Collaboration and Multiplicity

Problem Scenario

Let's suppose a store needs to save the information of the product and the customer who has bought the product.

Store wants to calculate the total purchases of a customer. Store wants to calculate the tax on the purchased products as well.

Problem Scenario: Product

Product information includes productName, Category, Price, and a behaviour that calculates the Tax

Problem Scenario: Customer

Customer information includes, Name, Address and Contact Number and a behaviour that can get the information of all Products the customer has bought.

Problem Scenario: Solution

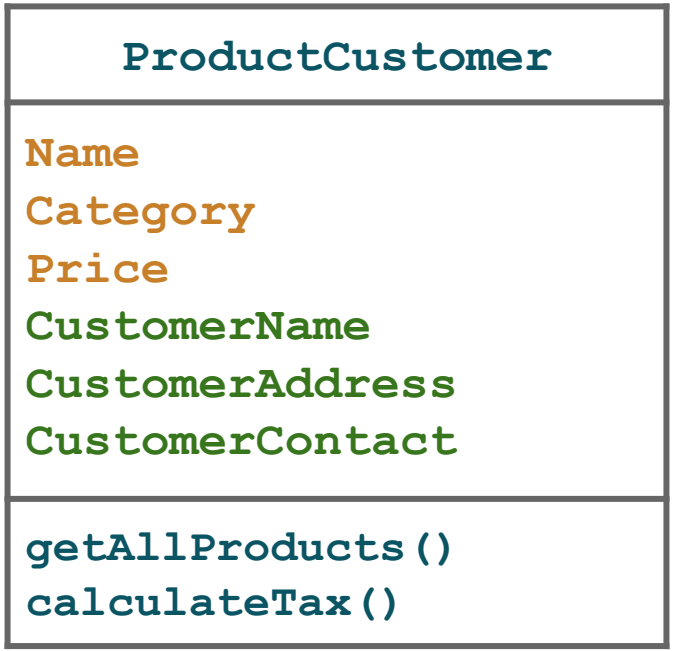
How can we do it ? with current knowledge of OOP.

Problem Scenario: Possible Solution

We can create a single class that holds the information of both product and customer.

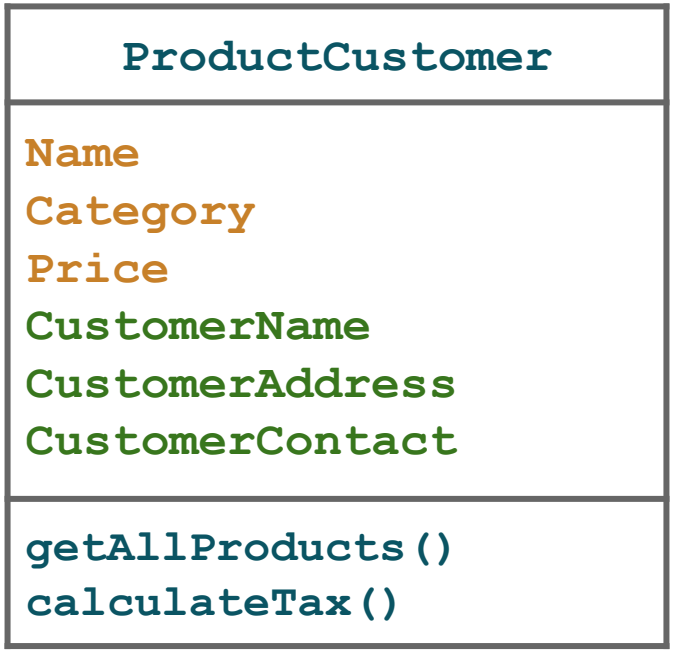
Single Class

The data can be combined into single class as follows:



Single Class: Any Issues?

Can you identify the problems with this approach.



Single Class: Any Issues?

If a customer bought multiple products, we have to create the multiple objects of the class

Object 1	Object 2	Object 3
Name: Apple Category: Fruits Price: 60 CustomerName: Hassan CustomerAddress: Lhr CustomerContact: 1101	Name: Eggs Category: Grocery Price: 200 CustomerName: Hassan CustomerAddress: Lhr CustomerContact: 1101	Name: Bread Category: Grocery Price: 70 CustomerName: Hassan CustomerAddress: Lhr CustomerContact: 1101
getAllProducts() calculateTax()	getAllProducts() calculateTax()	getAllProducts() calculateTax()

Single Class: Any Issues?

The problem with this approach is that the customer data is repeating for each product that he has bought.

Object 1	Object 2	Object 3
Name: Apple Category: Fruits Price: 60 CustomerName: Hassan CustomerAddress: Lhr CustomerContact: 1101	Name: Eggs Category: Grocery Price: 200 CustomerName: Hassan CustomerAddress: Lhr CustomerContact: 1101	Name: Bread Category: Grocery Price: 70 CustomerName: Hassan CustomerAddress: Lhr CustomerContact: 1101
getAllProducts() calculateTax()	getAllProducts() calculateTax()	getAllProducts() calculateTax()

Single Class: Any Issues?

Also, how to implement the getAllProducts behaviour to find the all products of a customer that he has purchased ?

Object 1	Object 2	Object 3
Name: Apple Category: Fruits Price: 60 CustomerName: Hassan CustomerAddress: Lhr CustomerContact: 1101	Name: Eggs Category: Grocery Price: 200 CustomerName: Hassan CustomerAddress: Lhr CustomerContact: 1101	Name: Bread Category: Grocery Price: 70 CustomerName: Hassan CustomerAddress: Lhr CustomerContact: 1101
getAllProducts() calculateTax()	getAllProducts() calculateTax()	getAllProducts() calculateTax()

Multiple Classes

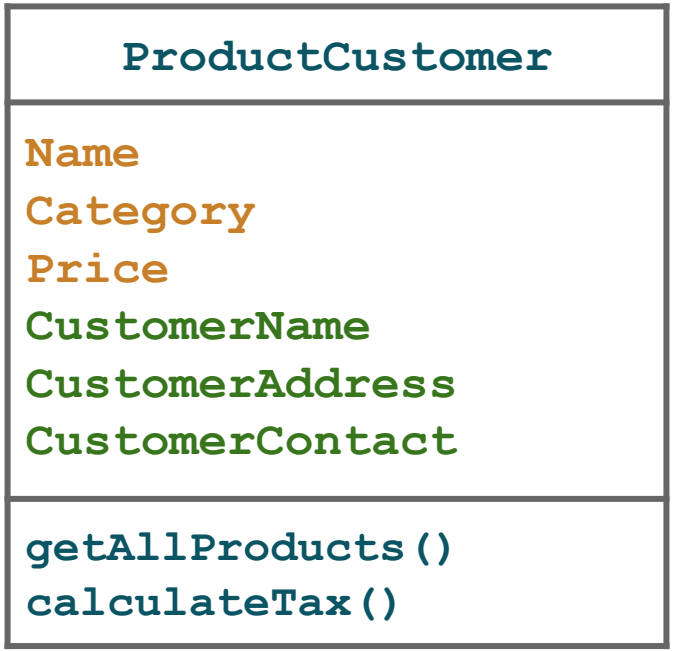
OOP Paradigm recommends to create multiple classes for each real world concept with its own attributes and behaviours those operate on these attributes.

Multiple Classes

For example, in this case, both product and customer are separate concepts in real world. Therefore, when we realize them in computational world we should have separate classes for them.

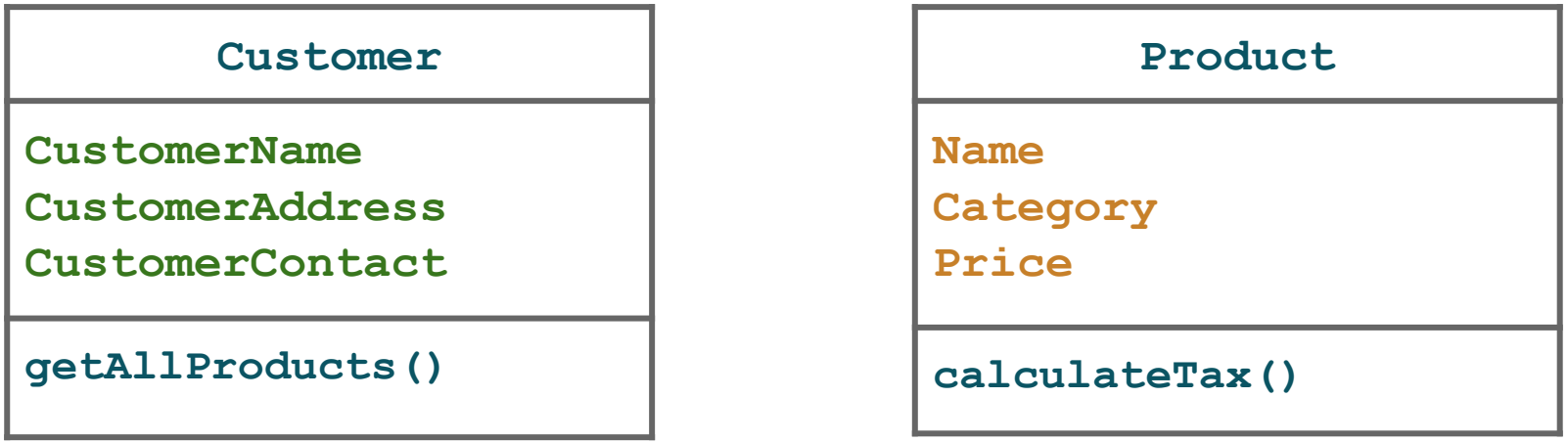
Multiple Classes

So instead of having single class we divide it into two classes.



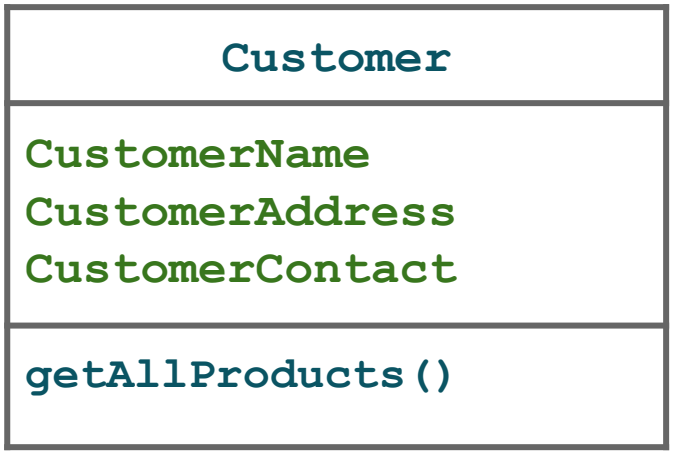
Multiple Classes

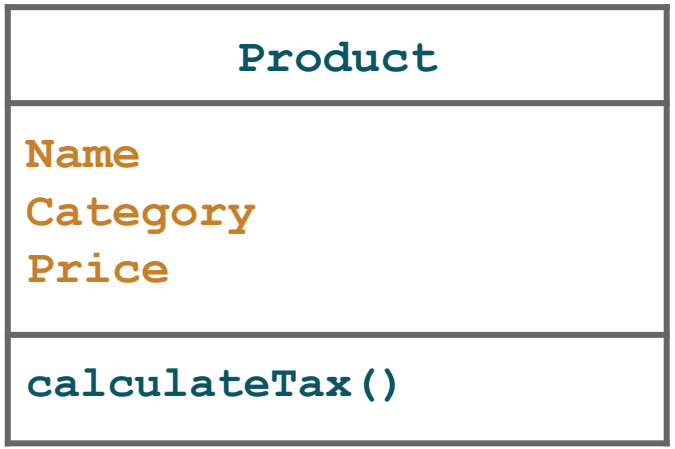
So instead of having single class we divide it into two classes.



Multiple Classes

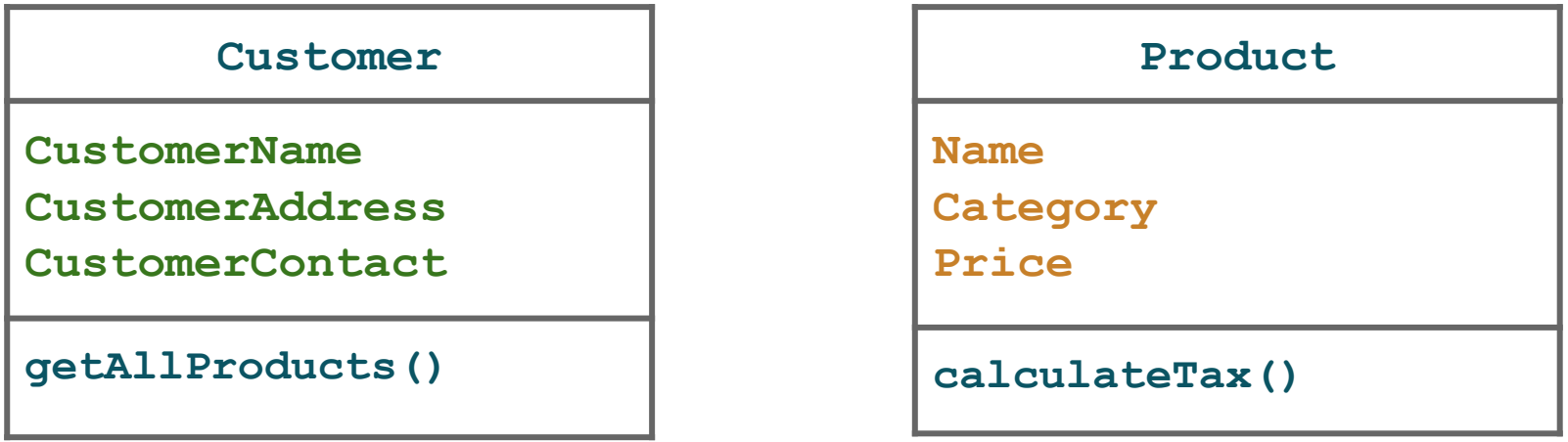
One problem is Still there, how to implement getAllProducts that a customer has purchased ?





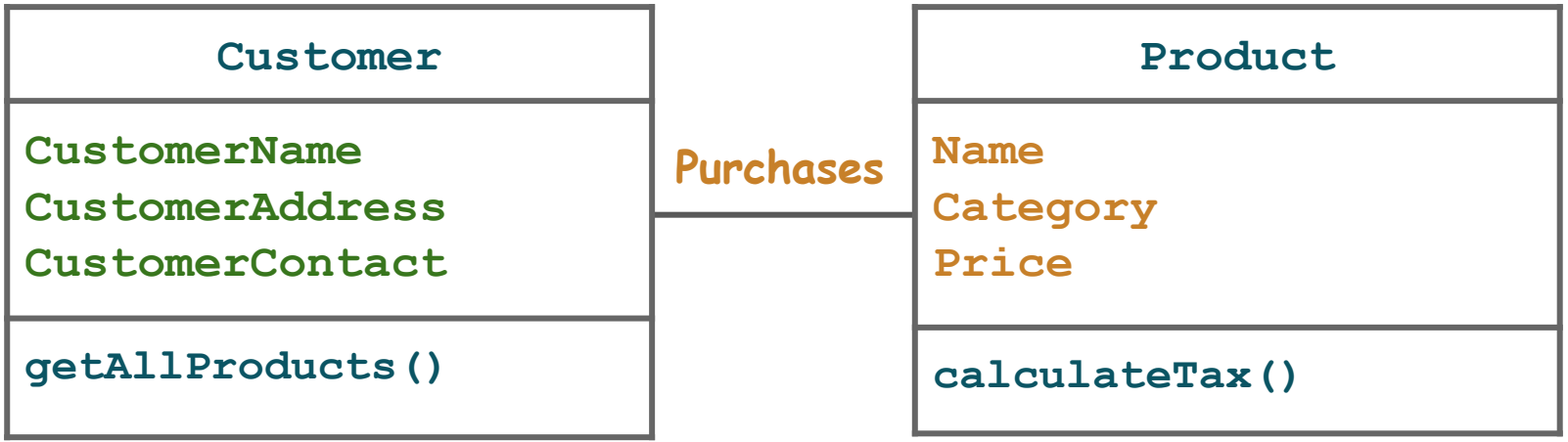
Multiple Classes

To implement the getAllProducts, Customer class should collaborate with the Product class.



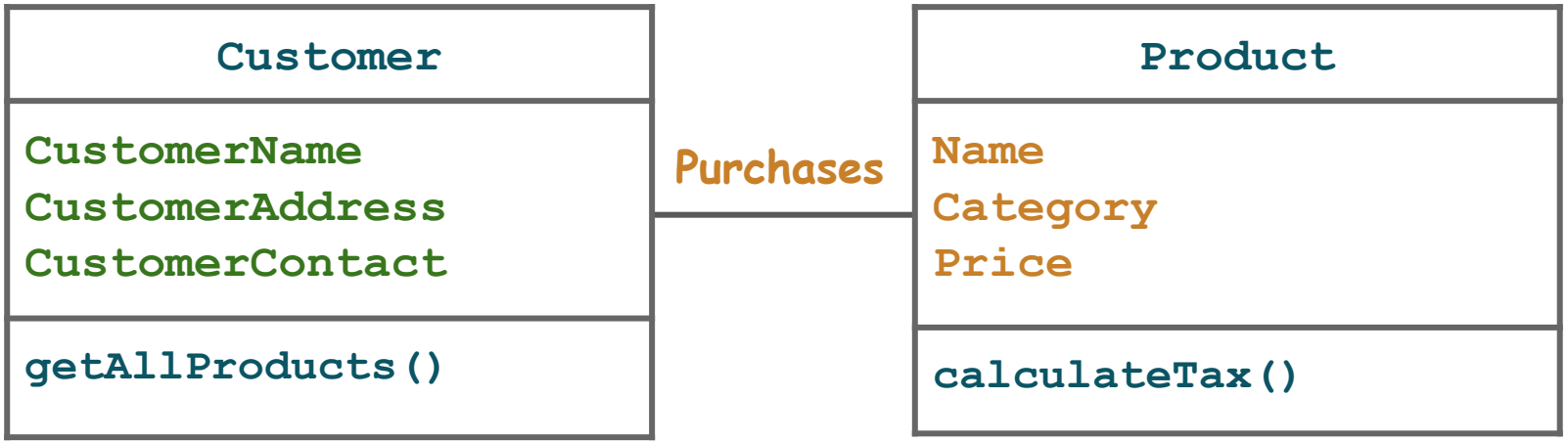
Multiple Classes

This collaboration can be shown in the CRC card as follows



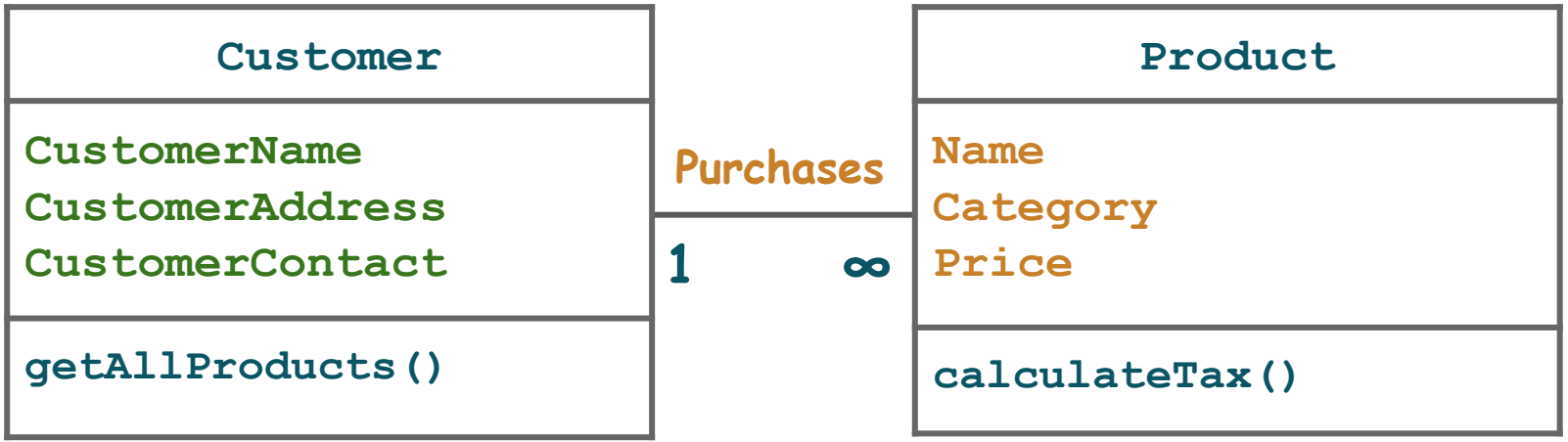
Multiple Classes

We should also mention how many instances of a class can participate in the relation.



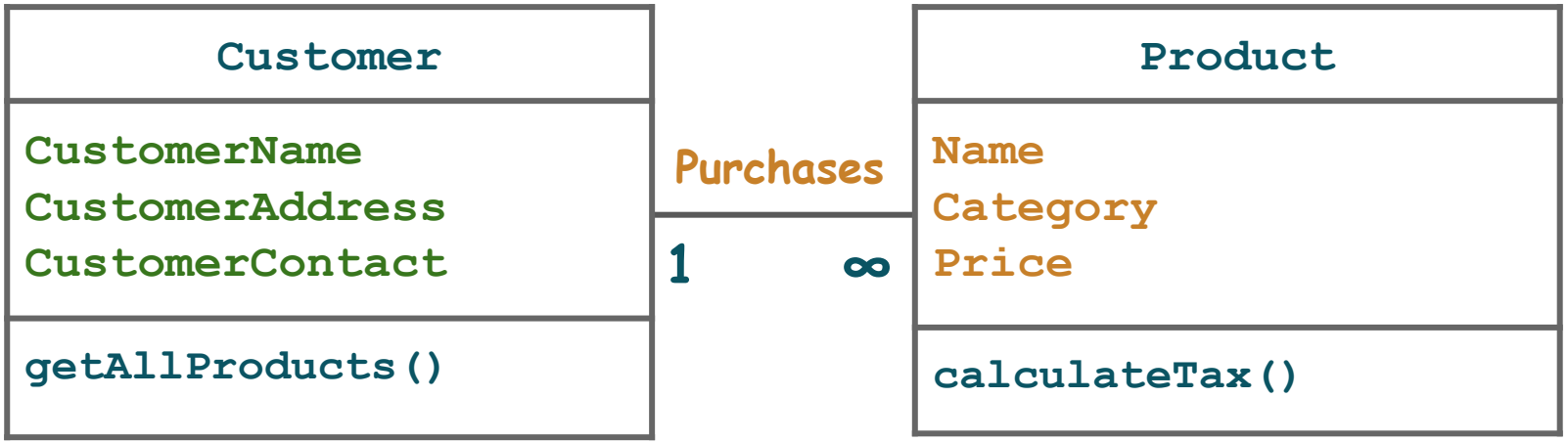
Multiple Classes

Here in this case one customer can purchase many products.



Multiplicity of Association

The information of number of instances appear in a relation is also called the Multiplicity of the Relation



Multiple Classes

This collaboration can be realized in code as follows

```
class Customer
{
     public string CustomerName;
     public string CustomerAddress;
     public string CustomerContact;
     public List<Product> products = new List<Product>();
     public List<Product> getAllProducts()
     {
          return products;
     }
     public addProduct(Product p)
     {
          products.Add(p);
     }
}
```

```
class Product
{
     public string name;
     public string category;
     public int price;
     public float calculateTax()
     {
          // Implementation
     }
}
```

Multiple Classes

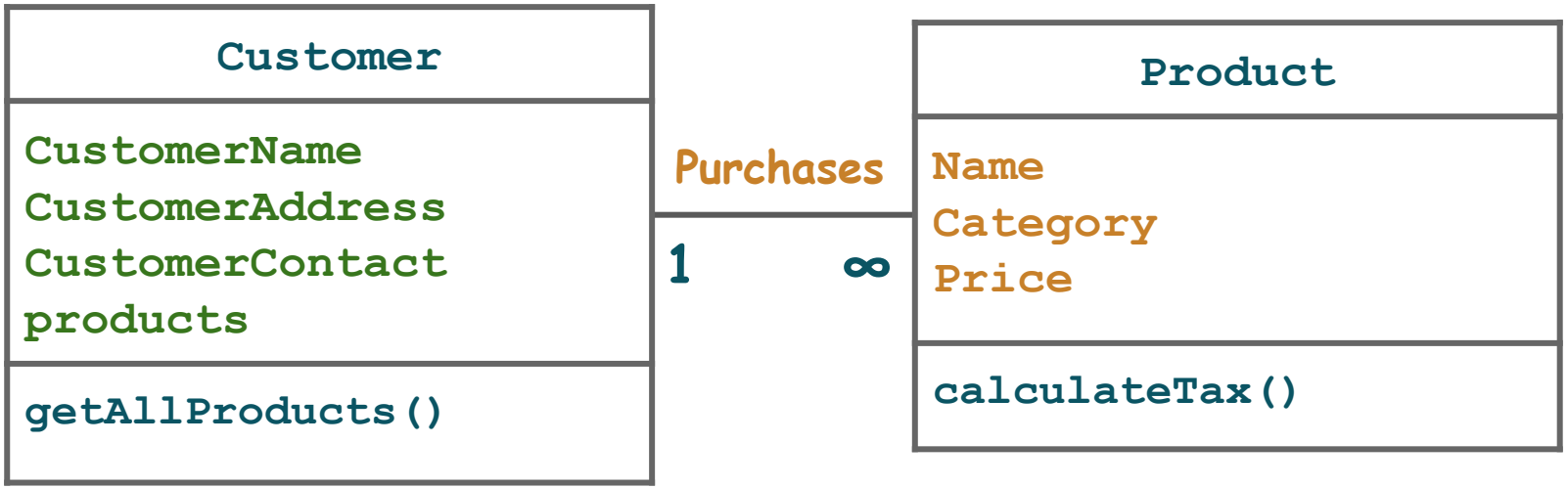
Products is the list of product objects that has been bought by the customer.

```
class Customer
{
      public string CustomerName;
      public string CustomerAddress;
      public string CustomerContact;
      public List<Product> products = new List<Product>();
      public List<Product> getAllProducts()
      {
          return products;
      }
      public addProduct(Product p)
      {
          products.Add(p);
      }
}
```

```
class Product
{
      public string name;
      public string category;
      public int price;
      public float calculateTax()
      {
          // Implementation
      }
}
```

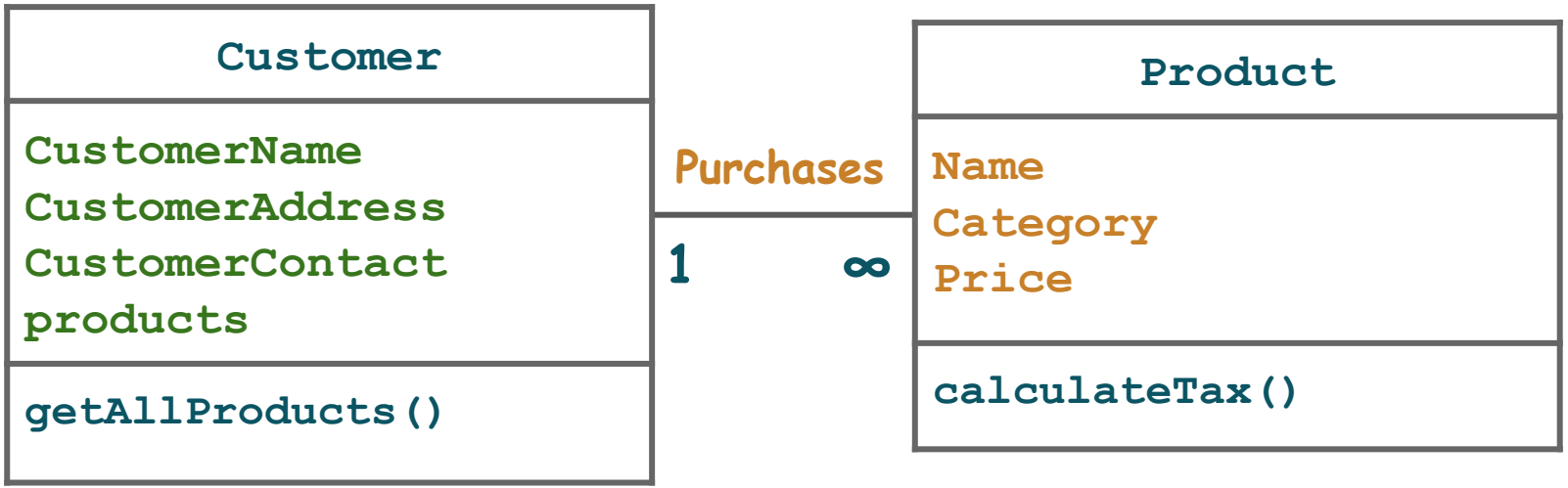
Multiple Classes

We can present the code using following diagram.



Domain Model

The graphical way to represent classes, attributes, operations, relation (collaboration) and Multiplicity between classes is called Class Diagram or Domain Model.



Conclusion

- OOP Paradigm recommends to create multiple classes for each real world concept with its own attributes and behaviours those operate on these attributes.
- The information of number of instances appear in a relation is called the Multiplicity of the Relation
- The graphical way to represent classes, attributes, operations, relation (collaboration) and Multiplicity between classes is called Class Diagram or Domain Model.

Learning Objective

Identify multiple classes and Association among these classes.

Self Assessment

1. Identify the classes within the following case study.

Academic branch offers different programs within different departments each program has a degree title and duration of degree.
Student Apply for admission in University and provides his/her name, age, FSC, and Ecat Marks and selects any number of preferences among the available programs.
Admission department prepares a merit list according to the highest merit and available seats and registers selected students in the program.
Academic Branch also add subjects for each program. A subject have subject code, credit hours, subjectType. A Program cannot have more than 20 Credit hour subjects.A Student Registers multiple subjects but he/she can not take more than 9 credit hours.
Fee department generate fees according to registered subjects of the students.