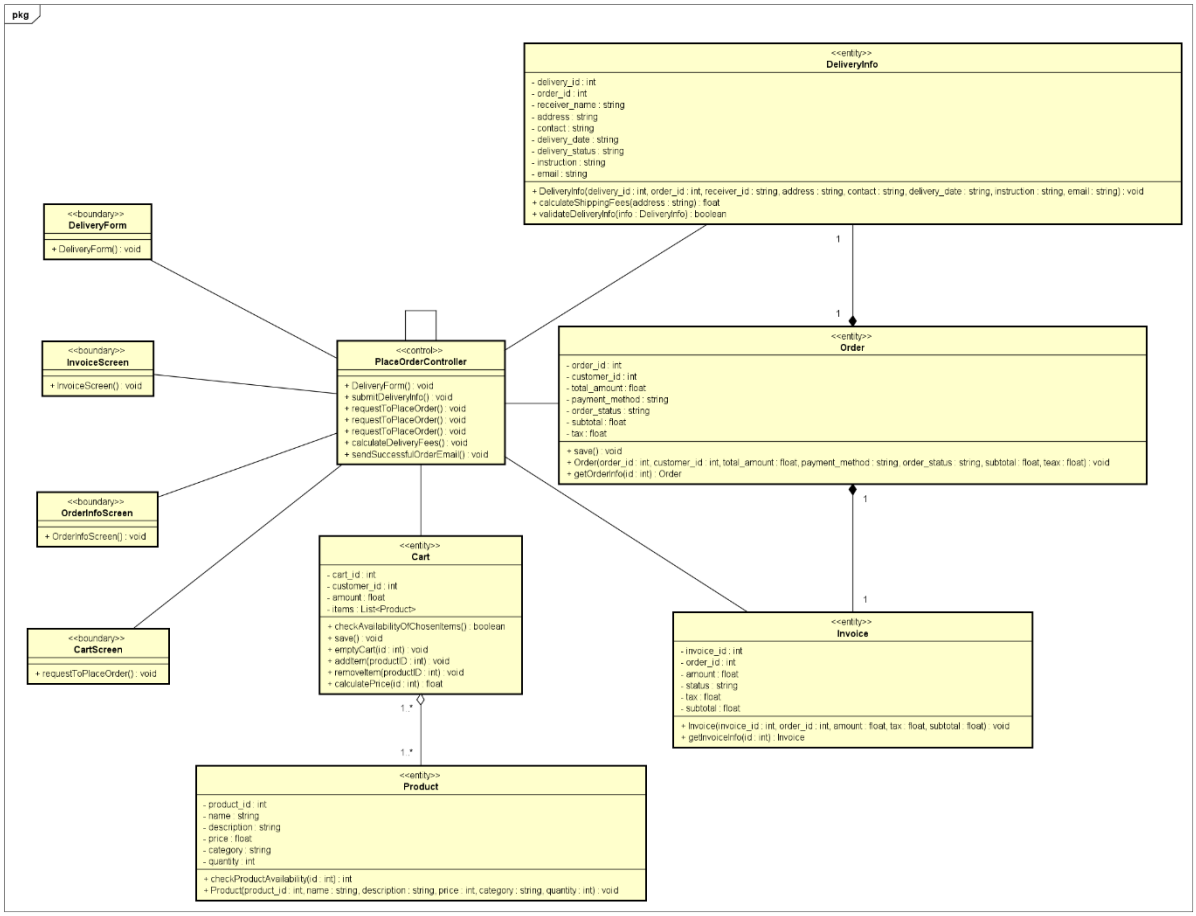
Use case: Place Order

Place Order Class diagram

Note: methods set/get for each entities are many and simple, so they will not be included in diagram just illustrated in class ‘DeliveryInfo’ bellow as an example



Design class

1. DeliveryInfo

Attributes table:

#	Name	Data type	Default value	Description

1	delivery_id	int	null	unique identifier for each delivery
2	order_id	int	null	unique identifier for order associated with delivery
3	receiver_name	string	null	name of the receiver
4	address	string	null	delivery address
5	contact	string	null	phone number of receiver
6	delivery_date	string	null	expected delivery date
7	delivery_status	string	'Processing'	status of delivery
8	instruction	string	null	instruction given by customer

Operations table:

#	Name	Return type	Description
1	setDeliveryAddress(address: string)	void	update delivery address
2	getDeliveryAddress()	string	return delivery address
3	setReceivername(name: string)	void	update name of receiver
4	getReceiverName()	string	return receiver's name
5	setContact(contact: string)	void	update receiver phone number
6	getContact()	string	return receiver's phone number

7inf	setDeliveryDate(date: string)	void	update expected delivery date
8	getDeliveryDate()	string	return expected delivery date
9	setDeliveryStatus(status: string)	void	update current delivery status
10	getDeliveryStatus()	string	return current delivery status
12	calculateShippingFee(address: string)	float	calculate shipping fees based on receiver address
13	validateDeliveryInfo(info: DeliveryInfo)	boolean	check if information input to delivery form is valid or not
14	DeliveryInfo(delivery_id : int, order_id : int, receiver_id : string, address : string, contact : string, delivery_date : string, instruction : string, email : string)	void	construction method to create a new DeliveryInfo

Parameters:

- delivery_id: int, this parameter is used to view the information of delivery
- order_id: int, this parameter is used to view the information of delivery
- receiver_name: string, this parameter is used to view the information of delivery
- address: string, this parameter is used to view the information and calculate shipping fees of delivery
- contact: string, this parameter is used to view the information of delivery
- delivery_date: string, this parameter is used to view the information of delivery
- delivery_status: string, this parameter is used to view the information of delivery
- instruction: string, this parameter is used to view the information of delivery

Exceptions:

- DatabaseConnectionException: exception thrown when database is unreachable while updating or fetching delivery information

Methods:

- set/get… methods: update delivery information from delivery form/ return delivery information
- calculateShippingFees(address: string): float
    - when delivery address is updated, shipping fees is automatically calculated and updated
- validateDeliveryInfo(info: DeliveryInfo): Boolean
    - when any fields updated, check their validity
    - if all fields are valid, return True
    - if any field is invalid, return False
- DeliveryInfo(delivery_id : int, order_id : int, receiver_id : string, address : string, contact : string, delivery_date : string, instruction : string, email : string): void
    - create a new DeliveryInfo
    - when first created, the delivery_status will be 'Processing' by default

States:

- Valid: all delivery fields are valid
- Invalid: any field is invalid

2. Order

Attributes table:

#	Name	Data type	Default value	Description
1	order_id	int	null	unique identifier for each order
2	customer_id	int	null	unique identifier for each customer

				associated with order
3	total_amount	float	null	total money of products and tax
4	subtotal	float	null	total money of products
5	tax	float	null	
6	order_status	string	‘Processing’	status of order
7	payment_method	string	null	method customer choose to pay

Operations table:

#	Name	Return type	Description
1	save()	void	store order information into database
2	Order(order_id : int, customer_id : int, total_amount : float, payment_method : string, order_status : string, subtotal : float, teax : float)	void	construction method to create a new Order
3	getOrderInfo(id: int)	Order	retrieve all information of an order

Parameters:

- order_id: int, this parameter is used to view the information of order
- customer_id: int, this parameter is used to view the information of order
- total_amount: float, this parameter is used to view the information of order
- subtotal: float, this parameter is used to view the information of order
- tax: float, this parameter is used to view the information of order

- order_status: string, this parameter is used to view the information of order
- payment_method: string, this parameter is used to view the information of order

Exception:

- InvalidOrderException: when order details are incomplete or invalid
- DatabaseConnectionException: database is unreachable while processing order
- OrderNotFoundException: when request for order based on order ID does not exist

Methods:

- save(): void
    - when order is created, order is stored to database
- Order(order_id : int, customer_id : int, total_amount : float, payment_method : string, subtotal : float, teax : float): void
    - contruction method to create a new order
    - when created, order_status is 'Processing' by default
- getOrderInfo(): Order
    - retrieve all order information

States:

- Processing: order is placed and waiting for acception or rejection
- Accepted: order is accepted by product manager
- Rejected: order is rejected by product manager
- Delivered: order is delivered to destination
- Cancelled: order is cancelled by customer

3. Invoice

Attributes table:

#	Name	Data type	Default value	Description
1	invoice_id	int	null	unique identifier for each invoice
2	order_id	int	null	unique identifier for order associated with invoice

3	amount	float	null	total amount to pay
4	status	string	‘Waiting payment’	status of invoice
5	subtotal	float	null	total amount of products
6	tax	float	null	

Operations table:

#	Name	Return type	Description
1	Invoice(invoice_id : int, order_id : int, amount : float, tax : float, subtotal : float)	void	construction method to create a new Invoice
2	getInvoiceInfo(id: int)	Invoice	input invoice_id and retrieve invoice’s information

Parameters:

- invoice_id: int, this parameter is used to view the information of invoice
- order_id: int, this parameter is used to view the information of invoice
- amount: float, this parameter is used to view the information of invoice
- status: string, this parameter is used to view the information of invoice
- tax: float, this parameter is used to view the information of invoice
- subtotal: float, this parameter is used to view the information of invoice

Exception:

- InvoiceNotFoundException: when request for invoice based on invoice ID does not exist
- PaymentFailedException: when payment transaction cannot be completed (insufficient fund,…)
- InvoiceAlreadyPaidException: when payment for this invoice has already been made
- DatabaseConnectionException: when database is unreachable while fetching or updating invoice

Methods:

- Invoice(invoice_id : int, order_id : int, amount : float, tax : float, subtotal : float): void
    - this is a construction method used to create new invoice when order is accepted
- getInvoiceInfo(id: int): Invoice
    - retrieve all invoice information

States:

- Waiting payment: when invoice has been created, and customer has not paid yet
- Paid: customer has paid

4. Cart

Attributes table:

#	Name	Data type	Default value	Description
1	cart_id	int	null	unique identifier for each cart, but this ID is temporary
2	customer_id	int	null	unique identifier for customer associated with cart
3	amount	float	null	total amount of products in cart
4	items	List<Product>	null	list of product int cart; if there are more than 1 product of the same type, it will appear many time in the list

Operations table:

#	Name	Return type	Description
1	checkAvailabilityOfChosenItems(List<Product>)	boolean	loop through list of items of a cart and check availability of each product
2	save()	void	temporarily save cart for processing, not store to database
3	emptyCart(id: int)	void	empty cart with given id
4	addItem(productID: int)	void	add item to cart
5	removeItem(productid: int)	void	remove product from cart
6	calculatePrice(id: int)	float	calculate total price of products in a cart

Parameters:

- cart_id: int, this this parameter is used to view the information of cart
- customer_id: int, this this parameter is used to view the information of cart
- amount: float, this this parameter is used to view the information of cart
- items: list<Product>, this this parameter is used to view the information of cart

Exception:

- AddingItemFailedException: when cannot add item into cart for some reasons (not enough quantity, wrong product id,…)
- RemoveItemFailedException: when cannot remove item from cart because item has been removed

Methods:

- checkAvailabilityOfChosenItems(List<Product>): Boolean
    - check availability of chosen item before adding to cart
- save(): void
    - save cart temporarily on system for later processing
- emptyCart(id: int): void
    - used to empty cart after place order complete
- addItem(productID: int): void
    - used to add chosen items into cart
- removeItem(productid: int): void
    - remove chosen items from cart
- calculatePrice(id: int): float
    - calculate total price of products in cart

States:

- Empty: cart is existing but no items inside
- Active: cart contains items inside, but no further process
- Processing: user choose to place order for this cart and system is processing further steps
- If user left before place order, cart will be deleted, so this is no state

5. Product

Attributes table:

#	Name	Data type	Default value	Description
1	product_id	int	null	unique identifier of each product
2	name	string	null	name of product
3	description	string	null	description of product

4	price	float	null	price of product
5	category	string	null	category of product
6	quantity	int	null	quantity of product

Operations table:

#	Name	Return type	Description
1	Product(product_id : int, name : string, description : string, price : int, category : string, quantity : int)	void	construction method to create a new product
2	checkProductAvailability(id: int)	int	check product availability

Parameters:

- product_id: int, this this parameter is used to view the information of product
- name: string, this this parameter is used to view the information of product
- description: string, this this parameter is used to view the information of product
- price: float, this this parameter is used to view the information of product
- category: string, this this parameter is used to view the information of product
- quantity: int, this this parameter is used to view the information of product

Exception:

- DuplicateProductException: when intending to change product quantity or price, product manager tries to create a product has already exist (check based on name) instead of update product quantity or prices
- DatabaseConnectionException: database is unreachable while retrieving product data

Methods:

- Product(product_id : int, name : string, description : string, price : int, category : string, quantity : int): void

    - a construction method used to create a new product
    - this also check if the product has already existed in database
- checkProductAvailability(id: int): int
    - retrieve the number of available products
    - return 0 if there is no products left

States:

- Available: product's quantity is available in database
- Unavailable: product's quantity is unavailable in database