Chapter 16

How to design an object-oriented program

Objectives

Applied

1. Design an object-oriented program and create a UML diagram for it.
2. Given the UML diagram for a program, develop the program with a three-tier architecture.

Objectives (cont.)

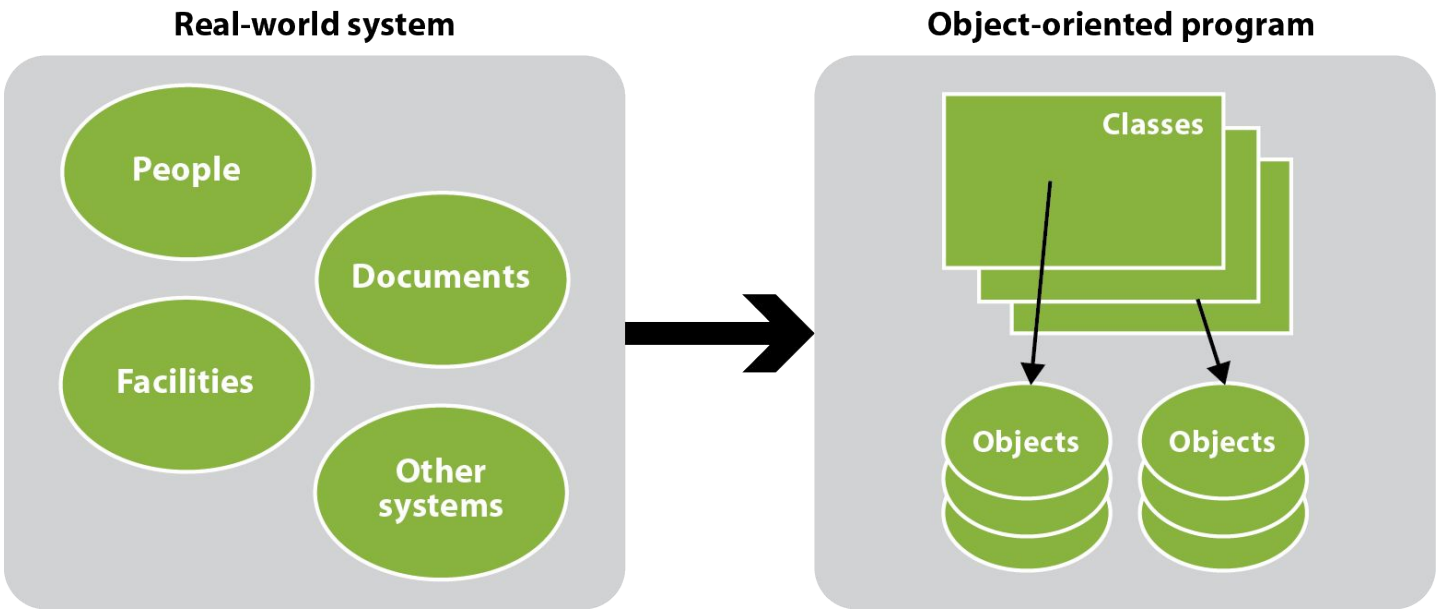
Knowledge

1. Describe each of these steps for designing the model for an object-oriented program:

   Identify the data attributes
   Subdivide each attribute into its smallest components
   Identify the classes
   Identify the methods
   Refine the classes, attributes, and methods

2. Describe the relationship between a class in an object-oriented program and an entity in the real world.
3. Distinguish between the presentation tier, the database tier, and the business tier in a three-tier architecture.

An object-oriented program is modeled after a real-world system



Five steps for designing an object-oriented program

Step 1: Identify the data attributes

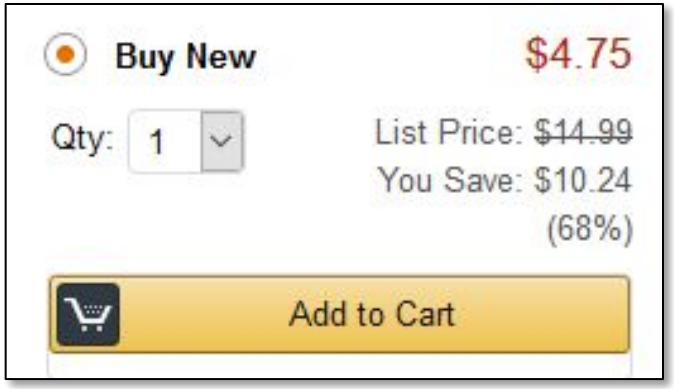
Step 2: Subdivide each attribute into its smallest useful components

Step 3: Identify the classes

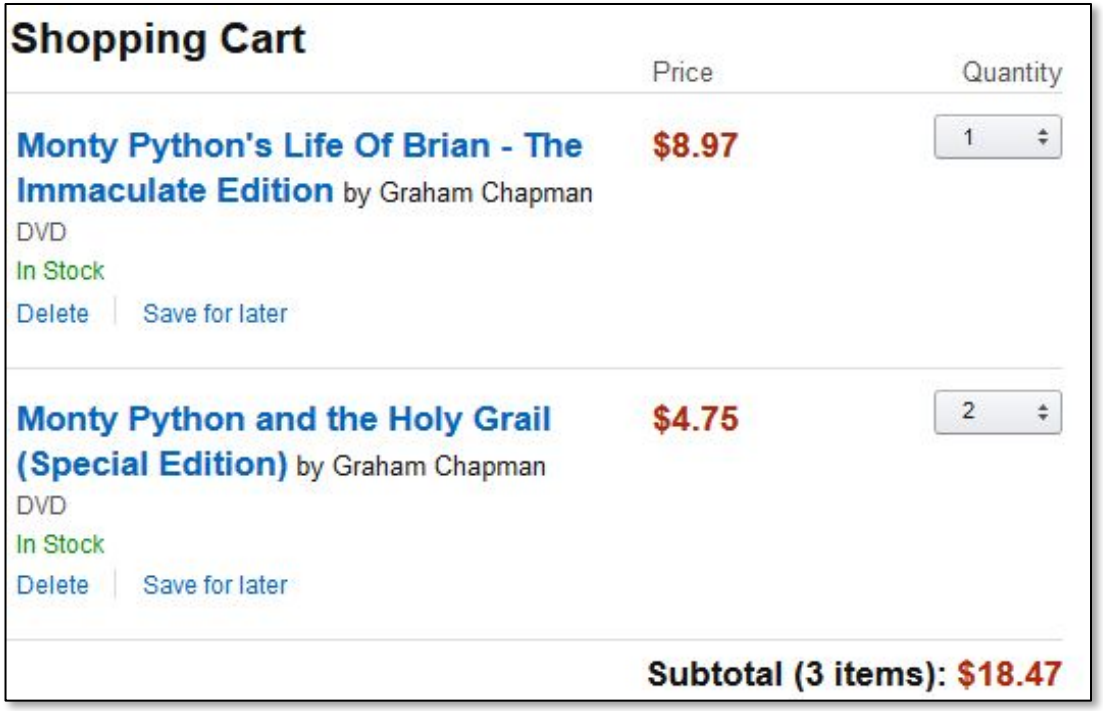
Step 4: Identify the methods

Step 5: Refine the classes, attributes, and methods

A screen capture that can be used to identify data attributes



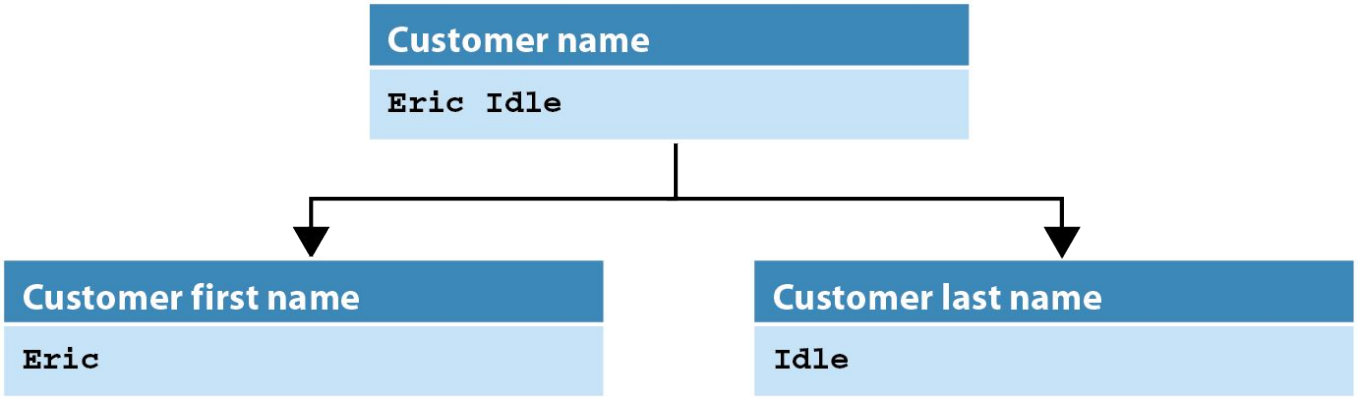
Another screen capture that can be used to identify data attributes



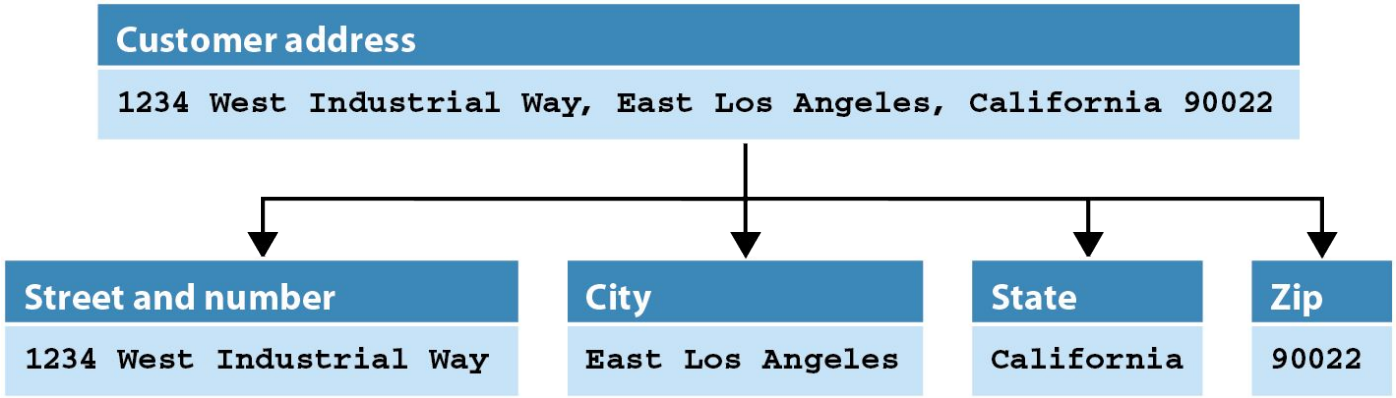
The data attributes identified from the screen captures

Product name	List price	Item count
Product creator	Discount percent	Cart total
Product format	Discount amount	Quantity
Stocking message	Discount price	

A customer name divided into first and last name



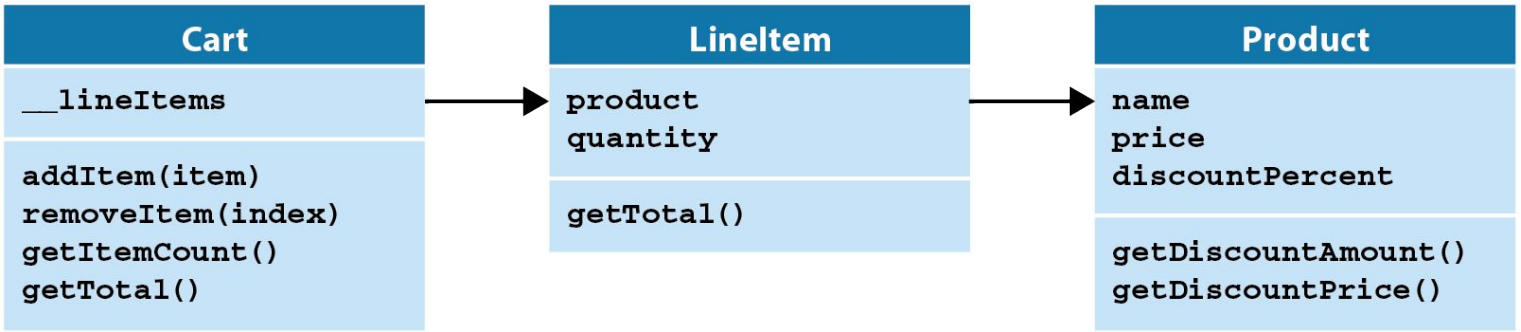
An address that's divided into its components



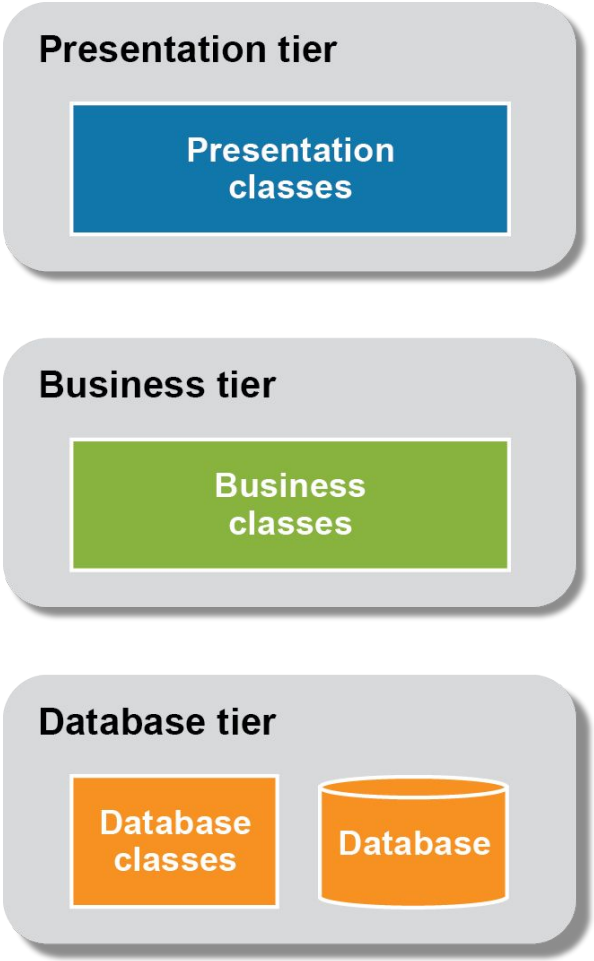
Possible classes and attributes for a Shopping Cart program

Product	LineItem	Cart
Product name*	Product name*	~~Item count~~
~~Product edition~~	Product discount price*	Cart total
~~Product creator~~	Quantity	
~~Product format~~	*Line item total*	
~~Product stocking message~~		
Product price		
Product discount percent		
Product discount amount		
Product discount price*		

The UML diagram for the classes of the Shopping Cart program



The three-tier architecture of an application



The business module

```
class Product:
    def __init__(self, name="", price=0.0, discountPercent=0):
        self.name = name
        self.price = price
        self.discountPercent = discountPercent

    def getDiscountAmount(self):
        discountAmount = self.price * self.discountPercent / 100
        return round(discountAmount, 2)

    def getDiscountPrice(self):
        discountPrice = self.price - self.getDiscountAmount()
        return round(discountPrice, 2)

class LineItem:
    def __init__(self, product=None, quantity=1):
        self.product = product
        self.quantity = quantity

    def getTotal(self):
        total = self.product.getDiscountPrice() * self.quantity
        return total
```

The business module (cont.)

```
class Cart:
    def __init__(self):
        self.__lineItems = []

    def addItem(self, item):
        self.__lineItems.append(item)

    def removeItem(self, index):
        self.__lineItems.pop(index)

    def getTotal(self):
        total = 0.0
        for item in self.__lineItems:
            total += item.getTotal()
        return total

    def getItemCount(self):
        return len(self.__lineItems)
```

The business module (cont.)

```
def __iter__(self):
    self.__index = -1
    return self

def __next__(self):
    if self.__index == len(self.__lineItems)-1:
        raise StopIteration
    self.__index += 1
    lineItem = self.__lineItems[self.__index]
    return lineItem
```

The products.csv file

```
The Holy Grail (DVD),4.75,30
Life of Brian (DVD),8.97,20
The Meaning of Life (DVD),6.50,15
```

The db module

```
import csv
from business import Product

FILENAME = "products.csv"

def get_products():
    products = []
    with open(FILENAME, newline="") as file:
        reader = csv.reader(file)
        for row in reader:
            # convert row to Product object
            product = Product(row[0], float(row[1]), int(row[2]))
            products.append(product)
    return products
```

Code that tests the database and business layers

```
import db
from business import Product, LineItem, Cart

products = db.get_products()
product = products[1]
lineItem = LineItem(product, 2)
cart = Cart()
cart.addItem(lineItem)
print("Product:  ", product.name)
print("Price:    ", product.getDiscountPrice())
print("Quantity: ", lineItem.quantity)
print("Total:    ", cart.getTotal())
```

The console

```
Product:   Life of Brian (DVD)
Price:     7.18
Quantity:  2
Total:     14.36
```

The user interface

```
The Shopping Cart program

COMMAND MENU
cart - Show the cart
add  - Add an item to the cart
del  - Delete an item from cart
exit - Exit program

PRODUCTS
Item  Name                          Price   Discount   Your Price
1     The Holy Grail (DVD)           4.75        30%         3.32
2     Life of Brian (DVD)            8.97        20%         7.18
3     The Meaning of Life (DVD)      6.50        15%         5.53

Command: add
Item number: 1
Quantity: 2
Item 1 was added.

Command: add
Item number: 3
Quantity: 1
Item 2 was added.
```

The user interface (cont.)

```
Command: cart
Item  Name                          Your Price   Quantity      Total
1     The Holy Grail (DVD)               3.32          2       6.64
2     The Meaning of Life (DVD)          5.53          1       5.53
                                                            12.17

Command: del
Item number: 1
Item 1 was deleted.

Command: cart
Item  Name                          Your Price   Quantity      Total
1     The Meaning of Life (DVD)          5.53          1       5.53
                                                             5.53

Command: exit
Bye!
```

The shopping_cart module

```
import db
from business import Product, LineItem, Cart

def show_title():
    print("The Shopping Cart program")
    print()

def show_menu():
    print("COMMAND MENU")
    print("cart - Show the cart")
    print("add  - Add an item to the cart")
    print("del  - Delete an item from cart")
    print("exit - Exit program")
    print()
```

The shopping_cart module (cont.)

```
def show_products(products):
    print("PRODUCTS")
    line_format1 = "{:<5s} {:<25s} {:>10s} {:>10s} {:>12s}"
    line_format2 = "{:<5d} {:<25s} {:>10.2f} {:>10s} {:>12.2f}"
    print(line_format1.format("Item", "Name", "Price",
                              "Discount", "Your Price"))
    for i in range(len(products)):
        product = products[i]
        print(line_format2.format(i+1,
              product.name,
              product.price,
              str(product.discountPercent) + "%",
              product.getDiscountPrice()))
    print()
```

The shopping_cart module (cont.)

```
def show_cart(cart):
    if cart.getItemCount() == 0:
        print("There are no items in your cart.\n")
    else:
        # items = cart.lineItems
        line1 = "{:<5s} {:<25s} {:>12s} {:>10s} {:>10s}"
        line2 = "{:<5d} {:<25s} {:>12.2f} {:>10d} {:>10.2f}"
        print(line1.format("Item", "Name", "Your Price",
                                   "Quantity", "Total"))
        i = 0
        for item in cart:
            print(line2.format(i+1,
                  item.product.name,
                  item.product.getDiscountPrice(),
                  item.quantity,
                  item.getTotal()))
            i += 1
        print("{:>66.2f}".format(cart.getTotal()))
        print()
```

The shopping_cart module (cont.)

```
def add_item(cart, products):
    number = int(input("Item number: "))
    quantity = int(input("Quantity: "))
    if number < 1 or number > len(products):
        print("No product has that number.\n")
    else:
        # Get Product object, store in LineItem object,
        # and add to Cart object
        product = products[number-1]
        item = LineItem(product, quantity)
        cart.addItem(item)
        print("Item " + str(cart.getItemCount()) +
              " was added.\n")

def remove_item(cart):
    number = int(input("Item number: "))
    if number < 1 or number > cart.getItemCount():
        print("The cart does not contain an item " +
              "with that number.\n")
    else:
        # Remove LineItem object at specified index from cart
        cart.removeItem(number-1)
        print("Item " + str(number) + " was deleted.\n")
```

The shopping_cart module (cont.)

```
def main():
    show_title()
    show_menu()

    # get a list of Product objects and display them
    products = db.get_products()
    show_products(products)

    # create a Cart object to store LineItem objects
    cart = Cart()
    while True:
        command = input("Command: ")
        if command == "cart":
            show_cart(cart)
        elif command == "add":
            add_item(cart, products)
        elif command == "del":
            remove_item(cart)
        elif command == "exit":
            print("Bye!")
            break
        else:
            print("Not a valid command. Please try again.\n")
```

The shopping_cart module (cont.)

```
if __name__ == "__main__":
    main()
```