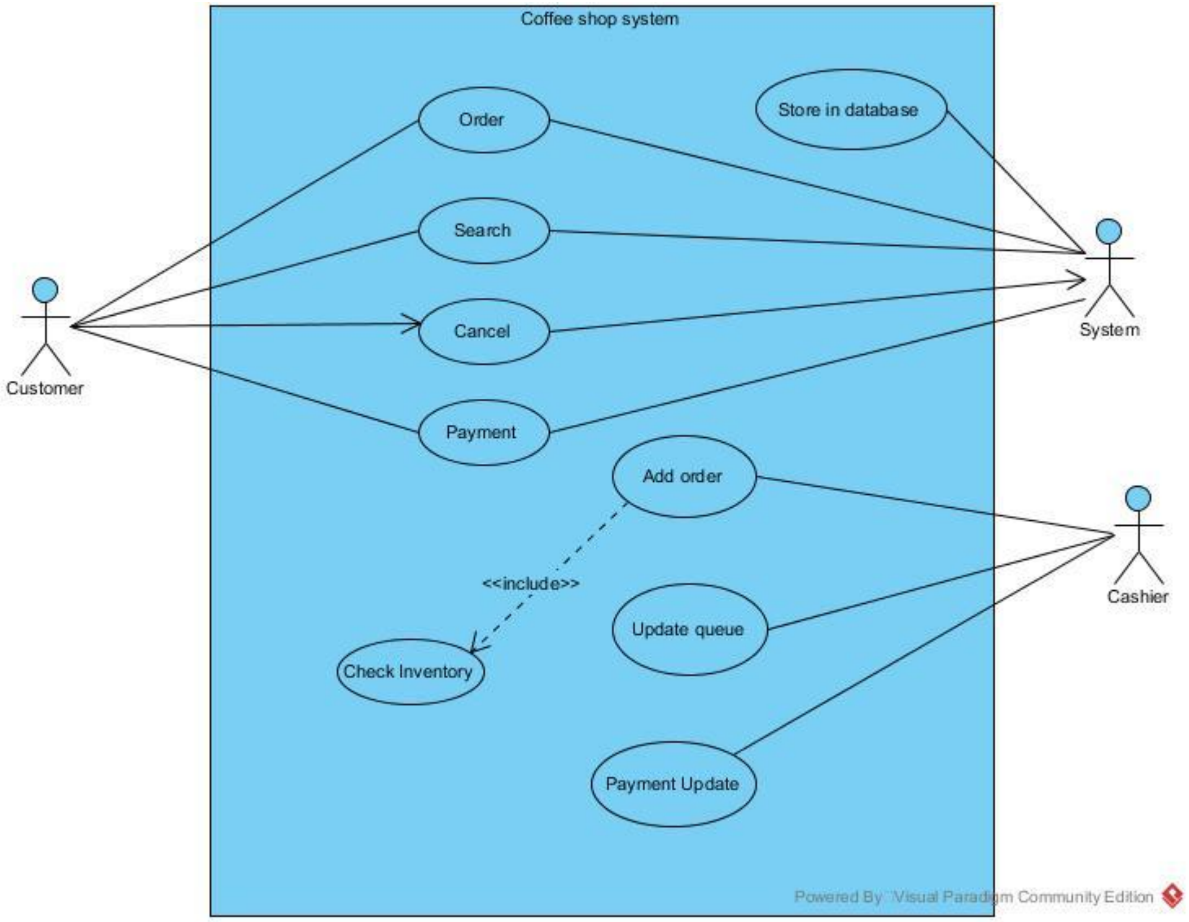
Coffee shop system

Use case Diagram



URS and SRS: Coffee shop system

URS (01): Customer will search menu about coffee for ordering.

URS (02): Customer orders coffee to Cashier.

SRS (01): Cashier will add order into the system.

SRS (02): The system will check inventory for make product.

SRS (03): The system will update status.

SRS (04): The system will calculate price of product

URS (03): Customer pay money with cashier for buy product.

SRS (05): Cashier receive money from customer.

SRS (06): Cashier make queue ticket and give to customer for waiting.

URS (04): Customer get queue ticket from cashier and waiting for product.

SRS (07): The system called customer queue.

URS (05): Customer will receive product

SRS (08): The system will update status and payment.

Use case Description: Coffee Shop System

Use Case ID	1			
Use Case Name	Order			
Created By	JIrapat		Last Update By	JIrapat
Date Created	14 / 04 / 2559		Last Revision Date	14 / 04 / 2559
Actors	Customer, System, Cashier			
Description	For ordering menu with cashier.			
Trigger	Cashier must click order button.			
Preconditions	Cashier must login before order.			
Use Case Input Specification				
Input	type	Constraint		Example
Name	String	This value can be input only alphabet. It can’t be number.		Latte
Post conditions	Update order into list			
Normal Flows	User		System	
	1. Customer choose menu. 2. Customer order coffee.		3. Cashier will add order in list of product. 4. System will update	
Alternative Flow	1. Customer not enough money for pay. 2. Update error			
Exception Flow	1. User want to change abrupt 2. Out of power			
Assumption	Customer must have order already.			

Use Case ID	2			
Use Case Name	Search			
Created By	Jirapat		Last Update By	Jirapat
Date Created	14 / 04 / 2559		Last Revision Date	14 / 04 / 2559
Actors	Cashier			
Description	Choice of customer for ordering.			
Trigger	Cashier must click search button for search menu.			
Preconditions	Cashier have to login before do this			
Use Case Input Specification				
Input	type	Constraint		Example
Name Price Type	String double String	This value can be input only alphabet. It can’t be number. Amount of money, It can’t be negative number. Type of product		Latte 50 Hot/Ice
Post conditions				
Normal Flows	User		System	
	1. Customer searching menu		2. System will show lists of menu	
Alternative Flow	1. Customer want to cancel ordered.			
Exception Flow	1. Out of power			
Assumption	Customer must have menu in your own for search.			

Use Case ID	3		
Use Case Name	Cancel		
Created By	Jirapat	Created By	Jirapat
Date Created	14 / 04 / 2559	Date Created	14 / 04 / 2559
Actors	Cashier		
Description	For customer want to change or cancel ordered.		
Trigger	Cashier must click cancel button for cancel list.		
Preconditions	Cashier have to login before do this.		
Use Case Input Specification			
Input	type	Constraint	Example
Name	String	This value can be input only alphabet. It can't be number.	Latte
Post conditions	Update list of product		
Normal Flows	User	System	
	1. Customer will cancel product	2. Cashier cancel product 3. The system remove product from list	
Alternative Flow	-		
Exception Flow	1. Out of power 2. Server error		
Assumption	Customer must tell cashier about cancel when customer want.		

Use Case ID	4		
Use Case Name	Payment		
Created By	Jirapat	Created By	Jirapat
Date Created	14 / 04 / 2559	Date Created	14 / 04 / 2559
Actors	Cashier		
Description	For customer buy product.		
Trigger	Customer must pay money to cashier.		
Preconditions	Customer must order before payment.		
Use Case Input Specification			
Input	type	Constraint	Example
Price	int	Amount of money, can't be negative number.	50
Post conditions	Update payment		
Normal Flows	User	System	
	2. After customer ordered they must pay money before get product 3. Customer get change (maybe have)	1. Cashier input list menu for check bill 3. Cashier receive money 4. The system will update payment 5. The system will calculate money maybe have change.	
Alternative Flow	1. System calculate incorrect 2. Customer pay amount of money incorrect		
Exception Flow	1. Out of power 2. Server error 3. Cashier input incorrect		
Assumption	Customer must have money for buy.		

Use Case ID	5		
Use Case Name	Store in database		
Created By	Jirapat	Last Update By	Jirapt
Date Created	14 / 04 / 2559	Last Revision Date	14 / 04 / 2559
Actors	Cashier		
Description	For keep data into server (update store data).		
Trigger	Cashier must click update button before the system will store.		
Preconditions	Cashier have to update payment before the system will store.		
Use Case Input Specification			
Input	type	Constraint	Example
Name	String	This value can be input only alphabet. It can’t be number.	Latte
Price	Money	Amount of money, can’t be negative number.	50
Date	date	dd/mm/yyyy	14 / 04 / 2559
Time	time	Time for pay, It can’t be negative number.	16:00 pm
Post conditions	Update store in database		
Normal Flows	User	System	
		1. System will update store	
Alternative Flow	-		
Exception Flow	1. Out of power 2. Server error		
Assumption	System must have data for store.		

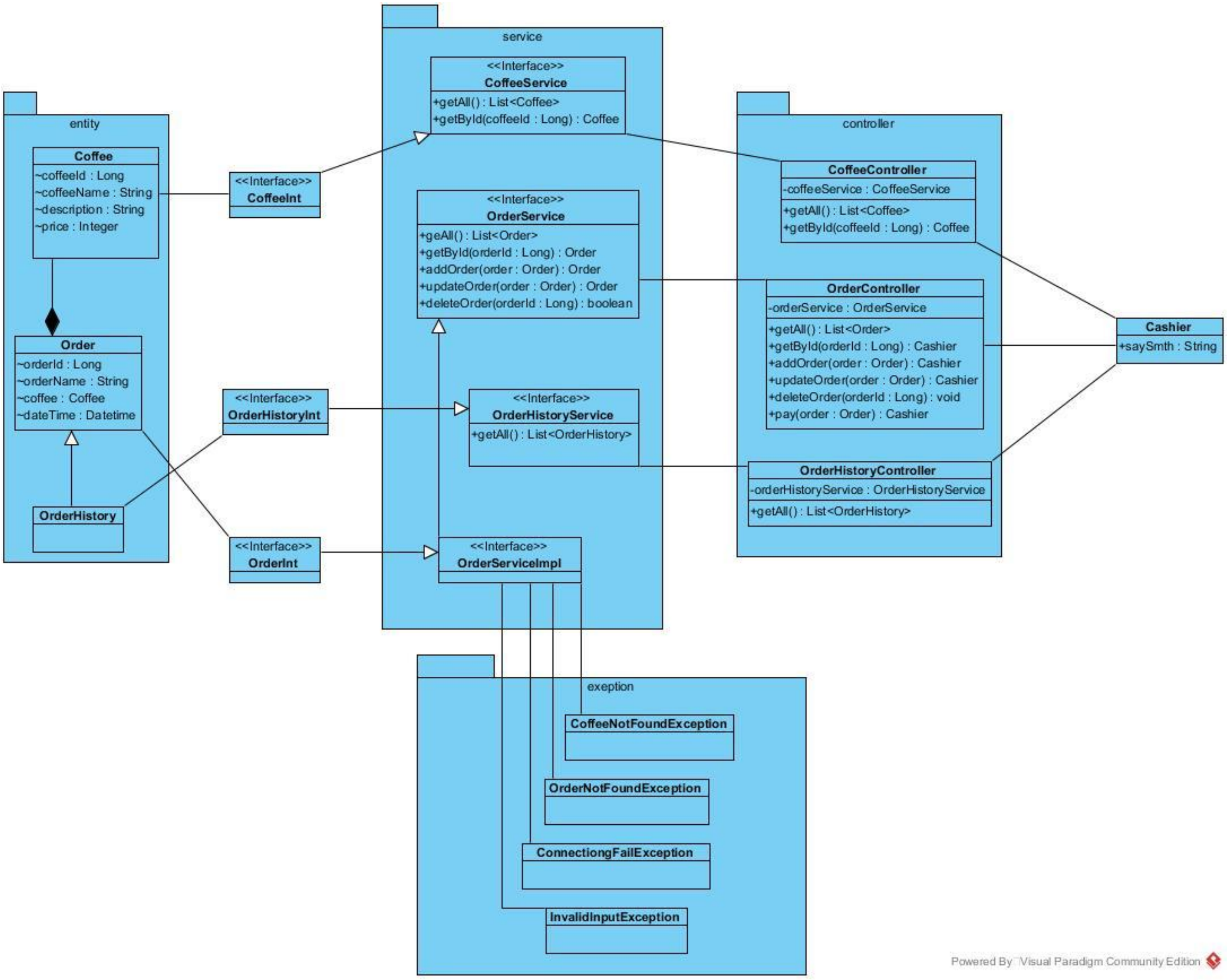
Use Case ID	6		
Use Case Name	Add Order		
Created By	JIrapat	Created By	Jirapat
Date Created	14 / 04 / 2559	Date Created	14 / 04 / 2559
Actors	Cashier		
Description	Use for cashier add order into list of product.		
Trigger	Cashier must click add button.		
Preconditions	Before customer ordered cashier will add order in system.		
Use Case Input Specification			
Input	type	Constraint	Example
Name	String	This value can be input only alphabet. It can’t be number	Latte
Price	double	Amount of money, can’t be negative number.	50
Type	String	Type	Hot / Ice
Amount	int	Amount	1
Post conditions	Add and update list		
Normal Flows	User	System	
	1. Customer order with cashier	2. Cashier receive order 3. Cashier input order client into system 4. System will update data	
Alternative Flow	1. Customer want to change order now 2. Not enough something		
Exception Flow	1. Out of power 2. Server error		
Assumption	Cashier must have ID for login this program.		

Use Case ID	7			
Use Case Name	Update queue			
Created By	Jirapat		Created By	Jirapat
Date Created	14 / 04 / 2559		Date Created	14 / 04 / 2559
Actors	Cashier			
Description	Continues to run a number of queue.			
Trigger	Cashier must click update button.			
Preconditions	Cashier must give queue to customer before update queue.			
Use Case Input Specification				
Input	type	Constraint		Example
Number	Int	This value can be input only number, It can't be negative number.		1, 2, 3, 4
Post conditions	Update status and queue			
Normal Flows	User		Normal Flows	
	2. Customer get number of queue and waiting for product.		1. The system will update queue after cashier give number of queue to customer.	
Alternative Flow	1. Customer cancel queue but they not tell cashier about cancel			
Exception Flow	1. Server error or has bug. 2. Out of power			
Assumption	Customer must pay money with cashier before get queue.			

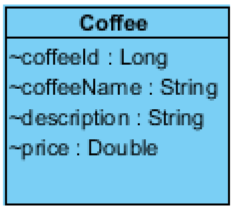
Use Case ID	8			
Use Case Name	Payment update			
Created By	JIrapat		Created By	JIrapat
Date Created	14 / 04 / 2559		Date Created	14 / 04 / 2559
Actors	Cashier			
Description	For update financial accounting in daily.			
Trigger	Customer must pay money to cashier payment system will update.			
Preconditions	Cashier must have money for pay.			
Use Case Input Specification				
Input	type	Constraint		Example
Price	double	Amount of money, It can’t be negative number.		50
Post conditions	Payment will update			
Normal Flows	User		System	
	1. Customer pay money with cashier.		2. Cashier receive money and input amount of money in program. 3. The system will keeping data and update payment.	
Alternative Flow	1. Customer pay money wrong/incorrect			
Exception Flow	1. Out of power 2. Server error			
Assumption	Cashier must have ID for login this program.			

Use Case ID	9		
Use Case Name	Check inventory		
Created By	Jirapat	Last Update By	Jirapat
Date Created	14 / 04 / 2559	Last Revision Date	14 / 04 / 2559
Actors	Customer, Cashier		
Description	Check about items or goods for make product.		
Trigger	Cashier must waiting for check inventory after add order.		
Preconditions	You must have order before add into inventory.		
Use Case Input Specification			
Input	type	Constraint	Example
Name	String	This value can be input only alphabet. It can't be number.	Cappuccino
Post conditions	Update status of inventory		
Normal Flows	User	System	
	1. Customer ordering with cashier	2. Cashier add order in coffee shop system 3. The system will receive order and check inventory before made it.	
Alternative Flow	1. Out of power 2. Server error		
Exception Flow	1. Empty inventory 2. Not enough something		
Assumption	Customer must have inventory.		

Class diagram: Coffee Shop System

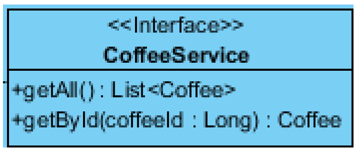


Class Description: Coffee Shop System



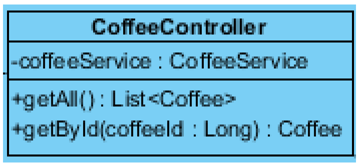
Attribute

ID	Name	Description	Remark
01	coffeeId	Define the ID of coffee to use to search.	Type: Long
02	coffeeName	Define the name of coffee.	Type: String
03	description	Define the description of the coffee to information about this coffee if the customer asked.	Type: String
04	price	Variable is the price of coffee to payment.	Type: Integer



Methods

ID	Name	Description	Remark
01	getAll	A method for display all the coffee in the menu.	Return: Arraylist
02	getById	A method for search the coffee by input coffeeId.	Return: Coffee

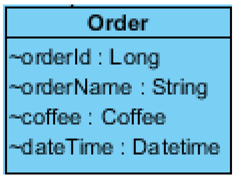

CoffeeController
-coffeeService : CoffeeService
+getAll() : List<Coffee>
+getById(coffeeId : Long) : Coffee

Attribute

ID	Name	Description	Remark
01	coffeeService	Implement CoffeeService interface.	Interface:CoffeeService

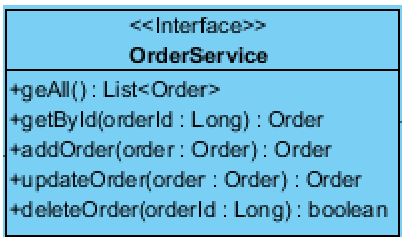
Methods

ID	Name	Description	Remark
01	getAll	A method for display all the coffee in the menu.	Return: Arraylist
02	getById	A method for search the coffee by input coffeeId.	Return: Coffee



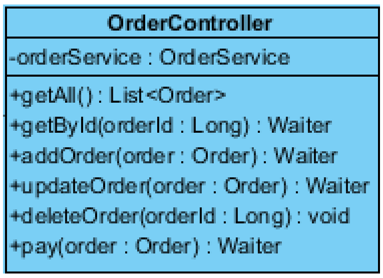
Attribute

ID	Name	Description	Remark
01	orderId	Define the ID of order to use to search.	Type = Long
02	orderName	Define the name who ordered the coffee.	Type = String
03	coffee	Create the coffee object.	Class = Coffee
04	dateTime	Define the date and time of order when the customer ordered the coffee.	Type = Datetime


<<Interface>>
OrderService
+geAll() : List<Order>
+getById(orderId : Long) : Order
+addOrder(order : Order) : Order
+updateOrder(order : Order) : Order
+deleteOrder(orderId : Long) : boolean

Methods

ID	Name	Description	Remark
01	getAll	A method for display all the list of coffee in one order bill.	Return: Arraylist
02	getById	A method for search the order bill by input orderId.	Return: Order
03	addOrder	A method for adding the new order of coffee to the order list when the customer was ordered.	Return: Order
04	updateOrder	A method for update/refreshes the newer of the order list.	Return: Order
05	deleteOrder	A method for deleting/canceling the order of coffee in the order list when the customer has canceled the order.	Return: Boolean


OrderController
-orderService : OrderService
+getAll() : List<Order>
+getById(orderId : Long) : Waiter
+addOrder(order : Order) : Waiter
+updateOrder(order : Order) : Waiter
+deleteOrder(orderId : Long) : void
+pay(order : Order) : Waiter

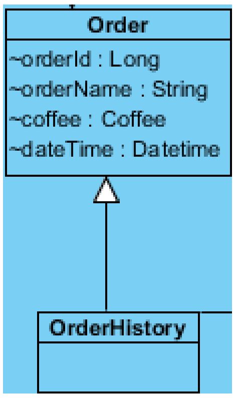
Attribute

ID	Name	Description	Remark
01	orderService	Implement OrderService interface.	Interface:OrderService

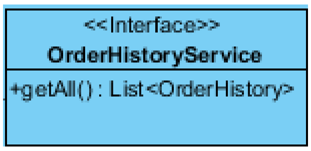
Methods

ID	Name	Description	Remark

01	getAll	A method for display all the list of coffee in one order bill.	Return: Arraylist
02	getById	A method for search the order bill by input orderId.	Return: Waiter
03	addOrder	A method for adding the new order of coffee to the order list when the customer was ordered.	Return: Waiter
04	updateOrder	A method for update/refreshes the newer of the order list.	Return: void
05	deleteOrder	A method for deleting/canceling the order of coffee in the order list when the customer has canceled the order.	Return: Boolean
06	pay	A method for calculate the price and cash to find the change.	Return: Waiter

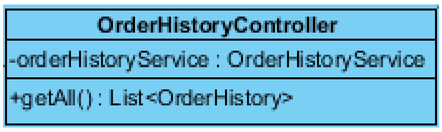


Description: Build the OrderHistory class generalization with Order class.



Methods

ID	Name	Description	Remark
01	getAll	A method for display all the list of coffee in one order bill in history.	Return: Arraylist

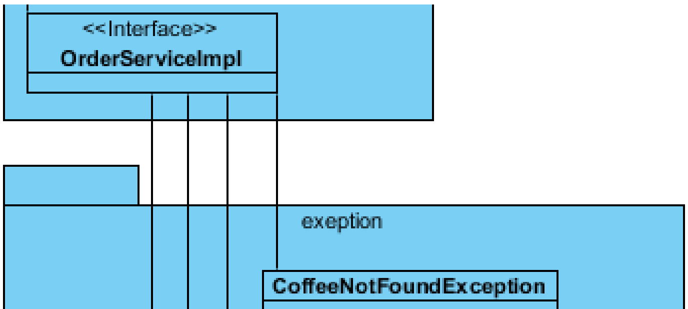

OrderHistoryController
-orderHistoryService : OrderHistoryService
+getAll() : List<OrderHistory>

Attribute

ID	Name	Description	Remark
01	orderHistoryService	Implement OrderHistoryService interface.	Interface: OrderHistoryService

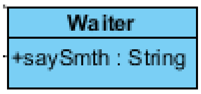
Methods

ID	Name	Description	Remark
01	getAll	A method for display all the list of coffee in one order bill in history.	Return: Arraylist



Attribute

ID	Name	Description	Remark
01	CoffeeNotFoundException	An exception if a method getById search the coffee by input coffeeId cannot found.	
02	OrderNotFoundException	An exception if a method getById search the order by input orderId cannot found.	
03	ConnectionFailException	An exception if the connection of the server has disconnected.	
04	InvalidInputException	An exception if the cashier has inputted the invalid information.	

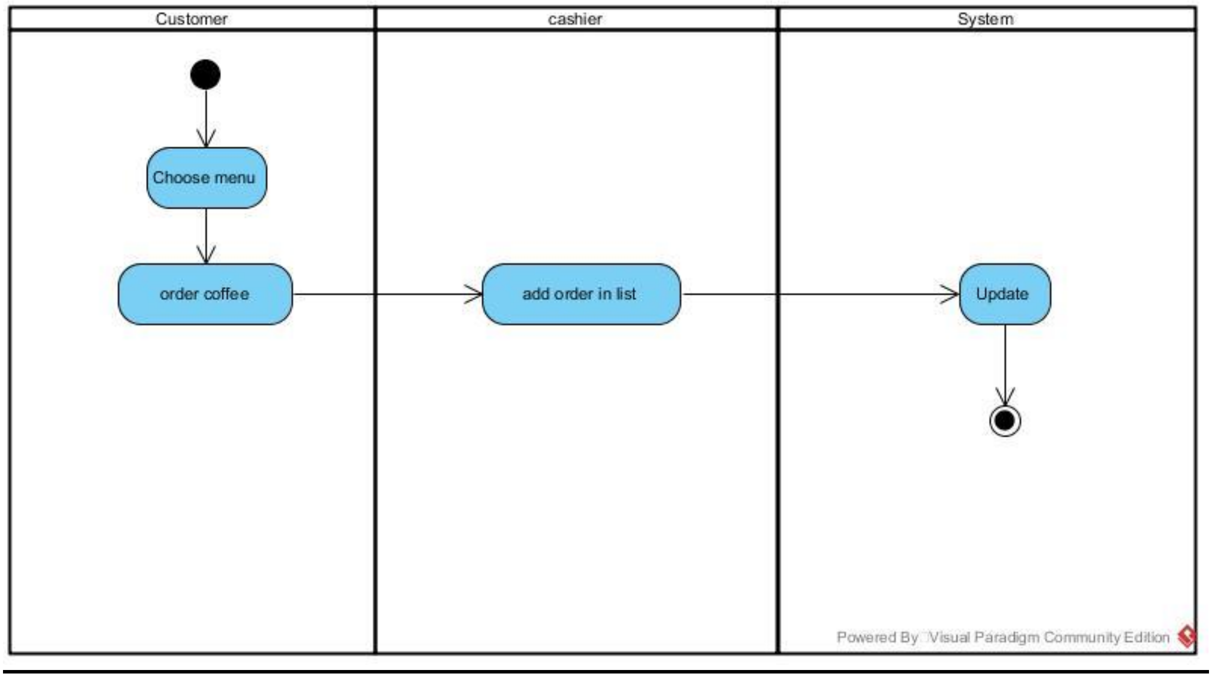


Attribute

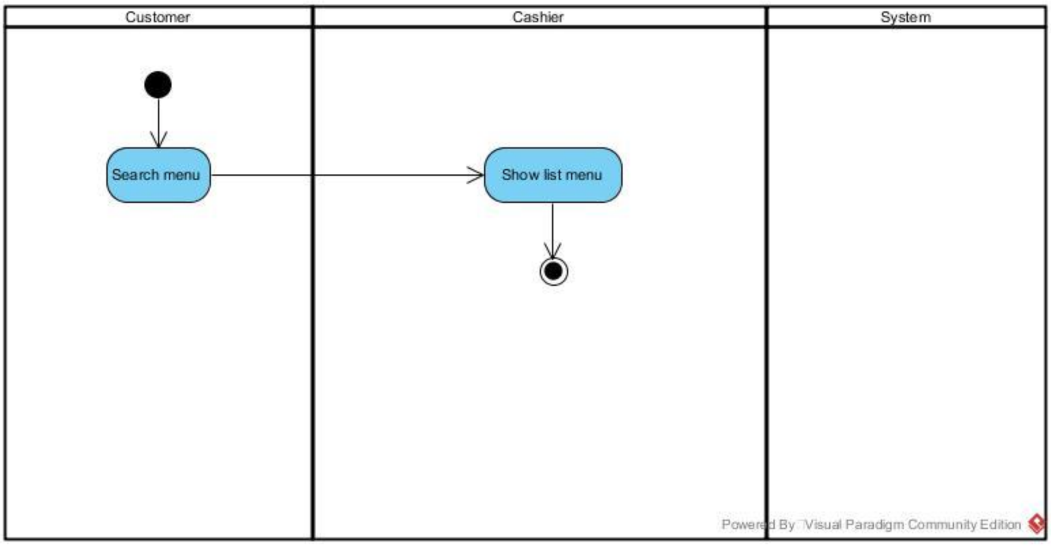
ID	Name	Description	Remark
01	saySmth	The cashier needs to say something with the customer (The order has finished, Out of stock, …..).	Type: String

Activity diagram: Coffee shop system

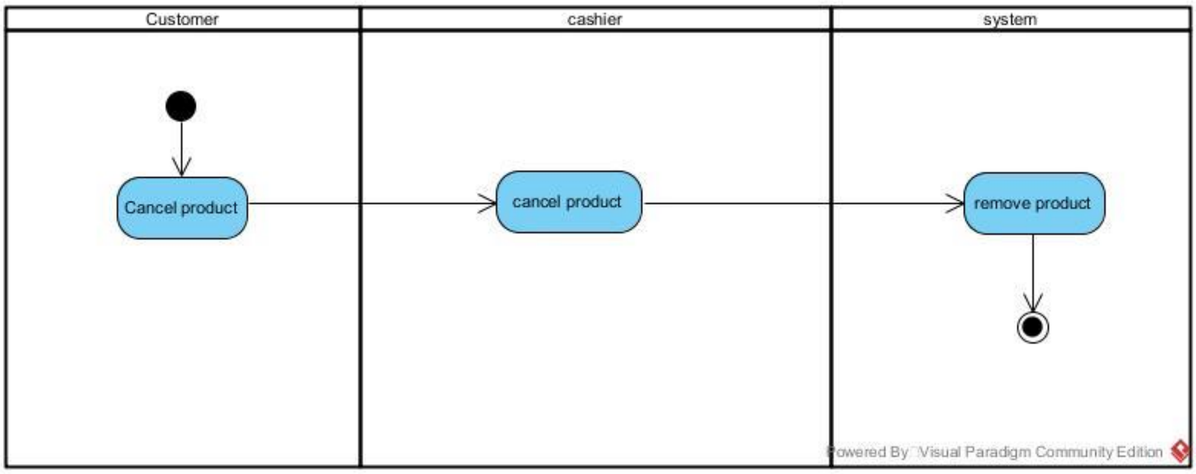
1. Order



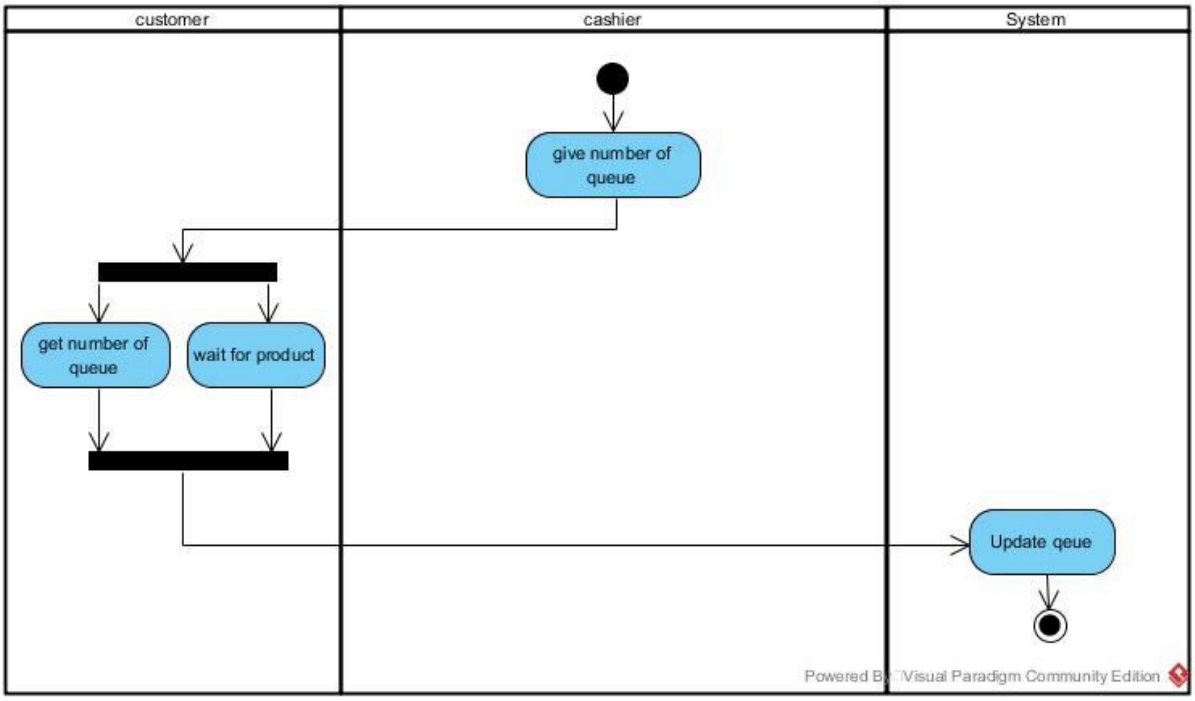
2. Search



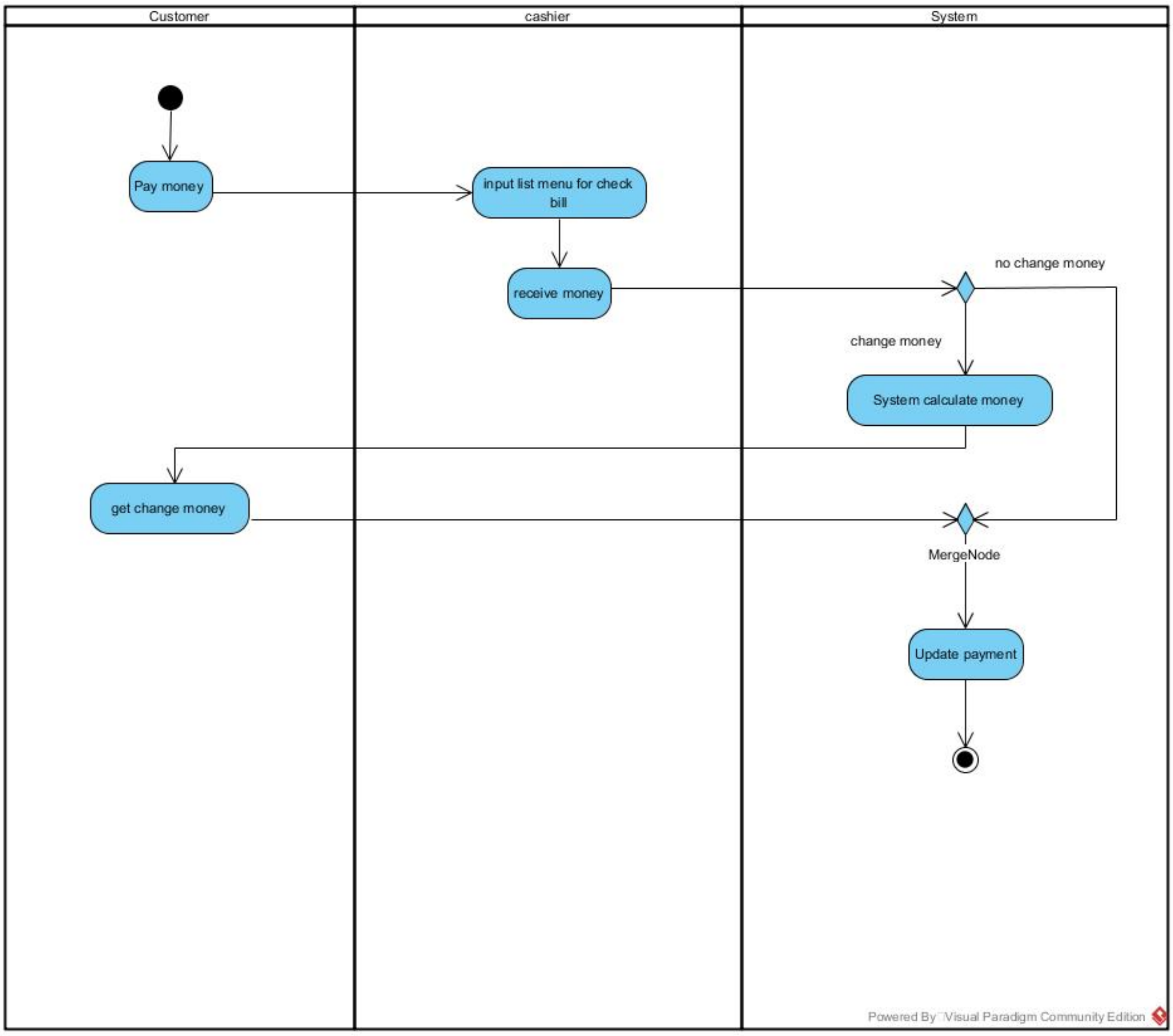
3. Cancel



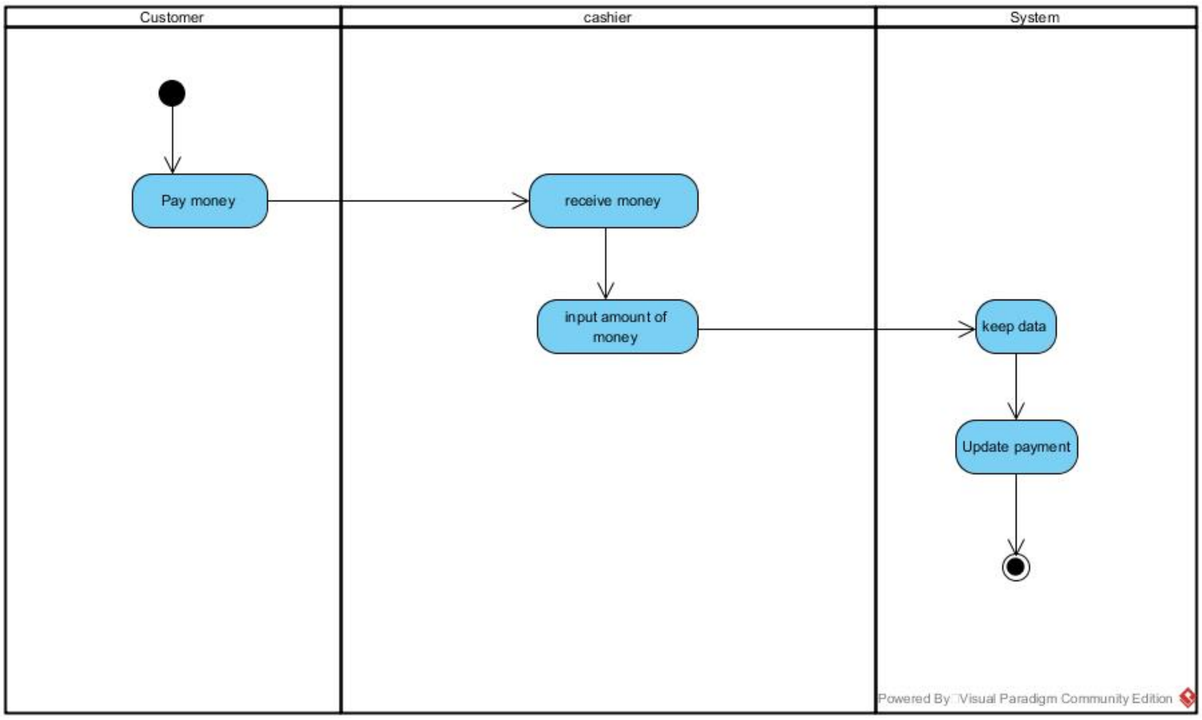
4. Update queue



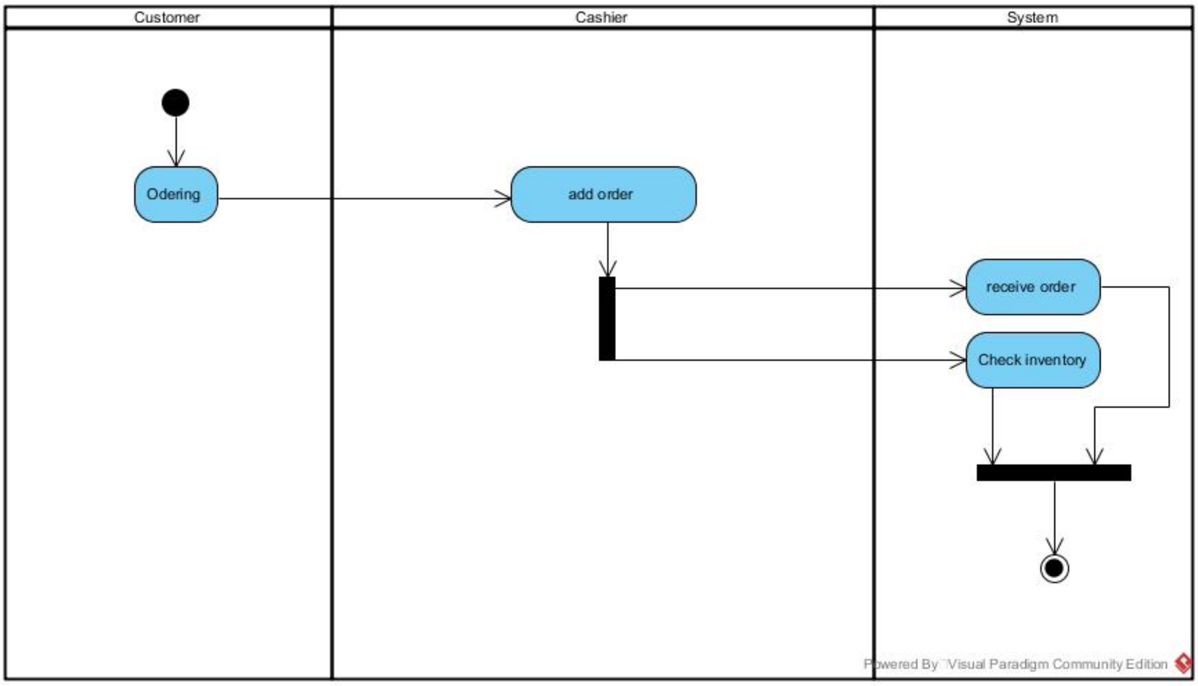
5. Payment



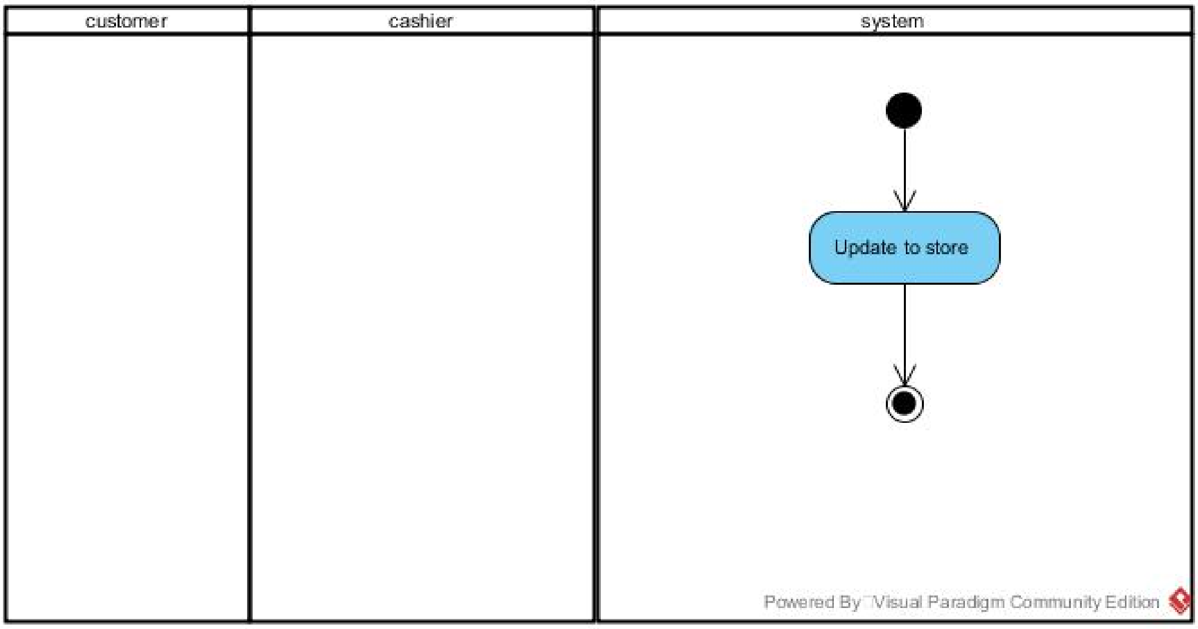
6. Payment Update



7. Check Inventory

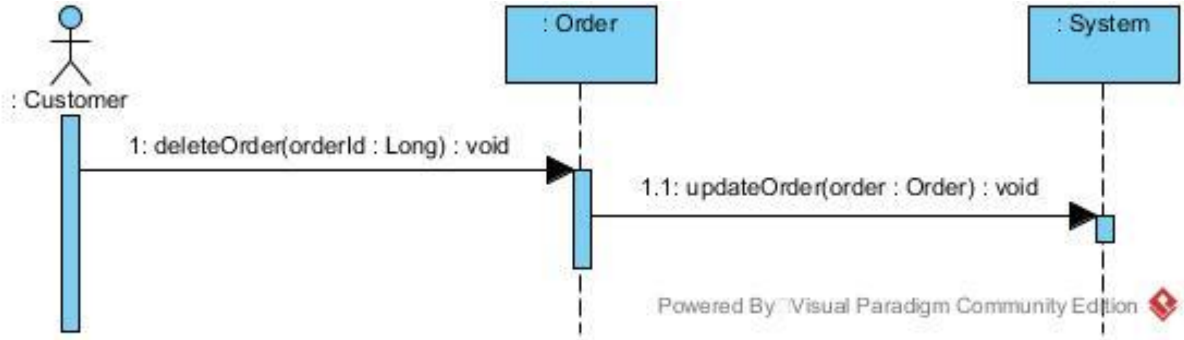


8. Store address

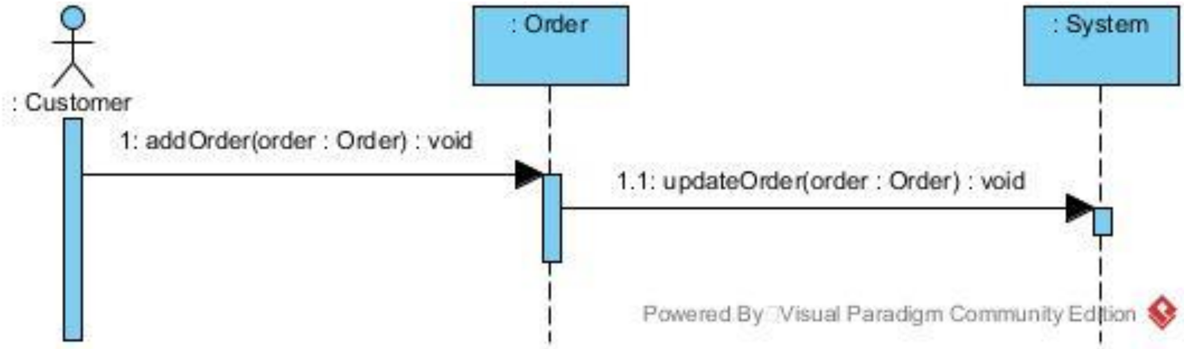


Sequence diagram: Coffee Shop System

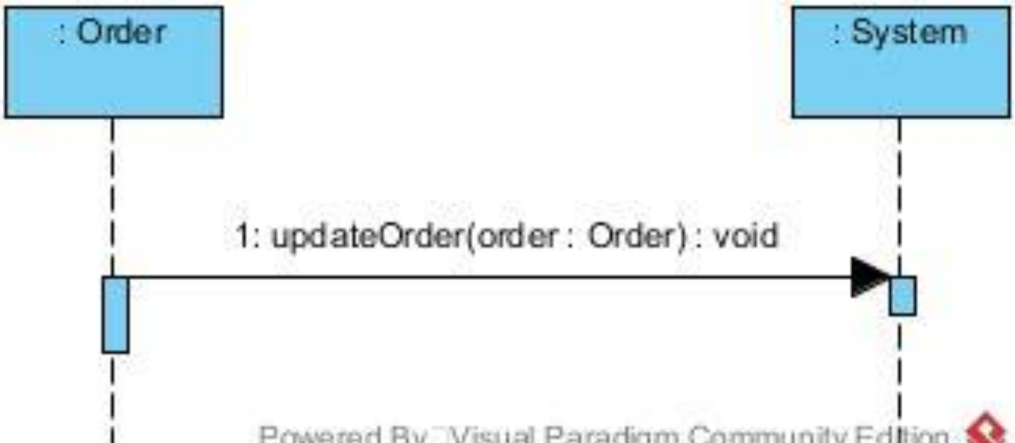
1. Cancel


: Customer
: Order
: System
1: deleteOrder(orderId : Long) : void
1.1: updateOrder(order : Order) : void
Powered By Visual Paradigm Community Edition

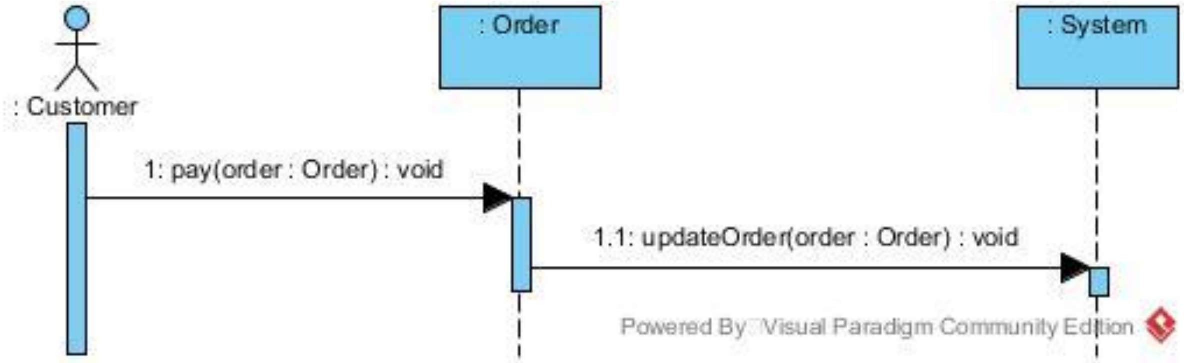
2. Order


: Customer
: Order
: System
1: addOrder(order : Order) : void
1.1: updateOrder(order : Order) : void
Powered By Visual Paradigm Community Edition

3. OrderHit


: Order
: System
1: updateOrder(order : Order) : void
Powered By Visual Paradigm Community Edition

4. Payment


: Customer
: Order
: System
1: pay(order : Order) : void
1.1: updateOrder(order : Order) : void
Powered By Visual Paradigm Community Edition