Advance computer programming final project

John remon maher	**2000150**
Mina samer rizk	**2001369**
Mario magdy saber	**2000980**
Ahmed Mahmoud Hamed Shawareb	**20P7699**

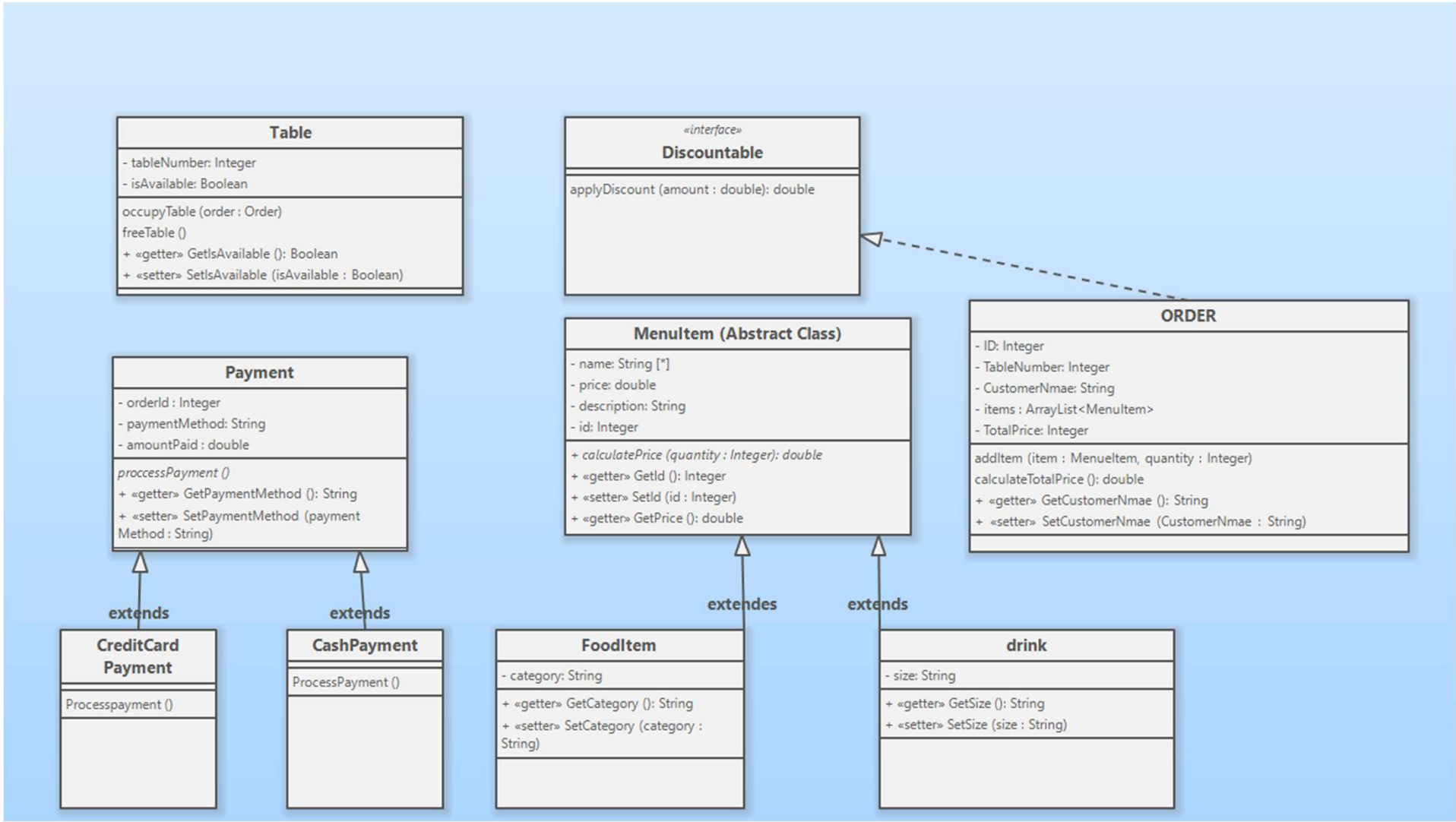
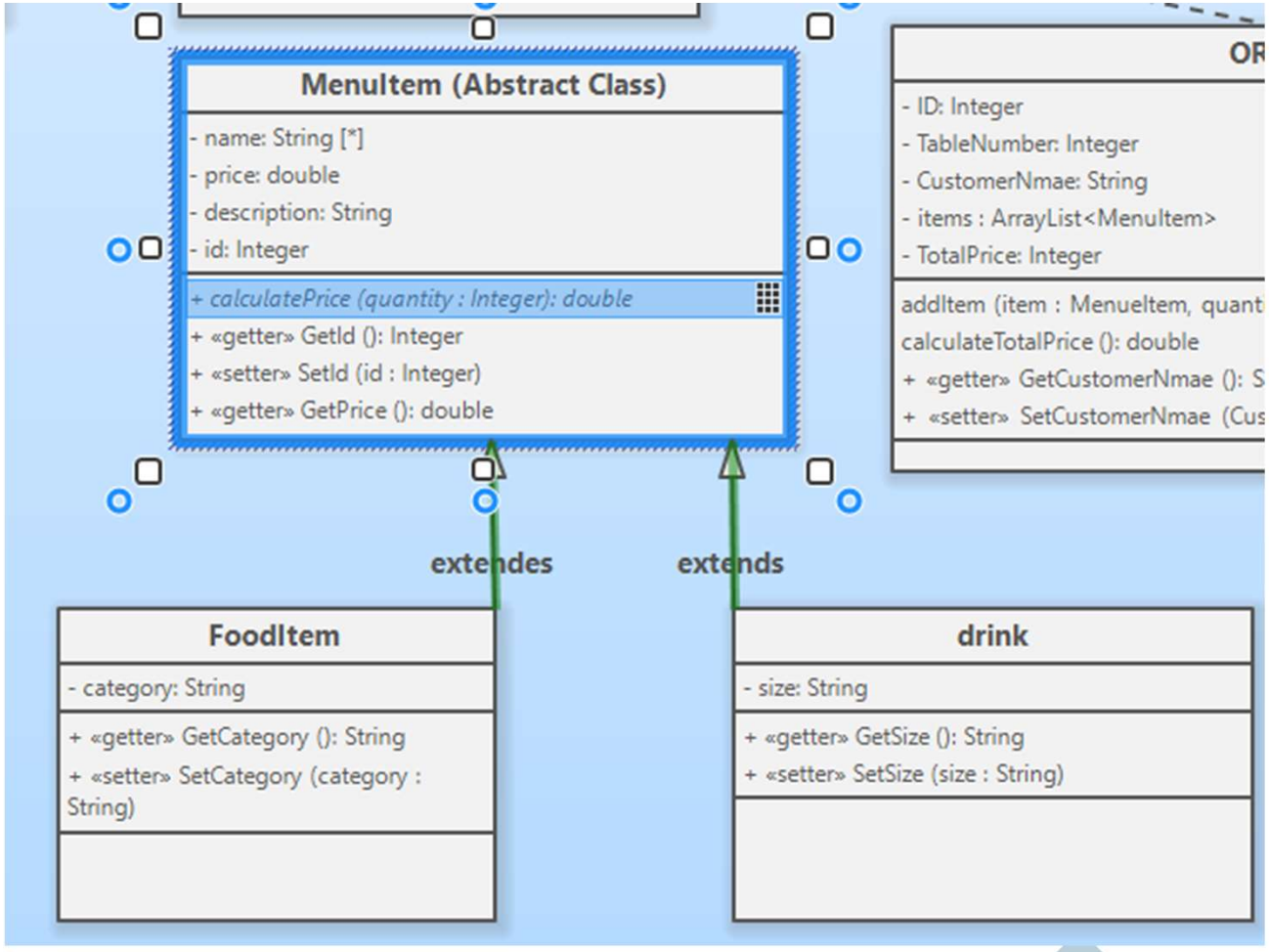
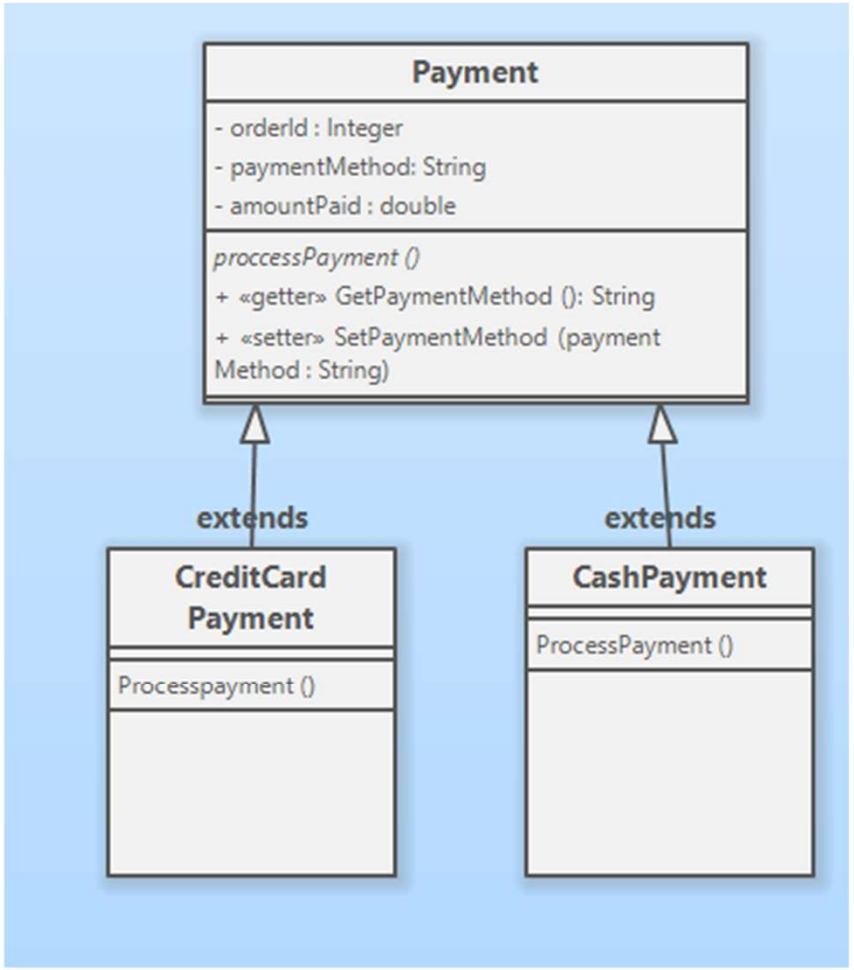


Table
- tableNumber: Integer
- isAvailable: Boolean
occupyTable (order : Order)
freeTable ()
+ «getter» GetIsAvailable (): Boolean
+ «setter» SetIsAvailable (isAvailable : Boolean)
«interface»
Discountable
applyDiscount (amount : double): double
ORDER
- ID: Integer
- TableNumber: Integer
- CustomerNmae: String
- items : ArrayList<MenuItem>
- TotalPrice: Integer
addItem (item : MenueItem, quantity : Integer)
calculateTotalPrice (): double
+ «getter» GetCustomerNmae (): String
+ «setter» SetCustomerNmae (CustomerNmae : String)
MenuItem (Abstract Class)
- name: String [*]
- price: double
- description: String
- id: Integer
+ calculatePrice (quantity : Integer): double
+ «getter» GetId (): Integer
+ «setter» SetId (id : Integer)
+ «getter» GetPrice (): double
Payment
- orderId : Integer
- paymentMethod: String
- amountPaid : double
proccessPayment ()
+ «getter» GetPaymentMethod (): String
+ «setter» SetPaymentMethod (payment Method : String)
extends
extends
extendes
extends
CreditCard Payment
Processpayment ()
CashPayment
ProcessPayment ()
FoodItem
- category: String
+ «getter» GetCategory (): String
+ «setter» SetCategory (category : String)
drink
- size: String
+ «getter» GetSize (): String
+ «setter» SetSize (size : String)


MenuItem (Abstract Class)
- name: String [*]
- price: double
- description: String
- id: Integer
+ calculatePrice (quantity : Integer): double
+ «getter» GetId (): Integer
+ «setter» SetId (id : Integer)
+ «getter» GetPrice (): double
OR
- ID: Integer
- TableNumber: Integer
- CustomerNmae: String
- items : ArrayList<MenuItem>
- TotalPrice: Integer
addItem (item : MenueItem, quant
calculateTotalPrice (): double
+ «getter» GetCustomerNmae (): S
+ «setter» SetCustomerNmae (Cus
extendes
extends
FoodItem
- category: String
+ «getter» GetCategory (): String
+ «setter» SetCategory (category : String)
drink
- size: String
+ «getter» GetSize (): String
+ «setter» SetSize (size : String)


Payment
- orderId : Integer
- paymentMethod: String
- amountPaid : double
proccessPayment ()
+ «getter» GetPaymentMethod (): String
+ «setter» SetPaymentMethod (payment Method : String)
extends
extends
CreditCard Payment
Processpayment ()
CashPayment
ProcessPayment ()

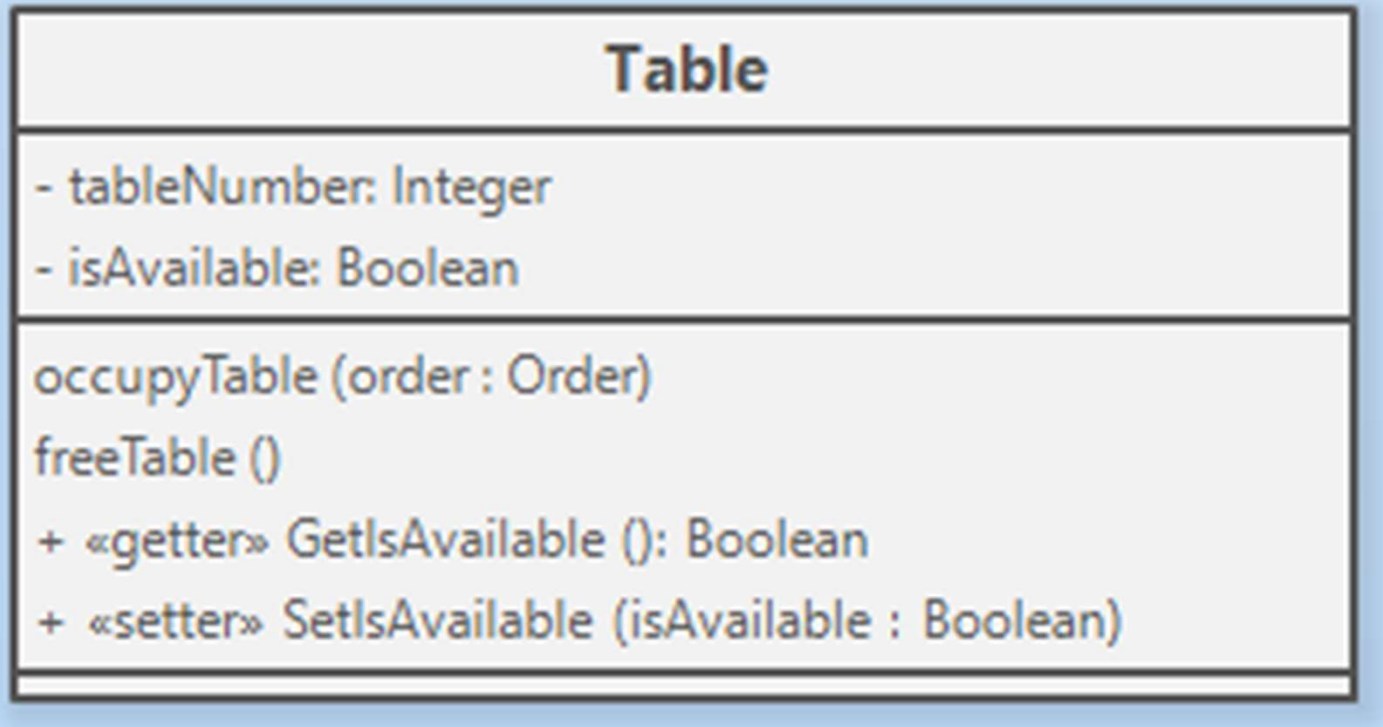
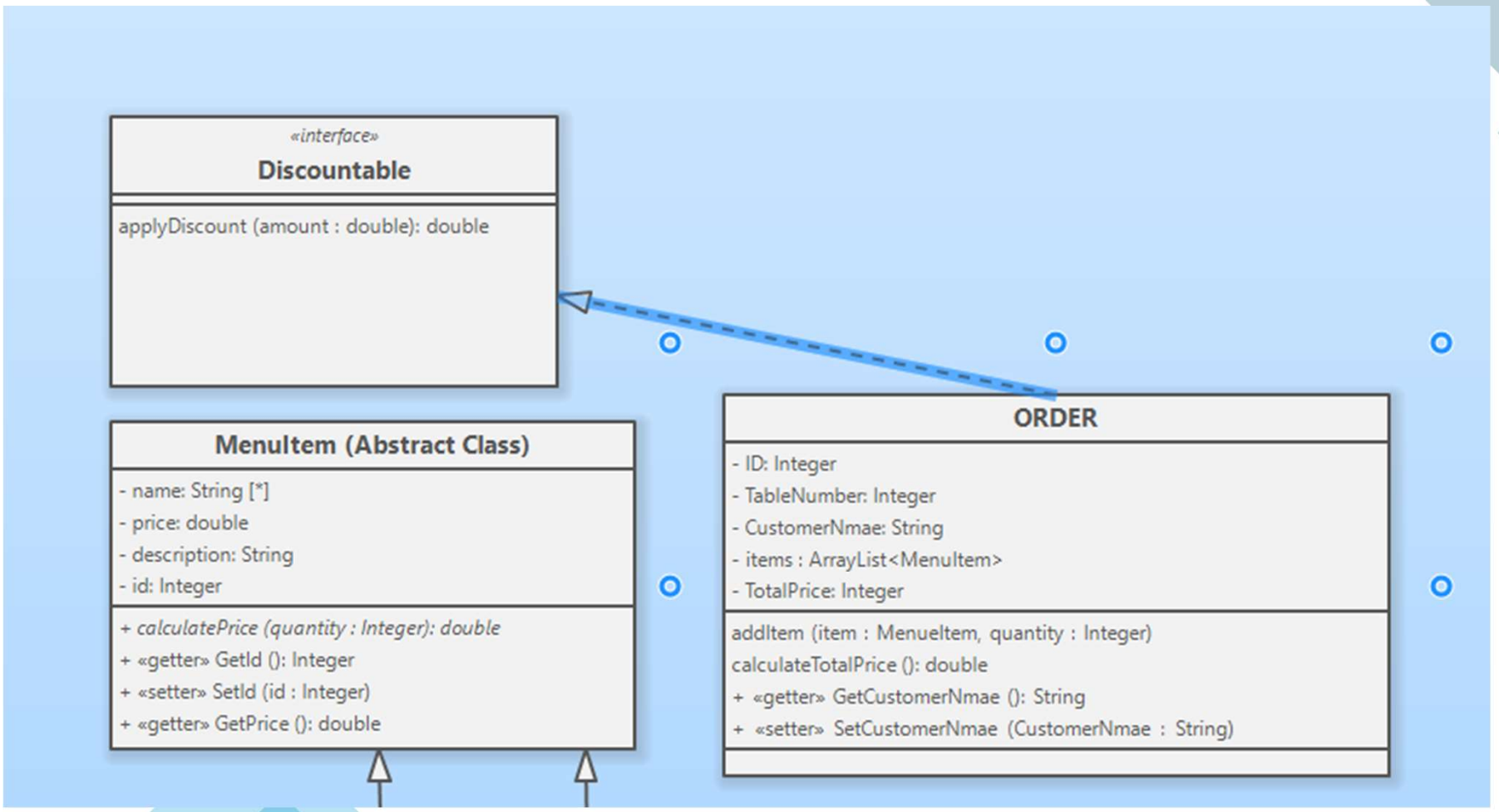


Table
- tableNumber: Integer
- isAvailable: Boolean
occupyTable (order : Order)
freeTable ()
+ «getter» GetIsAvailable (): Boolean
+ «setter» SetIsAvailable (isAvailable : Boolean)


«interface»
Discountable
applyDiscount (amount : double): double
MenuItem (Abstract Class)
- name: String [*]
- price: double
- description: String
- id: Integer
+ calculatePrice (quantity : Integer): double
+ «getter» GetId (): Integer
+ «setter» SetId (id : Integer)
+ «getter» GetPrice (): double
ORDER
- ID: Integer
- TableNumber: Integer
- CustomerNmae: String
- items : ArrayList<MenuItem>
- TotalPrice: Integer
addItem (item : MenueItem, quantity : Integer)
calculateTotalPrice (): double
+ «getter» GetCustomerNmae (): String
+ «setter» SetCustomerNmae (CustomerNmae : String)

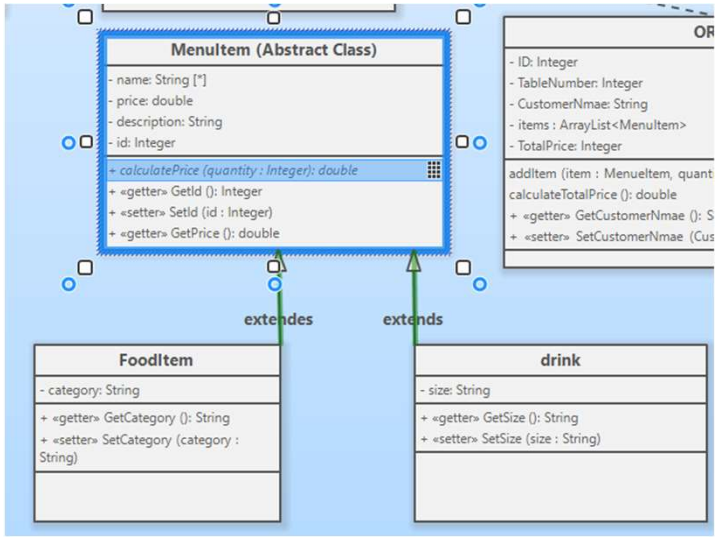
Now lets try the code and then we will discuses how we applied :

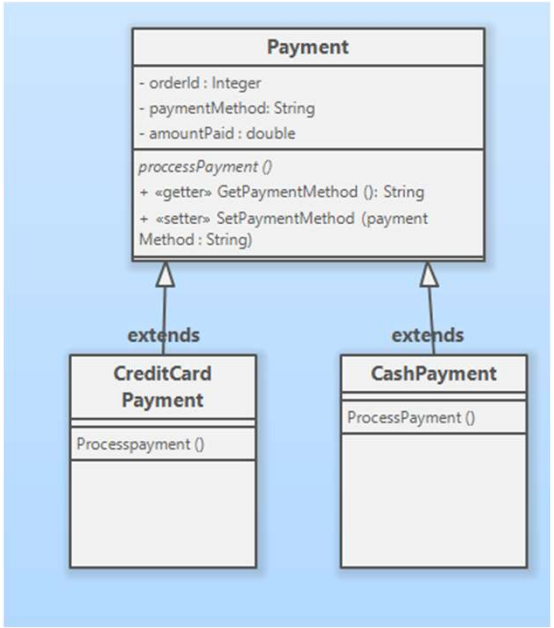
1- inheritance
2- polymorphism
3- using concrete or abstract super classes
4- at least one interface
5- using the generic
6- exception handling

How we applied inheritance

- As we saw from the uml that fooditem class extends minueitem

And also drink extends minueitem so inheritance was applies also cash payment and credit card payment extends payment





- **public class** FoodItem **extends** MenueItem
- **public class** Drink **extends** MenueItem
- **public class** CreditCardPayment **extends** Payment
- **public class** CashPayment **extends** Payment

How polymorphism was applies

- Polymorphism was applies in many places in the code for example

```
MenueItem item1 = new FoodItem("beaf burger", 66 ,"beaf burger with cheese ", 1, "Main Cou
MenueItem item2 = new FoodItem("Calamari", 50 ," Lightly fried squid rings served with mar
MenueItem item3 = new FoodItem("Chicken Wings", 80 ,"Crispy fried or baked chicken wings t
MenueItem item4 = new FoodItem("Chicken Parmesan", 100 ,"Breaded and fried chicken breast
MenueItem item5 = new FoodItem("Pasta Primavera", 60 ,"Seasonal vegetables tossed with pas
MenueItem item6 = new FoodItem("Chocolate Cake", 45 ,"Rich and decadent chocolate cake wit
MenueItem item7 = new FoodItem("Fruit Salad", 30 ,"Seasonal fresh fruits with a light hone
MenueItem item8 = new Drink("Coca-Cola", 10 , "big Coca-Cola", 8 , "big");
MenueItem item9 = new Drink("Orange juice", 15 , "big Orange juice", 9 , "big");
MenueItem item10 = new Drink("Coca-Cola", 7 , "small Coca-Cola", 10 , "small");
MenueItem item11 = new Drink("Orange juice", 10 , "medimum Orange juice", 11 , "medium");
```

```
t
case 1:
    Payment payment1 = new CashPayment(0, "cash payment",total_price);
    payment1.processPayment();
    break;
case 2:
    Payment payment2 = new CreditCardPayment(0, "cash payment",total_price);
    payment2.processPayment();
    break;
default:
```

```java
@Override //this is applying polymorphism
public double CalculatPrice() {
    return super.getPrice() ;


}


@Override
public double applyDiscount(double amount) {

    return calculateTotalPrice() - amount;
}


@Override
public void processPayment() {
    System.out.println("CreditCardPayment is processed");

}
```

using concrete or abstract super classes

- As we saw in the code
  the minueitem class is an abstract class
- And payment class is an abstract class

```
public abstract class MenueItem {
```

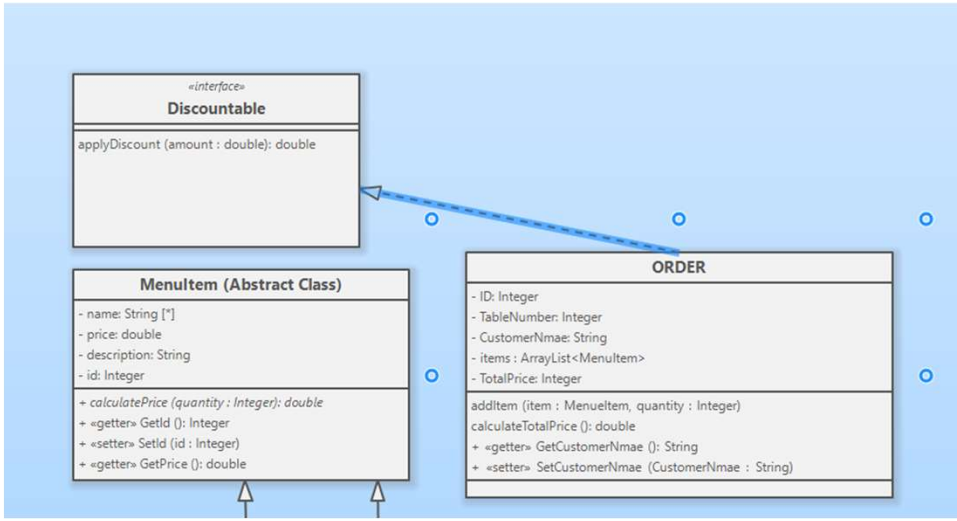
```
public abstract class Payment {
```

at least one interface

As we saw in the uml discountable is an interface and order implements it if the order can have discount

```
5
6 public class Order implements Discountable
```

```
3 public interface Discountable {
```



using the generic

```
ArrayList<MenueItem> items = new ArrayList<MenueItem>();
```

exception handling

There are many exception handling in the code

Every switch statement have a exception as there are many other exceptions

```
default:
    throw new IllegalArgumentException("Unexpected value: " + choice);
}
```

```
try
{
    if (dicount_amount > total_price)
    {
        throw new IllegalAccessException("the dicount is bigger than the order
    }
    else
    {
        total_price =order.applyDiscount(dicount_amount);
        System.out.println("the order price after dicount : " + total_price );
    }
}
catch (Exception e) {
System.out.println(e);
}
```

Thanks
any questions