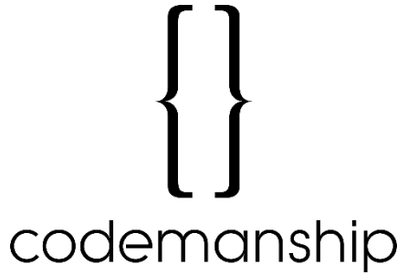


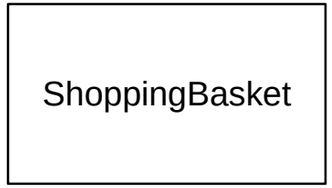
Class Diagrams

UML

Contents

- Classes
- Attributes
- Operations
- Associations
- Dependencies
- Inheritance
- Abstract Classes
- Interfaces
- Packages

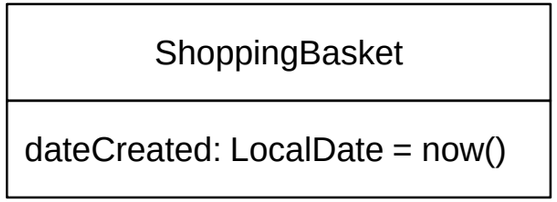
Classes




```
public class ShoppingBasket {

}
```

Attributes

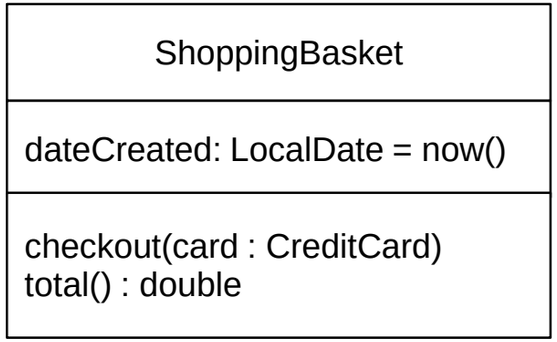
Also known as “fields”, “member variables”




```
public class ShoppingBasket {

    private final LocalDate dateCreated = now();

}
```

Operations

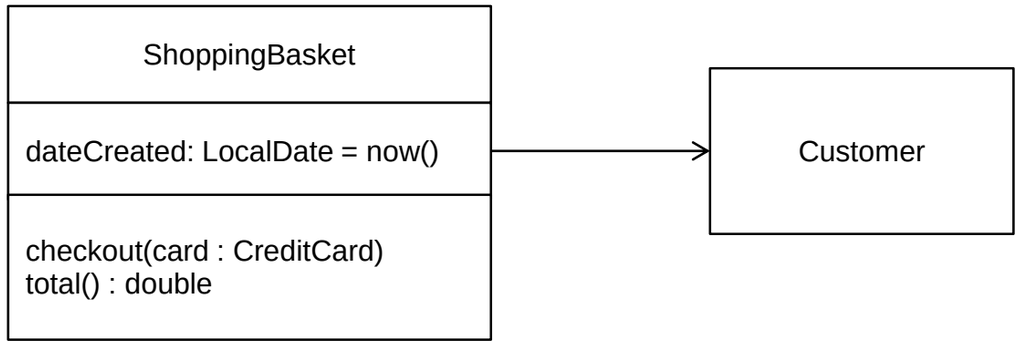
Also known as “methods”, “member functions”




```
public class ShoppingBasket {

    private final LocalDate dateCreated = now();

    public void checkout(CreditCard card) {

    }

    public double total(){
        return 0.0;
    }
}
```

Associations

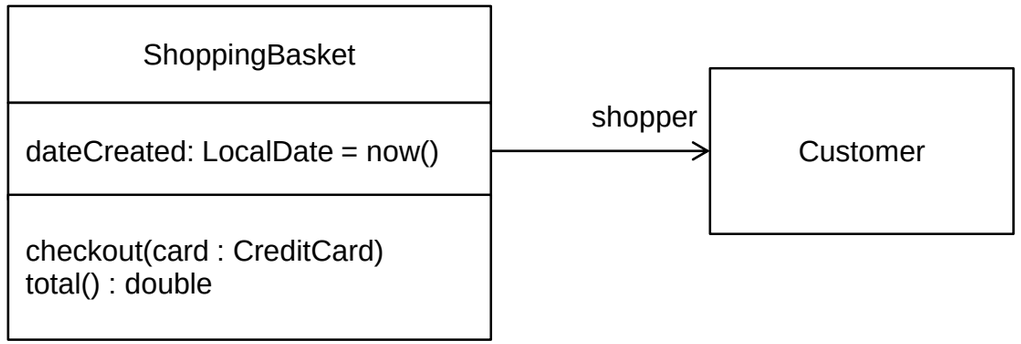
Associations are instance-level dependencies
(i.e., visible to all instance operations)




```
public class ShoppingBasket {

    private final LocalDate dateCreated = now();
    private final Customer customer;

    public ShoppingBasket(Customer customer) {
        this.customer = customer;
    }

    public void checkout(CreditCard card) {

    }

    public double total(){
        return 0.0;
    }
}
```

Role Names

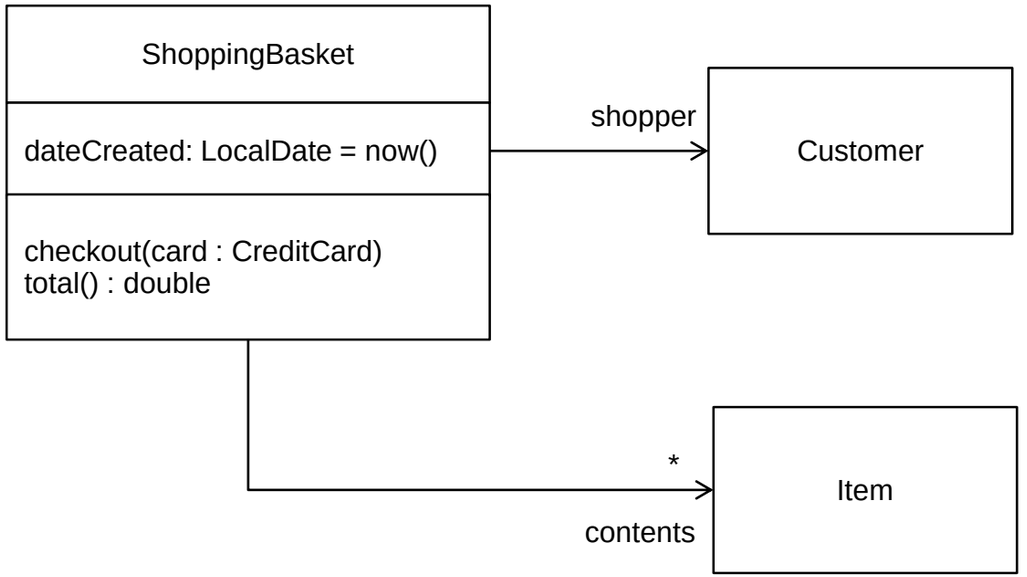
By default, the role name is the name of the class starting with a lowercase letter. But we can change the default like so:




```
public class ShoppingBasket {

    private final LocalDate dateCreated = now();
    private final Customer shopper;

    public ShoppingBasket(Customer shopper) {
        this.shopper = shopper;
    }

    public void checkout(CreditCard card) {

    }

    public double total(){
        return 0.0;
    }
}
```

Multiplicity

By default, the multiplicity of a role is 1, but we can change that…



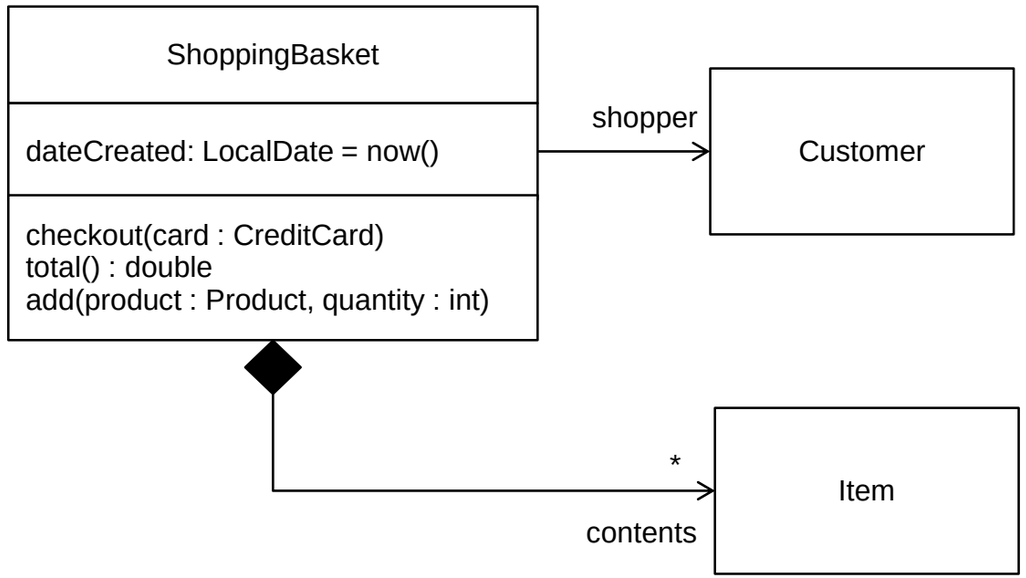

```
public class ShoppingBasket {

    private final LocalDate dateCreated = now();
    private final Customer shopper;
    private List<Item> contents;

    public ShoppingBasket(Customer shopper) {
        this.shopper = shopper;
        this.contents = new ArrayList<>();
    }

    public void checkout(CreditCard card) {

    }

    public double total(){
        return 0.0;
    }
}
```

Multiplicity Ranges

Multiplicity	Means
1	Exactly one (cannot be null)
0..1	Can be null
0..*	Many (including zero)
*	Many (including zero)
1..*	One or more
1..4	Between 1 and 4
1..4, 6..10	Between 1 and 4 or 6 and 10

Composition

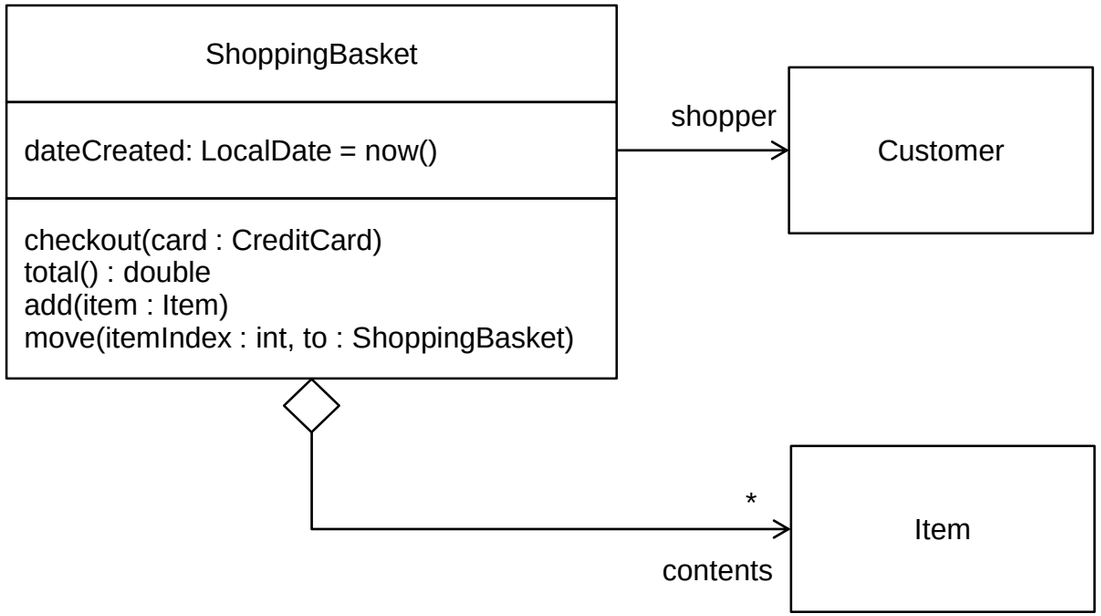
If contained objects can only ever be part of one containing object (e.g., basket items can only be part of one basket)…




```
public class ShoppingBasket {

    private final LocalDate dateCreated = now();
    private final Customer shopper;
    private List<Item> contents;

    public ShoppingBasket(Customer shopper) {
        this.shopper = shopper;
        this.contents = new ArrayList<>();
    }

    public void checkout(CreditCard card) {

    }

    public double total(){
        return 0.0;
    }

    public void add(Product product, int quantity){
        contents.add(new Item(product, quantity));
    }
}
```

Aggregation

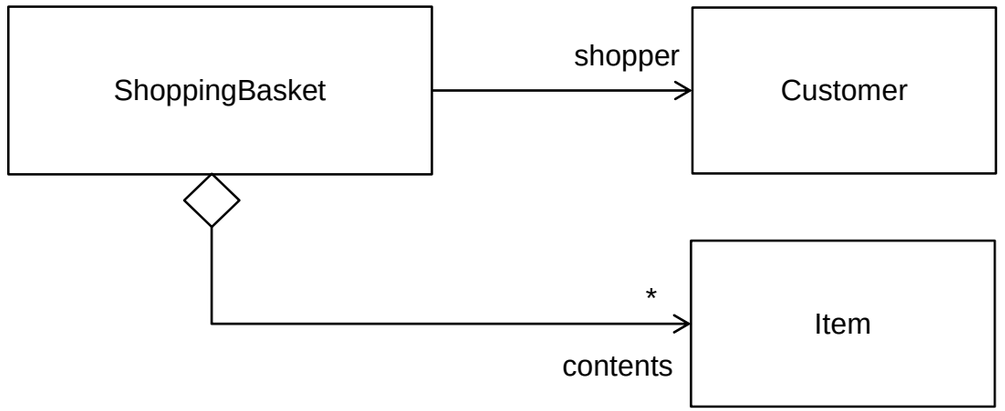
If contained objects can be moved between containers or exist outside of them…




```
public class ShoppingBasket {

    private final LocalDate dateCreated = now();
    private final Customer shopper;
    private List<Item> contents;

    public ShoppingBasket(Customer shopper) {
        this.shopper = shopper;
        this.contents = new ArrayList<>();
    }

    public void checkout(CreditCard card) {

    }

    public double total(){
        return 0.0;
    }

    public void add(Item item){
        contents.add(item);
    }

    public void move(int itemIndex, ShoppingBasket to){
        Item item = contents.remove(itemIndex);
        to.add(item);
    }
}
```

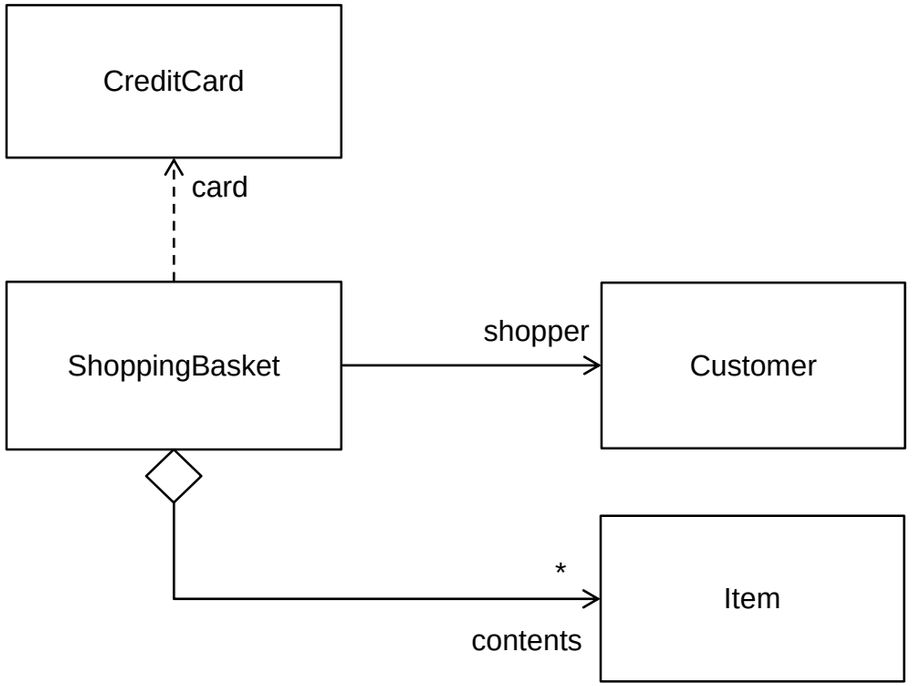
What To Show?

You choose what to show in your diagrams

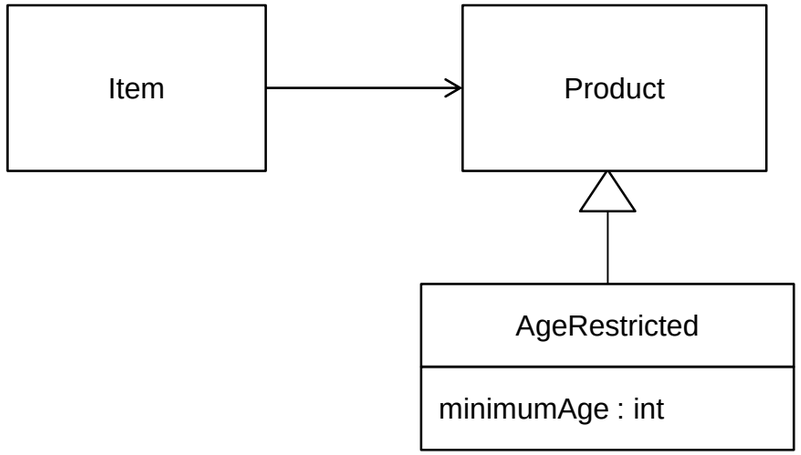


Dependencies

Some dependencies don't apply at the instance level, but to individual operations (e.g., CreditCard is only used by checkout())

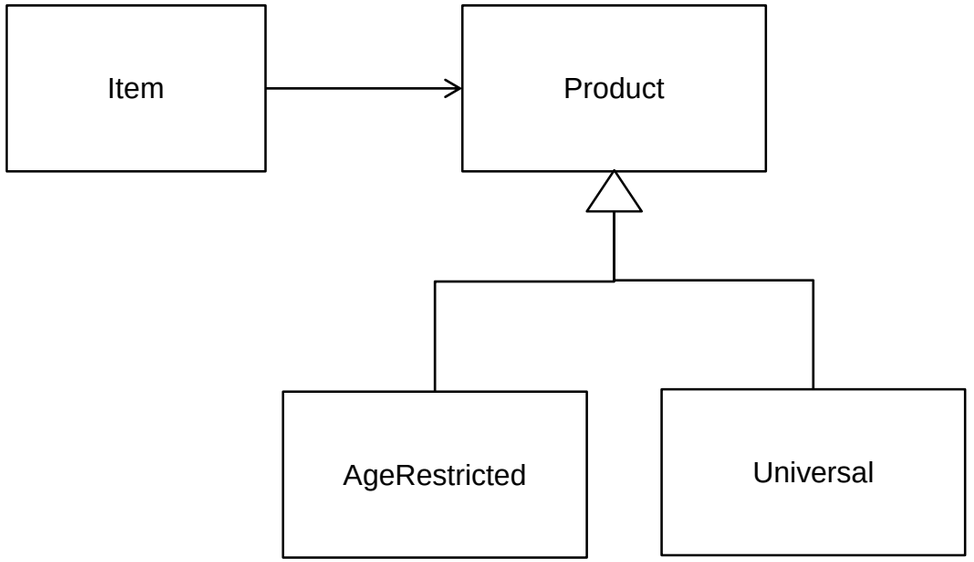


Inheritance

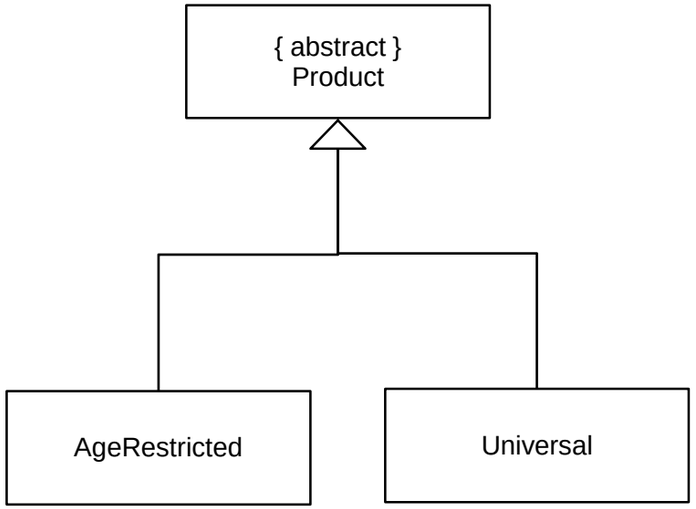


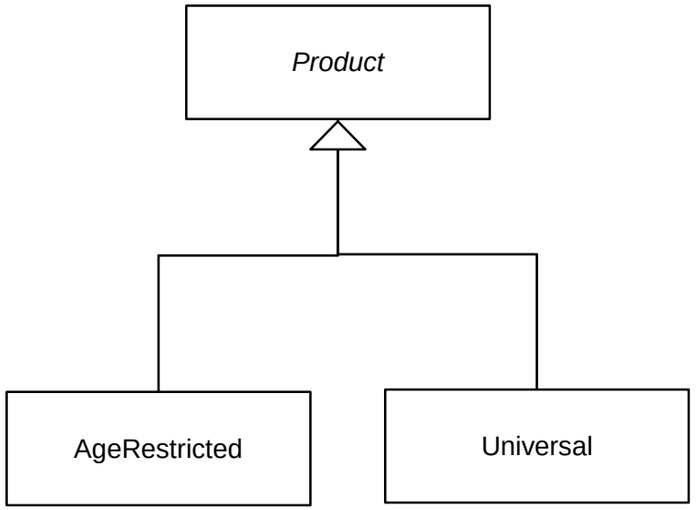

```
public class AgeRestricted extends Product {

}
```

Multiple Subclasses



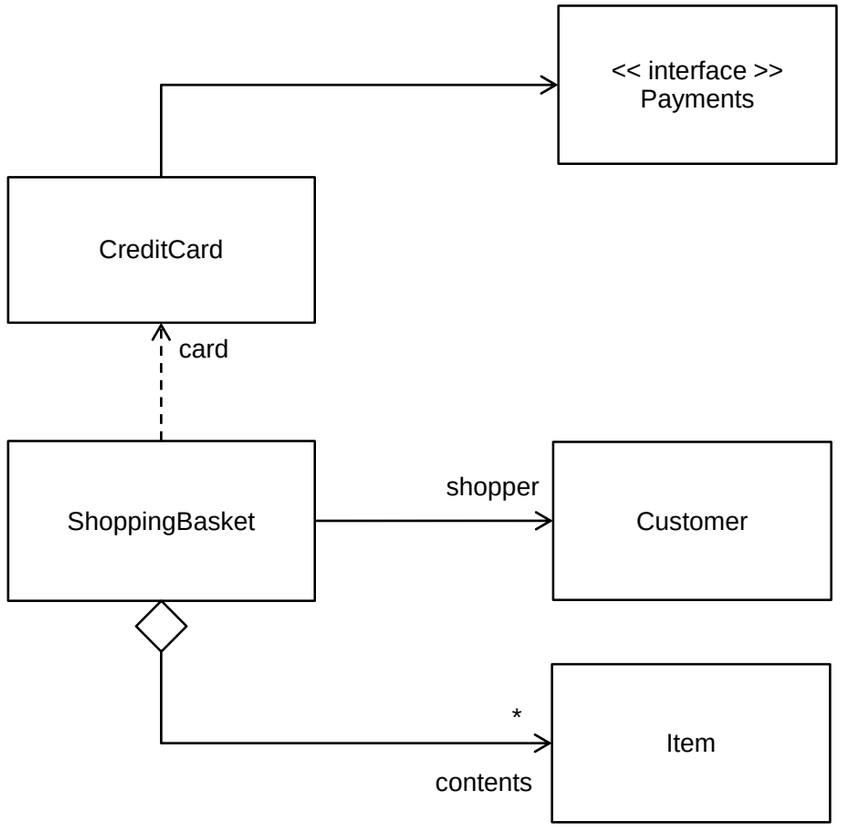
Abstract Classes






```
public abstract class Product {

}
```

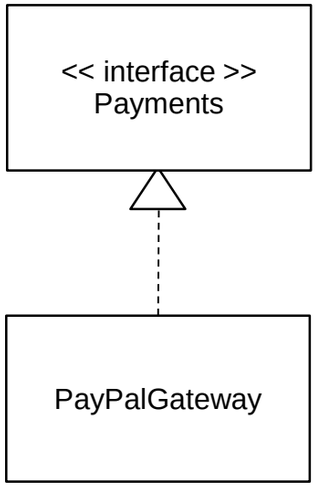
Interfaces




```
public class CreditCard {

   private final Payments payments;
   private final int longNumber;
   private final int expiryMonth;
   private final int expiryYear;
   private final int securityCode;

   public CreditCard(Payments payments,
                     int longNumber,
                     int expiryMonth,
                     int expiryYear,
                     int securityCode) {
      this.payments = payments;
      this.longNumber = longNumber;
      this.expiryMonth = expiryMonth;
      this.expiryYear = expiryYear;
      this.securityCode = securityCode;
   }

   public void pay(double amount){
      payments.process(amount, longNumber, expiryMonth, expiryYear, securityCode);
   }
}
```

```
public interface Payments {
   void process(double amount,
                int longNumber,
                int expiryMonth,
                int expiryYear,
                int securityCode);
}
```

Implementing Interfaces




```
public class PayPalGateway implements Payments {
    @Override
    public void process(double amount,
                        int longNumber,
                        int expiryMonth,
                        int expiryYear,
                        int securityCode) {

    }
}
```