ChefUI
:OrderDB
displayOrderEntryScreen()
acceptUserInput(Selection)
processServerInput(Selection)
<<create>>
:Order
displayOrder(Order)
register(Order)
Registered
displayMessage("Order Accepted")

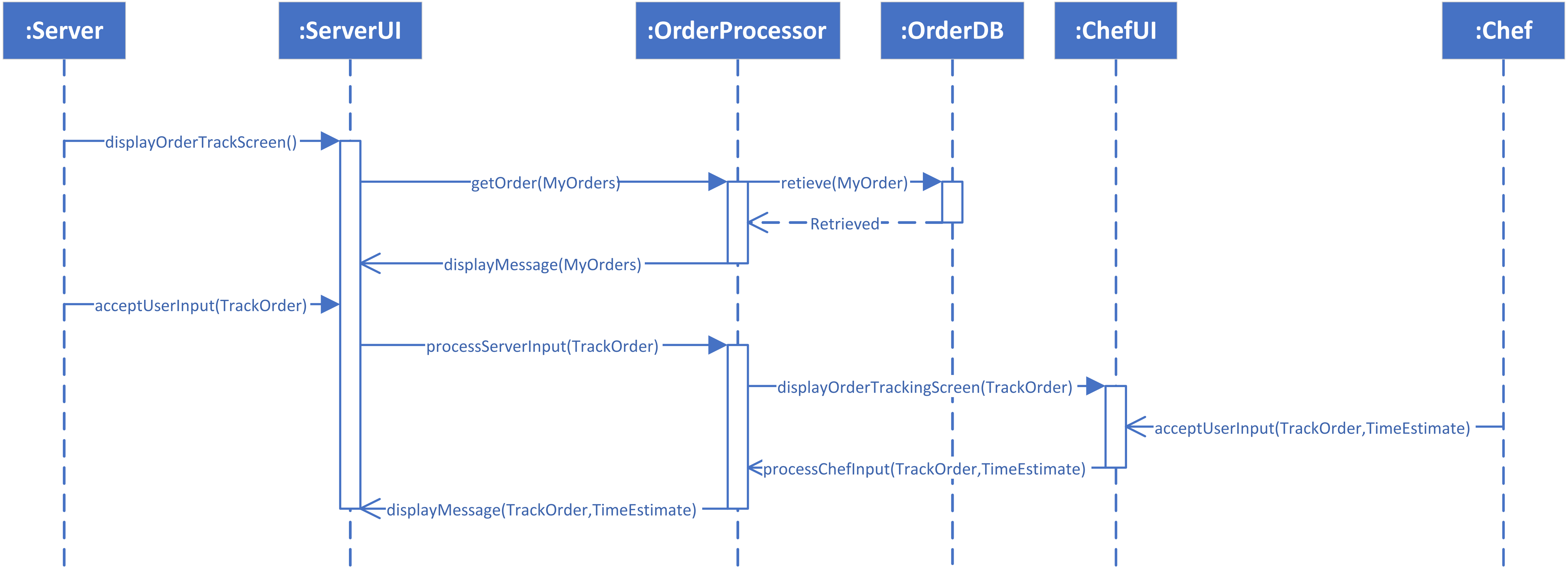
Identify interactions (cont.)

- Use case “Change an order”


:Server
:ServerUI
:OrderProcessor
:ChefUI
:OrderDB
displayOrderChangeScreen()
getOrder(MyOrders
retieve(MyOrder)
Retrieved
displayMessage(MyOrders)
acceptUserInput(ChangeOrder)
processServerInput(ChangeOrder)
update(Order)
Updated
displayOrder(ChangeOrder)
displayMessage("Order Updated")

Identify interactions (cont.)

- Use case “Track an order”


:Server
:ServerUI
:OrderProcessor
:OrderDB
:ChefUI
:Chef
displayOrderTrackScreen()
getOrder(MyOrders)
retieve(MyOrder)
Retrieved
displayMessage(MyOrders)
acceptUserInput(TrackOrder)
processServerInput(TrackOrder)
displayOrderTrackingScreen(TrackOrder)
acceptUserInput(TrackOrder,TimeEstimate)
processChefInput(TrackOrder,TimeEstimate)
displayMessage(TrackOrder,TimeEstimate)