实物类面向对象设计

文泰来 老师

扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: ninechapter
知乎专栏： http://zhuanlan.zhihu.com/jiuzhang
微博: http://www.weibo.com/ninechapter
官网: www.jiuzhang.com

Copyright © www.jiuzhang.com

课程大纲

- 实物类OOD题型
- 实物类OOD解题思路
- Vending machine
- Coffee maker
- Kindle

Copyright © www.jiuzhang.com

实物类OOD题型

- Vending machine
- Jukebox
- CD Player
- Coffee maker
- Kindle

Copyright © www.jiuzhang.com

实物类OOD题型

- 频率：中高

Copyright © www.jiuzhang.com

实物类OOD题型

- 频率：中高
- 难度：中低

Copyright © www.jiuzhang.com

实物类OOD解题技巧

- 考虑对于实物的输入输出

Copyright © www.jiuzhang.com

实物类OOD解题技巧

- 考虑对于实物的输入输出

例子：Coffee maker

Copyright © www.jiuzhang.com

实物类OOD解题技巧

- 考虑对于实物的输入输出

例子：Coffee maker

CofferMaker

Copyright © www.jiuzhang.com

实物类OOD解题技巧

- 考虑对于实物的输入输出

例子：Coffee maker

Copyright © www.jiuzhang.com

实物类OOD解题技巧

- Design pattern的运用

Copyright © www.jiuzhang.com

实物类OOD解题技巧

- Design pattern的运用

State pattern

Decorate pattern

Factory pattern

Copyright © www.jiuzhang.com

Vending Machine

- Can you design a vending machine?

Copyright © www.jiuzhang.com

Clarify

- What
- How
- Who

Copyright © www.jiuzhang.com

Clarify

- What

关键字： Vending machine

Copyright © www.jiuzhang.com

Clarify

- What

关键字： Vending machine

Copyright © www.jiuzhang.com

Clarify

- What

关键字： Vending machine

Copyright © www.jiuzhang.com

Clarify

- What

关键字： Vending machine, Coin, Item

Copyright © www.jiuzhang.com

Clarify

- 关键字： Vending machine

Copyright © www.jiuzhang.com

Clarify

- 关键字： Vending machine

厂家，重量，颜色…

Copyright © www.jiuzhang.com

Clarify

- 关键字： Vending machine

厂家，重量，颜色…

VendingMachine
- String manufacture
+ String getManufacture()

Copyright © www.jiuzhang.com

Clarify

- 关键字： Vending machine

大小： Vending machine的大小是否有限制？

Copyright © www.jiuzhang.com

Clarify

- 关键字： Item

Copyright © www.jiuzhang.com

Clarify

- 关键字： Item

What items does this vending machine sell?

Copyright © www.jiuzhang.com

Clarify

- 关键字： Item

What items does this vending machine sell?

Naïve design approach: each item matches a class

Copyright © www.jiuzhang.com

Clarify

- 关键字： Item

What to do when an item sold out?

Copyright © www.jiuzhang.com

Clarify

- 关键字： Item

What to do when an item sold out?

Design: Might need to support refill use case

Copyright © www.jiuzhang.com

Clarify

- 关键字： Payment

Copyright © www.jiuzhang.com

Clarify

- 关键字： Payment

What are the supported payment methods?

Copyright © www.jiuzhang.com

Clarify

- Payment

- Coin

- Paper money

- Credit card

Copyright © www.jiuzhang.com

Clarify

- Payment

- Coin/Paper money : 知道当前收了多少钱，找零

- Credit card: 直接当前Item的价格

Copyright © www.jiuzhang.com

Clarify

- 对于本题：

- 假设Vending machine的大小没有限制
- 假设目前只卖三种产品：Coke, Sprite和Mountain Dew
- 假设目前只接受硬币

Copyright © www.jiuzhang.com

Clarify

- How

Copyright © www.jiuzhang.com

Clarify

- How

Copyright © www.jiuzhang.com

Clarify

- How to select item to purchase?

Design: selectItem(?)

Copyright © www.jiuzhang.com

Clarify

- 对于本题：

- 假设输入一个input代表一种Item (e.g. A1 -> Coke)

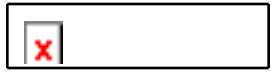
Copyright © www.jiuzhang.com

Who

- N/A

Copyright © www.jiuzhang.com

Core Object



VendingMachine

Copyright © www.jiuzhang.com

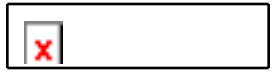
Core Object

Coin

VendingMachine

Copyright © www.jiuzhang.com

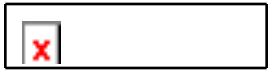
Core Object



Coin	VendingMachine	Coke

Copyright © www.jiuzhang.com

Core Object



Sprite

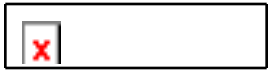
Coin

VendingMachine

Coke

Copyright © www.jiuzhang.com

Core Object



Sprite

Coin

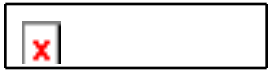
VendingMachine

Coke

MountainDew

Copyright © www.jiuzhang.com

Core Object



Sprite

Coin

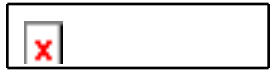
VendingMachine

Coke

MountainDew

Copyright © www.jiuzhang.com

Core Object



Sprite

Coin

VendingMachine

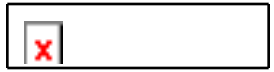
Item

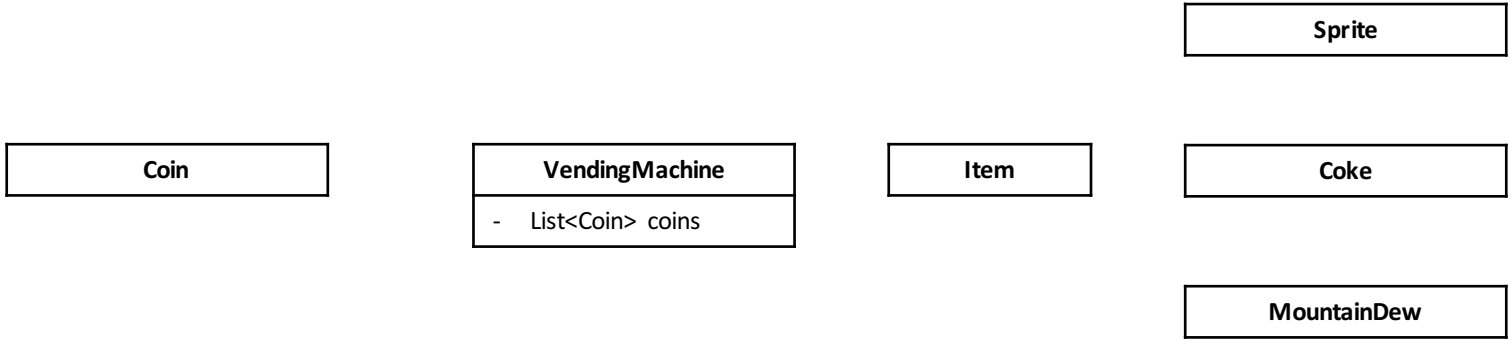
Coke

MountainDew

Copyright © www.jiuzhang.com

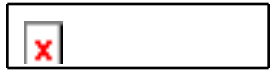
Core Object

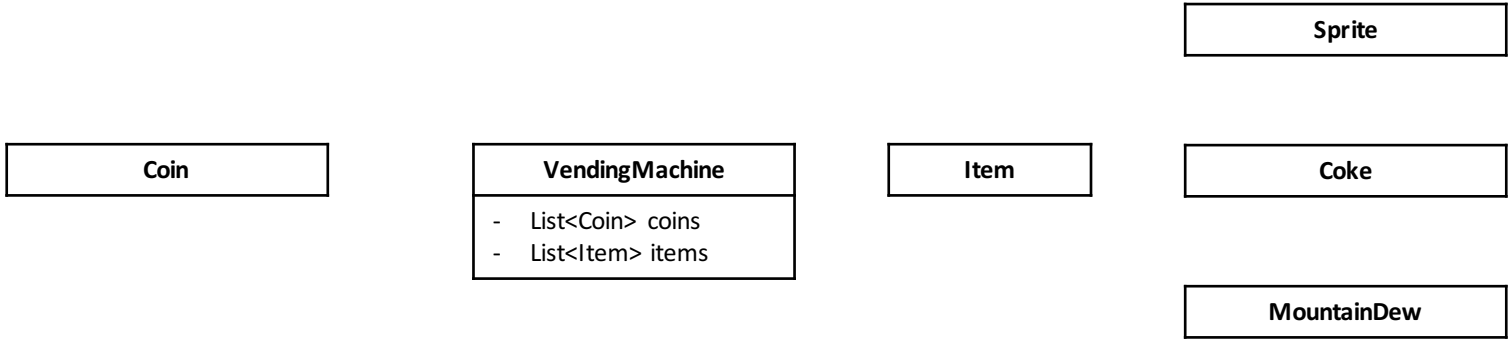




Copyright © www.jiuzhang.com

Core Object





Copyright © www.jiuzhang.com

Use cases

Vending machine

Copyright © www.jiuzhang.com

Use cases

Vending machine:

- Select item

Copyright © www.jiuzhang.com

Use cases

Vending machine:

- Select item
- Insert coin

Copyright © www.jiuzhang.com

Use cases

Vending machine:

- Select item
- Insert coin
- Execute transaction

Copyright © www.jiuzhang.com

Use cases

Vending machine:

- Select item
- Insert coin
- Execute transaction
- Cancel transaction

Copyright © www.jiuzhang.com

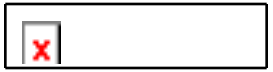
Use cases

Vending machine:

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- List<Item> items

Coin

Item

Sprite

Coke

MountainDew

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

Classes

Use case: Select item

- Vending machine takes an external input, shows the price of that item

Copyright © www.jiuzhang.com

Classes

VendingMachine

- List<Coin> coins
- List<Item> items

Coin

Item

Sprite

Coke

MountainDew

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

Classes

VendingMachine

- List<Coin> coins
- List<Item> items

+ float selectItem(String selection)

Coin

Item

Sprite

Coke

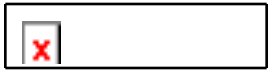
MountainDew

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- List<Item> items
- Map<String, Item> itemIdentifiers

+ float selectItem(String selection)

Coin

Item

Sprite

Coke

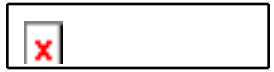
MountainDew

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

Challenge

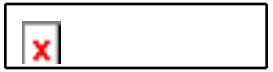


"A1" ->

Vending machine:

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- List<Item> items
- Map<String, Item> itemIdentifiers

+ float selectItem(String selection)

Coin

ItemInfo

Item

Sprite

Coke

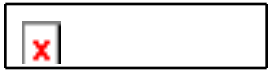
MountainDew

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- List<Item> items
- Map<String, Item> itemIdentifiers

+ float selectItem(String selection)

Coin

ItemInfo

- Float price

+ float getPrice()

Item

Sprite

Coke

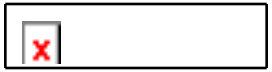
MountainDew

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- Map<ItemInfo, List<Item>> items
- Map<String, Item> itemIdentifiers

+ float selectItem(String selection)

Coin

ItemInfo

- Float price

+ float getPrice()

Item

Sprite

Coke

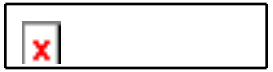
MountainDew

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- Map<ItemInfo, List<Item>> items
- Map<String, ItemInfo> itemIdentifiers

+ float selectItem(String selection)

Coin

ItemInfo

- Float price

+ float getPrice()

Item

Sprite

Coke

MountainDew

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

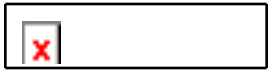
Classes

Use case: Insert coin

- Insert a list of coins into vending machine

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- Map<ItemInfo, List<Item>> items
- Map<String, ItemInfo> itemIdentifiers

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)

Coin

ItemInfo

- Float price

+ float getPrice()

Item

Sprite

Coke

MountainDew

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

Classes

Use case: Execute transaction

- Get the current selected item
- Compare the item price and inserted coins
- If not enough money, throw an exception
- Else, return the item purchased
- Refund if any

Copyright © www.jiuzhang.com

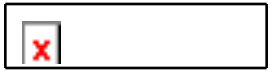
Classes

Use case: Execute transaction

- Get the current selected item

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- Map<ItemInfo, List<Item>> items
- Map<String, ItemInfo> itemIdentifiers
- ItemInfo currentSelection

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)

Coin

ItemInfo

- Float price

+ float getPrice()

Item

Sprite

Coke

MountainDew

Use cases

Select item

Insert coin

Execute transaction

Cancel transaction

Refill items

Copyright © www.jiuzhang.com

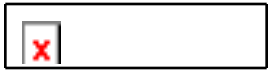
Classes

Use case: Execute transaction

- Get the current selected item
- Compare the item price and inserted coins

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- Map<ItemInfo, List<Item>> items
- Map<String, ItemInfo> itemIdentifiers
- ItemInfo currentSelection
- List<Coin> currentCoins

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)

Coin

ItemInfo

- Float price

+ float getPrice()

Item

Sprite

Coke

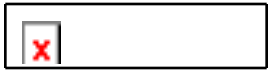
MountainDew

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- Map<ItemInfo, List<Item>> items
- Map<String, ItemInfo> itemIdentifiers
- ItemInfo currentSelection
- List<Coin> currentCoins

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)

<<enumeration>>
Coin

PENNY(1)
NICKLE(5)
DIME(10)
QUARTER(25)

- Float value

ItemInfo

- Float price

+ float getPrice()

Item

Sprite

Coke

MountainDew

Use cases

Select item
Insert coin
Execute transaction
Cancel transaction
Refill items

Copyright © www.jiuzhang.com

Classes

Use case: Execute transaction

- Get the current selected item
- Compare the item price and inserted coins
- If not enough money, throw an exception

Copyright © www.jiuzhang.com

Classes

VendingMachine

- List<Coin> coins
- Map<ItemInfo, List<Item>> items
- Map<String, ItemInfo> itemIdentifiers
- ItemInfo currentSelection
- List<Coin> currentCoins

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)

<<enumeration>>
Coin

PENNY(1)
NICKLE(5)
DIME(10)
QUARTER(25)

- Float value

ItemInfo

- Float price

+ float getPrice()

Item

Sprite

Coke

MountainDew

NotEnoughMoneyException

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

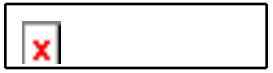
Classes

Use case: Execute transaction

- Get the current selected item
- Compare the item price and inserted coins
- If not enough money, throw an exception
- Else, return the item purchased

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- Map<ItemInfo, List<Item>> items
- Map<String, ItemInfo> itemIdentifiers
- ItemInfo currentSelection
- List<Coin> currentCoins

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()

<<enumeration>>
Coin

PENNY(1)
NICKLE(5)
DIME(10)
QUARTER(25)

- Float value

ItemInfo

- Float price

+ float getPrice()

Item

Sprite

Coke

MountainDew

NotEnoughMoneyException

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

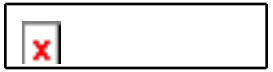
Classes

Use case: Execute transaction

- Get the current selected item
- Compare the item price and inserted coins
- If not enough money, throw an exception
- Else, return the item purchased
- Refund if any

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- Map<ItemInfo, List<Item>> items
- Map<String, ItemInfo> itemIdentifiers
- ItemInfo currentSelection
- List<Coin> currentCoins

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
- List<Coin> refund()

<<enumeration>>
Coin

PENNY(1)
NICKLE(5)
DIME(10)
QUARTER(25)

- Float value

ItemInfo

- Float price

+ float getPrice()

Item

Sprite

Coke

MountainDew

NotEnoughMoneyException

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

Coin change

- http://www.cnblogs.com/grandyang/p/4840713.html

Copyright © www.jiuzhang.com

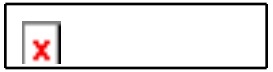
Classes

Use case: Cancel transaction

- Return the current coins that has been inserted

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- Map<ItemInfo, List<Item>> items
- Map<String, ItemInfo> itemIdentifiers
- ItemInfo currentSelection
- List<Coin> currentCoins

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()
- List<Coin> refund()

<<enumeration>>
Coin

PENNY(1)
NICKLE(5)
DIME(10)
QUARTER(25)

- Float value

ItemInfo

- Float price

+ float getPrice()

Item

Sprite

Coke

MountainDew

NotEnoughMoneyException

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

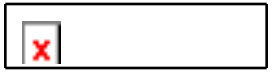
Classes

Use case: Refill items

- Refill items on top of current stock

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- Map<ItemInfo, List<Item>> items
- Map<String, ItemInfo> itemIdentifiers
- ItemInfo currentSelection
- List<Coin> currentCoins

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()
+ void refillItems(List<Item> items)
- List<Coin> refund()

<<enumeration>>
Coin

PENNY(1)
NICKLE(5)
DIME(10)
QUARTER(25)

- Float value

ItemInfo

- Float price

+ float getPrice()

Item

- ItemInfo info

Sprite

Coke

MountainDew

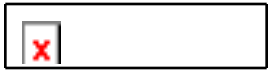
NotEnoughMoneyException

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- Map<String, ItemInfo> itemIdentifiers
- ItemInfo currentSelection
- List<Coin> currentCoins
- Map<ItemInfo, List<Item>> stock

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()
+ void refillItems(List<Item> items)
- List<Coin> refund()

<<enumeration>>
Coin

PENNY(1)
NICKLE(5)
DIME(10)
QUARTER(25)

- Float value

ItemInfo

- Float price

+ float getPrice()

Item

- ItemInfo info

Sprite

Coke

MountainDew

NotEnoughMoneyException

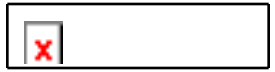
NotEnoughItemException

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

Classes - Final view



VendingMachine

- List<Coin> coins
- Map<String, ItemInfo> itemIdentifiers
- ItemInfo currentSelection
- List<Coin> currentCoins
- Map<ItemInfo, List<Item>> stock

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()
+ void refillItems(List<Item> items)
- List<Coin> refund()

<<enumeration>>
Coin

PENNY(1)
NICKLE(5)
DIME(10)
QUARTER(25)

- Float value

ItemInfo

- Float price

+ float getPrice()

Item

- ItemInfo info

Sprite

Coke

MountainDew

NotEnoughMoneyException

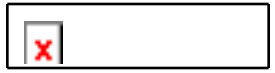
NotEnoughItemException

Use cases

Select item
Insert coin
Execute transaction
Cancel transaction
Refill items

Copyright © www.jiuzhang.com

Good practice

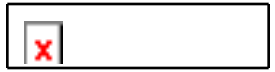


```
stock = new HashMap<ItemInfo, List<Item>>();
```

```
public void refillItem(List<Item> items)
{
    for(Item item : items)
    {
        ItemInfo info = item.getInfo();
        List<Item> itemsInStock = stock.get(info);
        itemsInStock.add(item);
        stock.put(info, itemsInStock);
    }
}
```

Copyright © www.jiuzhang.com

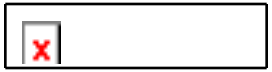
Good practice



```
class Stock
{
    private HashMap<ItemInfo, List<Item>> stock;

    public void add(Item item)
    {
        ItemInfo info = item.getInfo();
        List<Item> itemsInStock = stock.get(info);
        itemsInStock.add(item);
        stock.put(info, itemsInStock);
    }
}

stock = new Stock();

public void refillItem(List<Item> items)
{
    for(Item item : items)
    {
        stock.add(item);
    }
}
```

Copyright © www.jiuzhang.com

Good practice



VendingMachine

- List<Coin> coins
- List<Item> items
- Map<String, ItemInfo> itemIdentifiers
- ItemInfo currentSelection
- List<Coin> currentCoins
- Map<ItemInfo, List<Item>> stock

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()
+ void refillItems(List<Item> items)
- List<Coin> refund()

<<enumeration>>
Coin

PENNY(1)
NICKLE(5)
DIME(10)
QUARTER(25)

- Float value

ItemInfo

- Float price

+ float getPrice()

Item

- ItemInfo info

Sprite

Coke

MountainDew

NotEnoughMoneyException

NotEnoughItemException

Stock

- Map<ItemInfo, List<Item>> stock

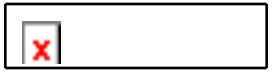
+ int getQuantity(ItemInfo info)
+ void add(Item t)
+ void deduct(ItemInfo info)

Use cases

Select item
Insert coin
Execute transaction
Cancel transaction
Refill items

Copyright © www.jiuzhang.com

Good practice



VendingMachine

- List<Coin> coins
- List<Item> items
- Map<String, ItemInfo> itemIdentifiers
- ItemInfo currentSelection
- List<Coin> currentCoins
- Stock stock

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()
+ void refillItems(List<Item> items)
- List<Coin> refund()

<<enumeration>>
Coin

PENNY(1)
NICKLE(5)
DIME(10)
QUARTER(25)

- Float value

ItemInfo

- Float price

+ float getPrice()

Item

- ItemInfo info

Sprite

Coke

MountainDew

NotEnoughMoneyException

NotEnoughItemException

Stock

- Map<ItemInfo, List<Item>> stock

+ int getQuantity(ItemInfo info)
+ void add(Item t)
+ void deduct(ItemInfo info)

Use cases

Select item
Insert coin
Execute transaction
Cancel transaction
Refill items

Copyright © www.jiuzhang.com

Challenge

- For these use cases:

- Select item
- Insert coin
- Execute transaction
- Cancel transaction

Copyright © www.jiuzhang.com

Challenge

- For these use cases:

- Select item
- Insert coin
- Execute transaction
- Cancel transaction

What will happen if some item has been selected?

Copyright © www.jiuzhang.com

Challenge

- For these use cases:

- Select item : throws a selections has already been made
- Insert coin
- Execute transaction
- Cancel transaction

What will happen if some item has been selected?

Copyright © www.jiuzhang.com

Challenge

- For these use cases:

- Select item : throws a selections has already been made
- Insert coin : update current inserted value
- Execute transaction
- Cancel transaction

What will happen if some item has been selected?

Copyright © www.jiuzhang.com

Challenge

- For these use cases:

- Select item : throws a selections has already been made
- Insert coin : update current inserted value
- Execute transaction : Get selected item if money is enough
- Cancel transaction

What will happen if some item has been selected?

Copyright © www.jiuzhang.com

Challenge

- For these use cases:

- Select item : throws a selections has already been made
- Insert coin : update current inserted value
- Execute transaction : Get selected item if money is enough
- Cancel transaction : return money and empty selected item

What will happen if some item has been selected?

Copyright © www.jiuzhang.com

Challenge

- For these use cases:

- Select item
- Insert coin
- Execute transaction
- Cancel transaction

What will happen if none item has been selected?

Copyright © www.jiuzhang.com

Challenge

- For these use cases:

- Select item : make a selection
- Insert coin
- Execute transaction
- Cancel transaction

What will happen if none item has been selected?

Copyright © www.jiuzhang.com

Challenge

- For these use cases:

- Select item : make a selection
- Insert coin : throws to ask user make a selection first
- Execute transaction
- Cancel transaction

What will happen if none item has been selected?

Copyright © www.jiuzhang.com

Challenge

- For these use cases:

- Select item : make a selection
- Insert coin : throws to ask user make a selection first
- Execute transaction : throws to ask user to make a selection first
- Cancel transaction

What will happen if none item has been selected?

Copyright © www.jiuzhang.com

Challenge

- For these use cases:

- Select item : make a selection
- Insert coin : throws to ask user make a selection first
- Execute transaction : throws to ask user to make a selection first
- Cancel transaction : maybe not doing anything or throw

What will happen if none item has been selected?

Copyright © www.jiuzhang.com

Challenge

- Insert coin

```
public void insertCoin(List<Coin> coins)
{
    if(selectedItem == null)
    {
        throw new Exception("You need to make a selection first");
    }
    else if(selectedItem != null)
    {
        currentCoins.add(coins);
    }
}
```

Copyright © www.jiuzhang.com

Challenge

- 我们刚刚考虑了HAS_SELECTION 和 NO_SELECTION 的情况

Copyright © www.jiuzhang.com

Challenge

- 我们刚刚考虑了HAS_SELECTION 和 NO_SELECTION 的情况

- 那么对于：

- COINS_INSERTED
- NO_COIN
- SOLD
- SOLD_OUT

应该怎么办？

Copyright © www.jiuzhang.com

Challenge

```
public void insertCoin(List<Coin> coins)
{
    if(selectedItem == null)
    {
        throw new Exception("You need to make a selection first");
    }
    else if(selectedItem != null)
    {
        currentCoins.add(coins);
    }
    else if(SOLD)
    {
        throw new Exception("Be patient, item is coming out, dont need to pay once more");
    }
    else if(SOLD_OUT)
    {
        throw new Exception("Item has been sold out, please dont pay for nothing");
    }
    ...
}
```

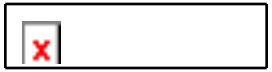
Copyright © www.jiuzhang.com

Challenge

- State Design Pattern

Copyright © www.jiuzhang.com

Challenge



- State Design Pattern

States:

- HAS_SELECTION
- NO_SELECTION
- COINS_INSERTED
- NO_COIN
- SOLD
- SOLD_OUT

Copyright © www.jiuzhang.com

Challenge

- State Design Pattern

State related actions:

- select item
- insert coin
- execute transaction
- cancel transaction

Copyright © www.jiuzhang.com

Classes

VendingMachine
- List<Coin> coins - List<Item> items - Map<String, ItemInfo> itemIdentifiers - ItemInfo currentSelection - List<Coin> currentCoins - Map<ItemInfo, List<Item>> stock
+ float selectItem(String selection) + void insertCoins(List<Coin> coins) + Item executeTransaction() + List<Coin> cancelTranscation() + void refillItems(List<Item> items) - List<Coin> refund()

Use cases
Select item
Insert coin
Execute transaction
Cancel transaction
Refill items

Copyright © www.jiuzhang.com

Classes

VendingMachine

- List<Coin> coins
- List<Item> items
- Map<String, ItemInfo> itemIdentifiers
- ItemInfo currentSelection
- List<Coin> currentCoins
- Map<ItemInfo, List<Item>> stock

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()
+ void refillItems(List<Item> items)
- List<Coin> refund()

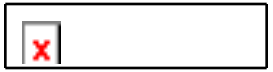
<<interface>>
State

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()

Use cases
Select item
Insert coin
Execute transaction
Cancel transaction
Refill items

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- List<Item> items
- Map<String, ItemInfo> itemIdentifiers
- ItemInfo currentSelection
- List<Coin> currentCoins
- Map<ItemInfo, List<Item>> stock

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()
+ void refillItems(List<Item> items)
- List<Coin> refund()

<<interface>>
State

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()

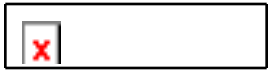
HasSelectionState

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()

Use cases
Select item
Insert coin
Execute transaction
Cancel transaction
Refill items

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- List<Item> items
- Map<String, ItemInfo> itemIdentifiers
- ItemInfo currentSelection
- List<Coin> currentCoins
- Map<ItemInfo, List<Item>> stock

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()
+ void refillItems(List<Item> items)
- List<Coin> refund()

<<interface>>
State

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()

HasSelectionState

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()

...

SoldOutState

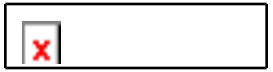
+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- List<Item> items
- Map<String, ItemInfo> itemIdentifiers
- ItemInfo currentSelection
- List<Coin> currentCoins
- Map<ItemInfo, List<Item>> stock
- State state

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()
+ void refillItems(List<Item> items)
- List<Coin> refund()
+ void setState(State s)

<<interface>>
State

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()

HasSelectionState

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()

...

SoldOutState

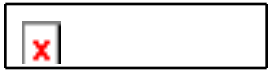
+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

Classes



VendingMachine

- List<Coin> coins
- List<Item> items
- Map<String, ItemInfo> itemIdentifiers
- ItemInfo currentSelection
- List<Coin> currentCoins
- Map<ItemInfo, List<Item>> stock
- State state
- HasSelectionState hasSelectionState
- …
- SoldOutState soldOutState

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()
+ void refillItems(List<Item> items)
- List<Coin> refund()
+ void setState(State s)

<<interface>>
State

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()

HasSelectionState

+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()

…

SoldOutState

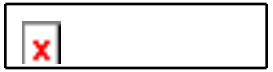
+ float selectItem(String selection)
+ void insertCoins(List<Coin> coins)
+ Item executeTransaction()
+ List<Coin> cancelTranscation()

Use cases

- Select item
- Insert coin
- Execute transaction
- Cancel transaction
- Refill items

Copyright © www.jiuzhang.com

Vending machine



```
public interface State {
    public void selectItem(String selection);
    public void insertMoney(int value);
    public void executeTransaction();
    public int cancelTransaction();
}
```

Copyright © www.jiuzhang.com

Vending machine

```java
public class VendingMachine {

    private AbstractState state;
    private NoSelectionState noSelectionState;
    private HasSelectionState hasSelectionState;
    private InsertedMoneyState insertedMoneyState;


    public VendingMachine()
    {

        noSelectionState = new NoSelectionState(this);
        hasSelectionState = new HasSelectionState(this);
        insertedMoneyState = new InsertedMoneyState(this);
        state = noSelectionState;

    }

    public void changeToNoSelectionState()
    {
        state = noSelectionState;
    }

    public void changeToHasSelectionState()
    {
        state = hasSelectionState;
    }

    public void changeToInsertedMoneyState()
    {
        state = insertedMoneyState;
    }

    public void selectItem(String selection)
    {
        state.selectItem(selection);
    }

    public void addMoney(int value)
    {
        state.insertMoney(value);
    }

    public void executeTransaction()
    {
        state.executeTransaction();
    }

    public int cancelTransaction()
    {
        return state.cancelTransaction();
    }
}
```

```java
public class NoSelectionState implements AbstractState{

    VendingMachine vendingMachine;

    public NoSelectionState(VendingMachine vendingMachine) {
        this.vendingMachine = vendingMachine;
    }

    @Override
    public void selectItem(String selection) {
        // TODO Auto-generated method stub
        vendingMachine.setSelectedItem(selection);
        vendingMachine.changeToHasSelectionState();
    }

    @Override
    public void insertMoney(int value) {
        // TODO Auto-generated method stub
        System.out.println("Please make a selection first");
    }

    @Override
    public void executeTransaction() {
        // TODO Auto-generated method stub
        System.out.println("Please make a selection first");
    }

    @Override
    public int cancelTransaction() {
        // TODO Auto-generated method stub
        System.out.println("Please make a selection first");
        return 0;
    }

}
```

Copyright © www.jiuzhang.com

Real life object

- 难度不大

- 从Input / Output 考虑
- 继承关系
- 考虑Exception
- Design pattern if possible

Copyright © www.jiuzhang.com

Coffee maker

Copyright © www.jiuzhang.com

Clarify

- What

关键字： Coffee maker

Copyright © www.jiuzhang.com

Clarify

- What

关键字： Coffee maker

Copyright © www.jiuzhang.com

Clarify

Copyright © www.jiuzhang.com

Clarify

- Input

Copyright © www.jiuzhang.com

Clarify

- Output

Copyright © www.jiuzhang.com

Clarify

- 对于本题

Input: Coffee packs

Output: Expresso

Copyright © www.jiuzhang.com

Clarify

- How

Copyright © www.jiuzhang.com

Clarify

- How

Copyright © www.jiuzhang.com

Clarify

- What are the functions that out coffee maker supports?

Copyright © www.jiuzhang.com

Clarify

- 对于本题：

- Brew

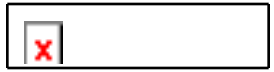
Copyright © www.jiuzhang.com

Core object

CoffeeMaker

Copyright © www.jiuzhang.com

Core object

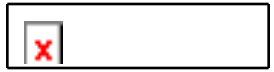


CoffeePack

CoffeeMaker

Copyright © www.jiuzhang.com

Core object



CoffeePack

CoffeeMaker

Expresso

Copyright © www.jiuzhang.com

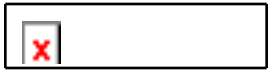
Use cases

- Coffee maker

- Brew

Copyright © www.jiuzhang.com

Classes



CoffeePack

CoffeeMaker

Expresso

Copyright © www.jiuzhang.com

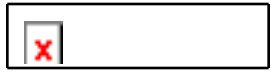
Classes

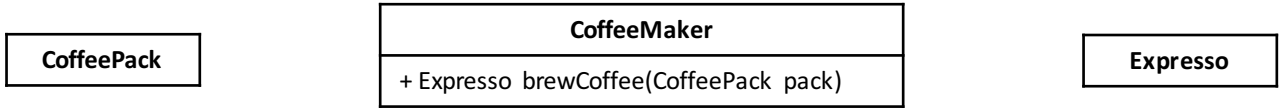
- Use case: Brew

Coffee machine expected to use a coffee pack to get expresso coffee

Copyright © www.jiuzhang.com

Classes





Copyright © www.jiuzhang.com

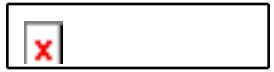
Use cases
Brew

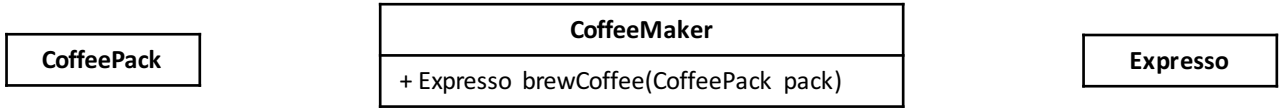
Challenge

如果需要能制作出多种咖啡(价格不同)，需要怎么做？

Copyright © www.jiuzhang.com

Classes

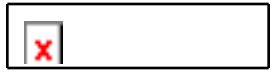


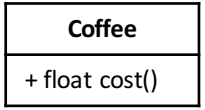


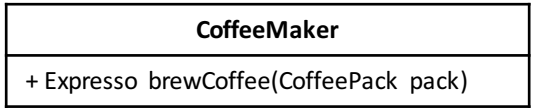
Copyright © www.jiuzhang.com

Use cases
Brew

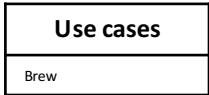
继承



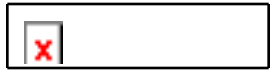


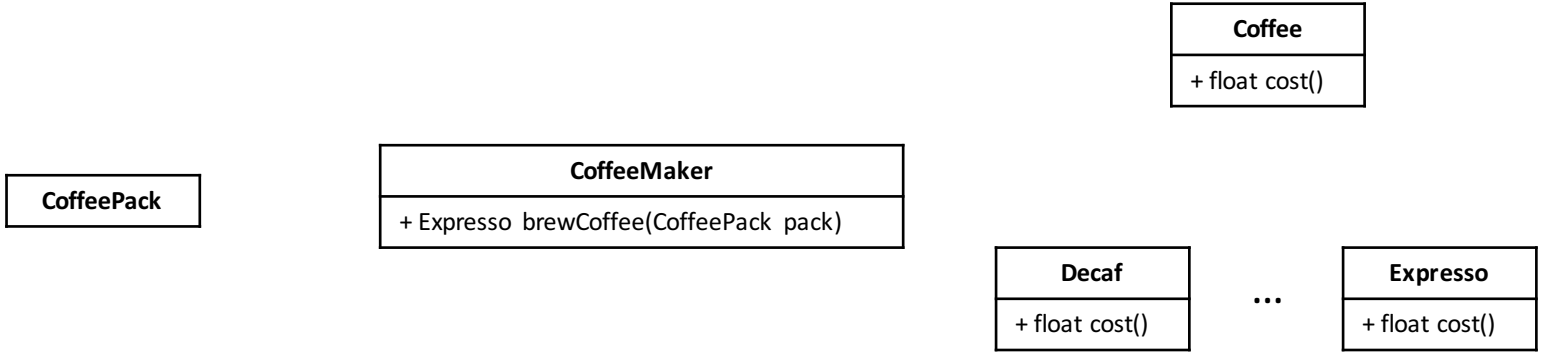


Copyright © www.jiuzhang.com

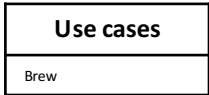


继承

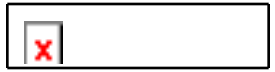


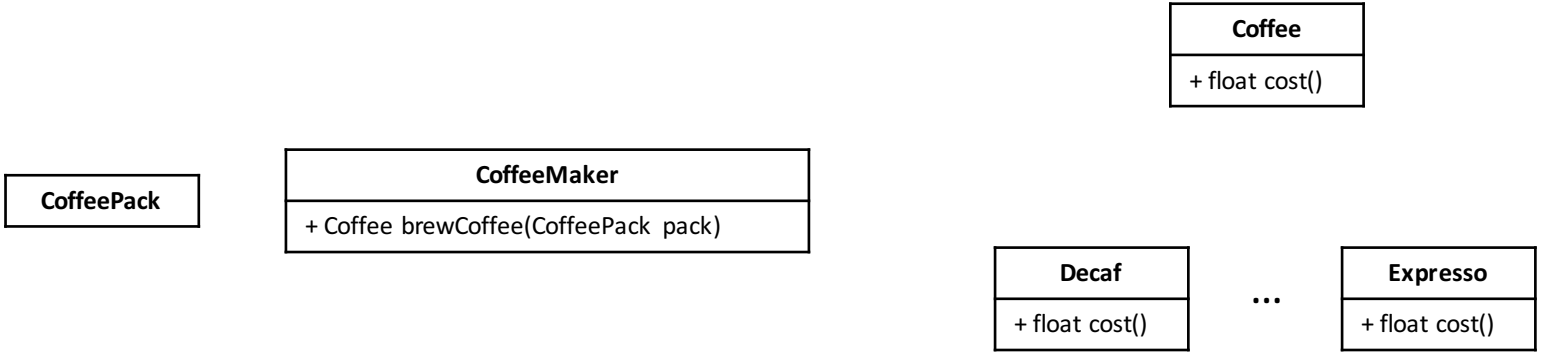


Copyright © www.jiuzhang.com



继承

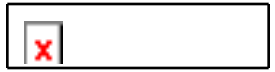


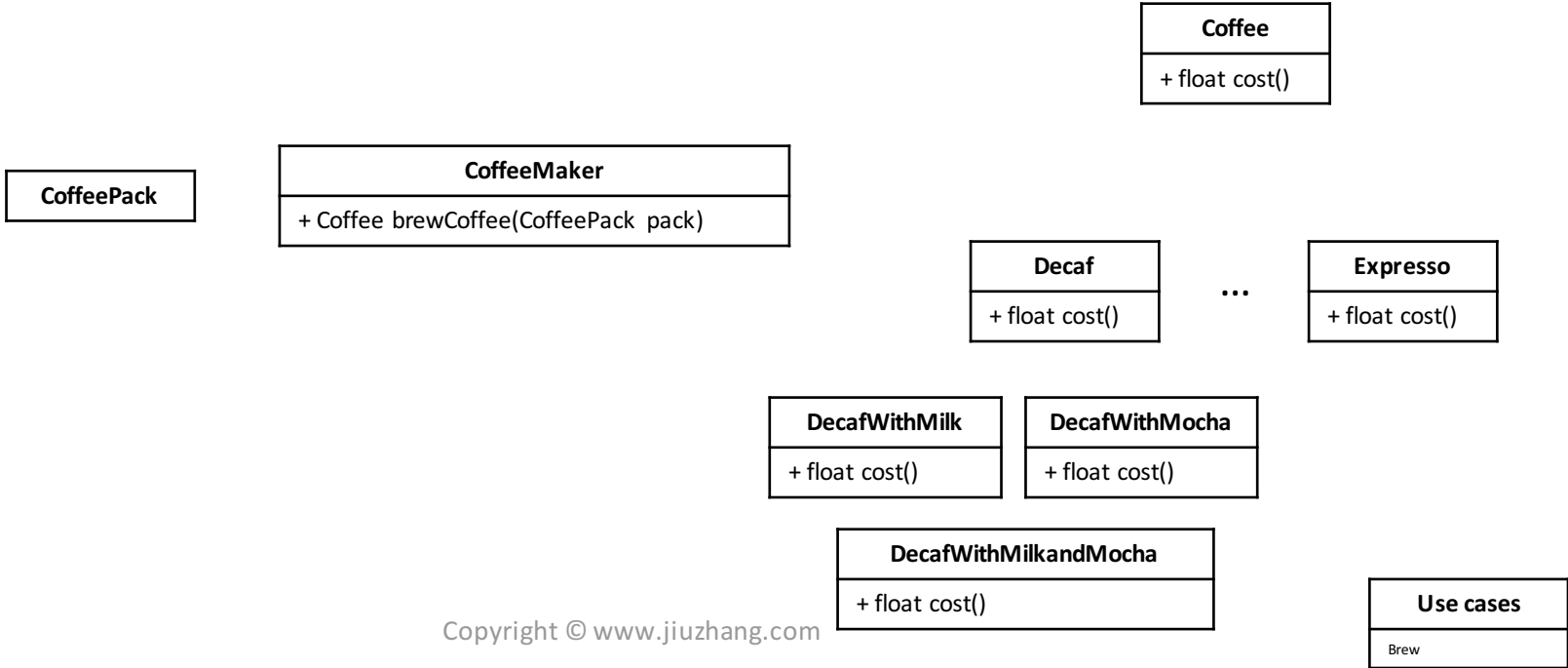


Copyright © www.jiuzhang.com

Use cases
Brew

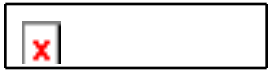
继承

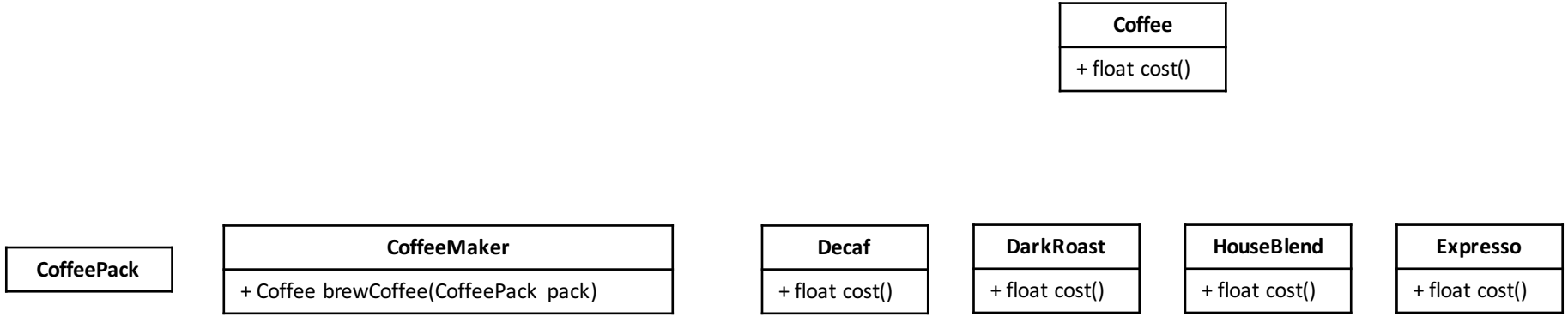




Copyright © www.jiuzhang.com

另一种继承

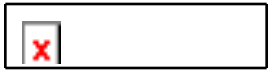


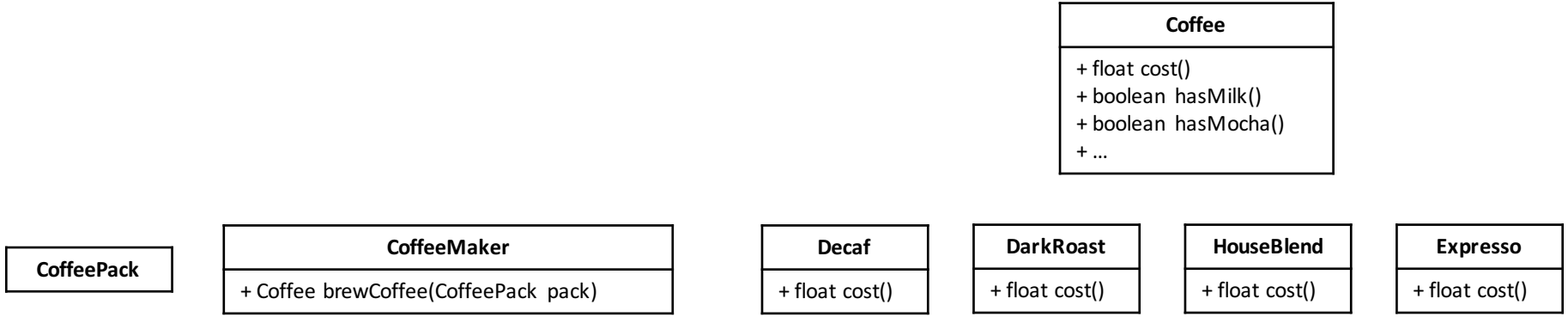


Copyright © www.jiuzhang.com

Use cases
Brew

另一种继承





Copyright © www.jiuzhang.com

Use cases
Brew

另一种继承

```
public float cost()
{
    if(hasMilk())
    {
        cost += 0.5;
    }

    if(hasMocha())
    {
        cost += 0.5;
    }

    if(hasSoy())
    {
        cost += 0.5;
    }

    ...

    return cost;
}
```

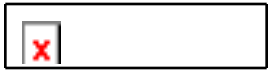
Copyright © www.jiuzhang.com

Decorator Design Pattern

- Decorator pattern allows a user to add new functionality to an existing object without altering its structure. This type of design pattern comes under structural pattern as this pattern acts as a wrapper to existing class.

Copyright © www.jiuzhang.com

Decorator



CoffeeMaker
+ Coffee brewCoffee(CoffeePack pack)

Coffee
+ float cost()

CoffeePack

Decaf
+ float cost()

DarkRoast
+ float cost()

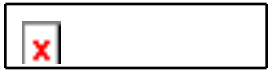
HouseBlend
+ float cost()

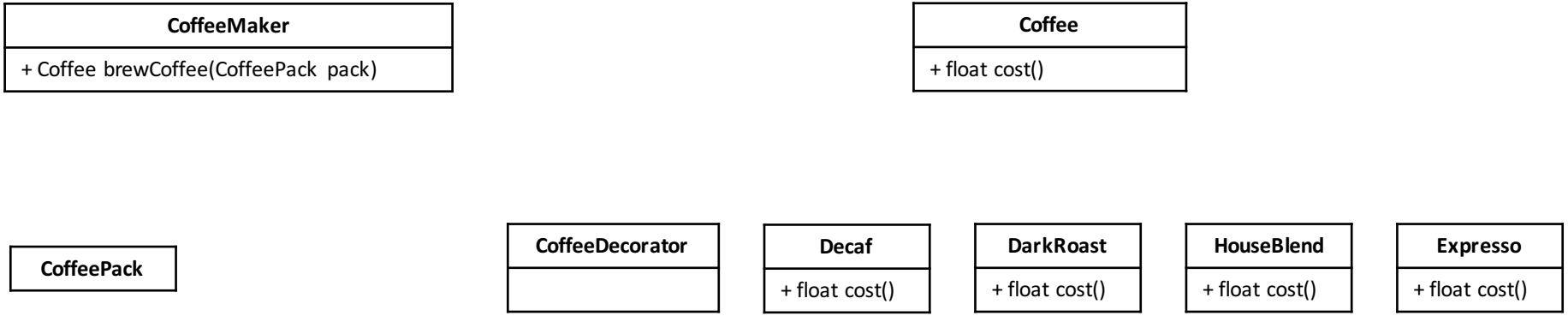
Expresso
+ float cost()

Use cases
Brew

Copyright © www.jiuzhang.com

Decorator

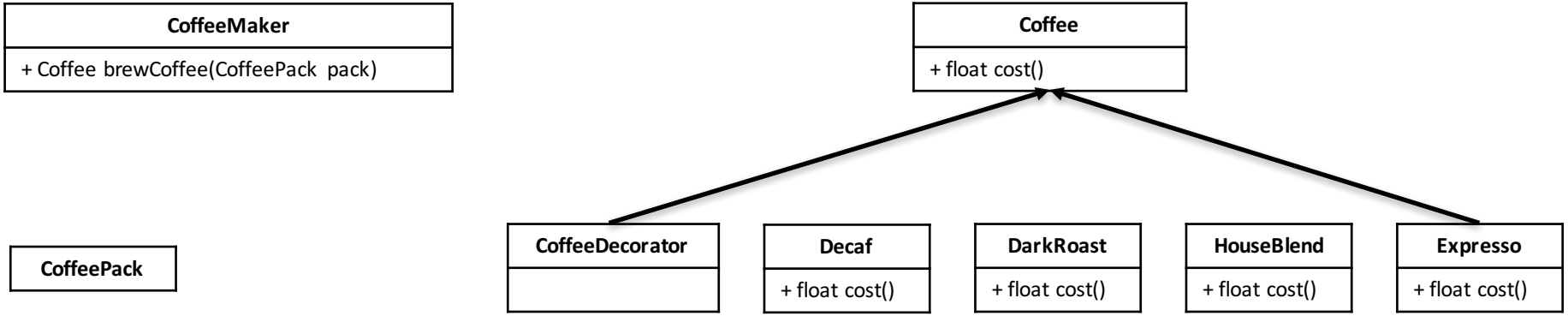




Use cases
Brew

Copyright © www.jiuzhang.com

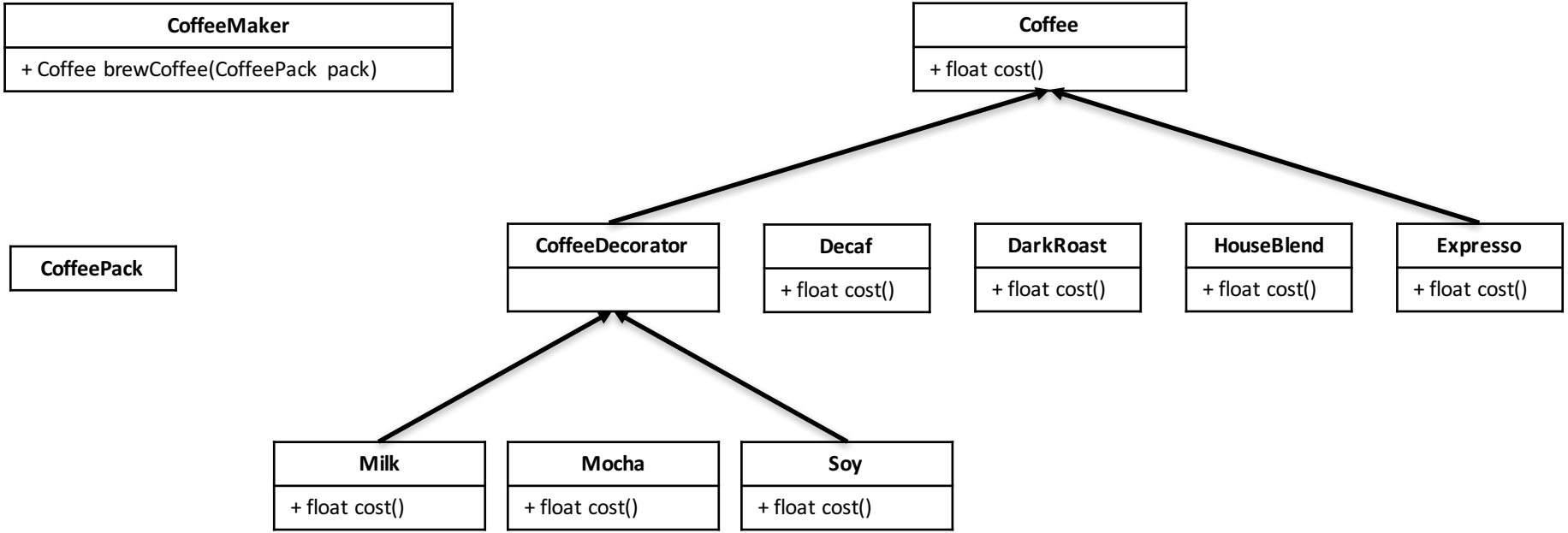
Decorator



Use cases
Brew

Copyright © www.jiuzhang.com

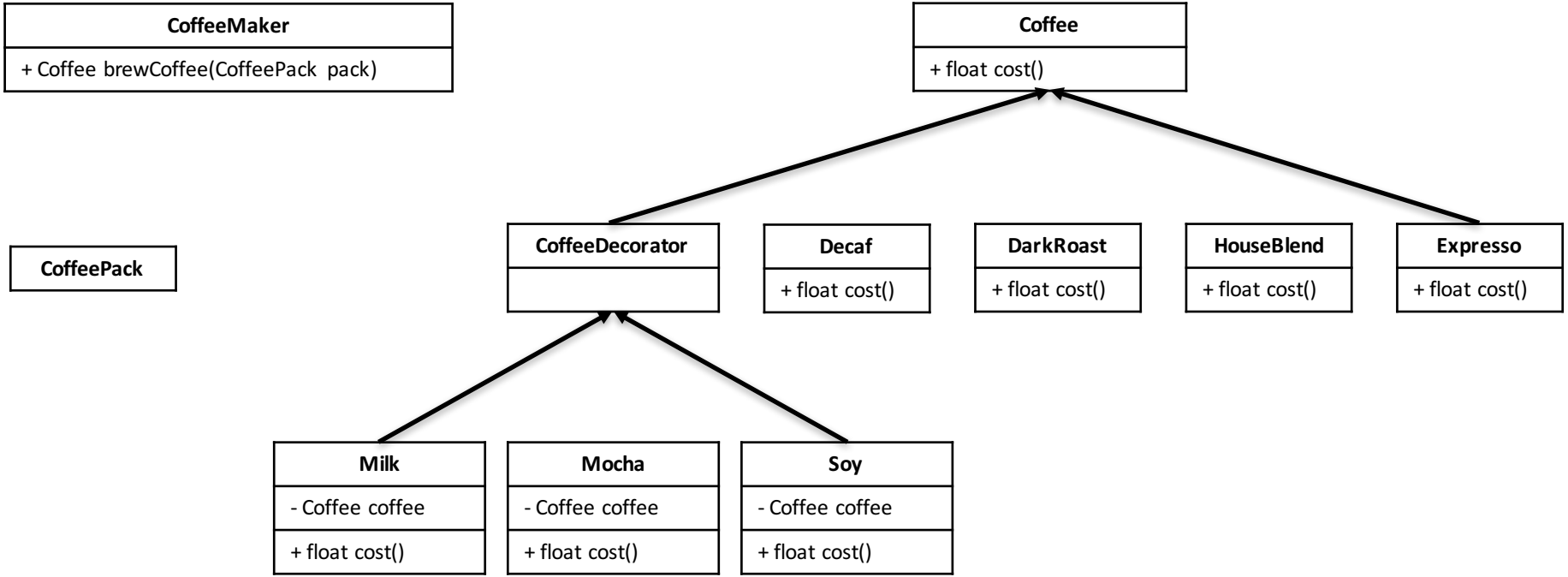
Decorator



Use cases
Brew

Copyright © www.jiuzhang.com

Decorator



Use cases
Brew

Copyright © www.jiuzhang.com

Decorator

```
public class Expresso extends Coffee
{

    public Expresso()
    {
        description = "Expresso";
    }

    public float cost()
    {
        return 1.99;
    }
}
```

Copyright © www.jiuzhang.com

Decorator

```java
public class Mocha extends CoffeeDecorator
{
    Coffee coffee;

    public Mocha(Coffee coffee)
    {
        this.coffee = coffee;
    }

    public String getDescription()
    {
        return coffee.getDescription() + ", Mocha";
    }

    public float cost()
    {
        return coffee.getCost() + 0.5;
    }
}
```

Copyright © www.jiuzhang.com

Decorator

```java
public void test()
{
    Coffee coffee_1 = new Expresso();

    System.out.println(coffee_1.getDescription() + " $ " + coffee_1.getCost());

    Coffee coffee_2 = new Mocha(coffee_1);

    System.out.println(coffee_2.getDescription() + " $ " + coffee_2.getCost());

    Coffee coffee_3 = new Mocha(coffee_2);

    System.out.println(coffee_3.getDescription() + " $ " + coffee_3.getCost());
}
```

Copyright © www.jiuzhang.com

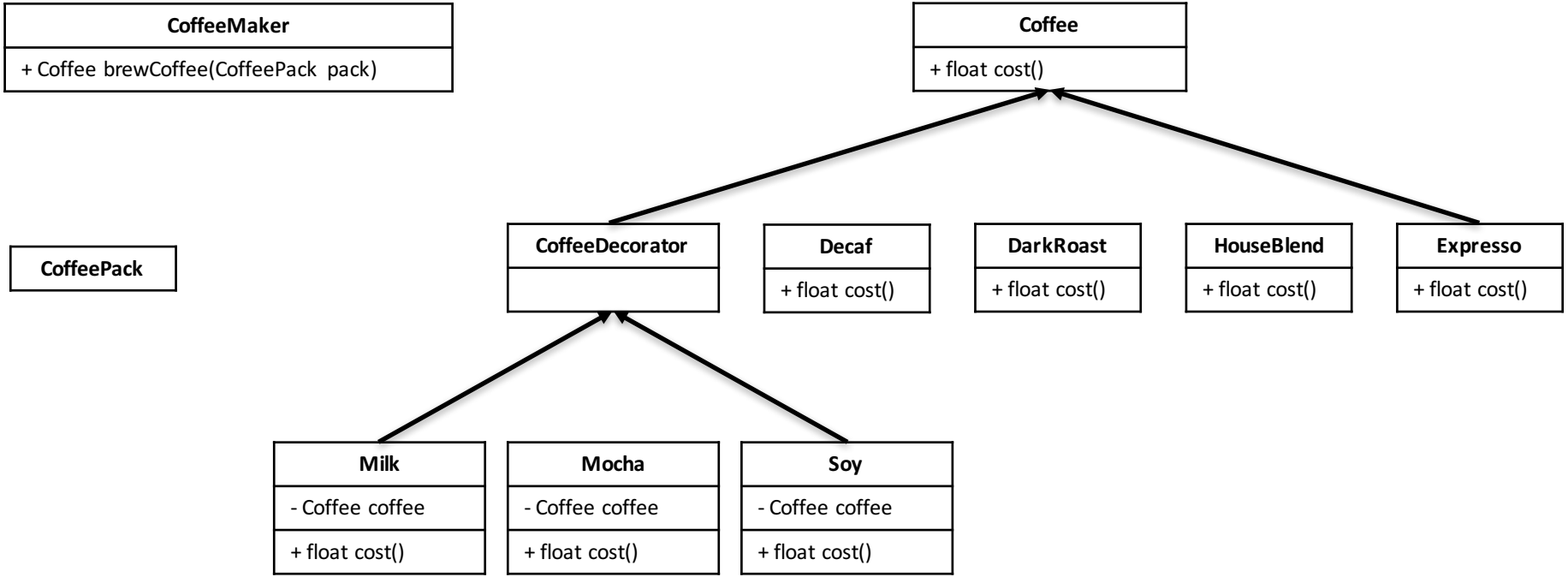
Decorator

```java
public void test()
{
    Coffee coffee_1 = new Expresso();

    System.out.println(coffee_1.getDescription() + " $ " + coffee_1.getCost());

    Coffee coffee_2 = new Mocha(coffee_1);

    System.out.println(coffee_2.getDescription() + " $ " + coffee_2.getCost());

    Coffee coffee_3 = new Mocha(coffee_2);

    System.out.println(coffee_3.getDescription() + " $ " + coffee_3.getCost());
}
```

```
Expresso $ 1.99

Expresso, Mocha $ 2.49

Expresso, Mocha, Mocha $ 2.99
```

Copyright © www.jiuzhang.com

Decorator

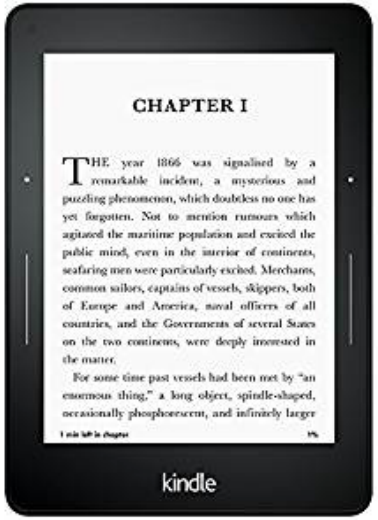


Use cases
Brew

Copyright © www.jiuzhang.com

Kindle

Can you design Kindle?



Copyright © www.jiuzhang.com

Clarify

- What

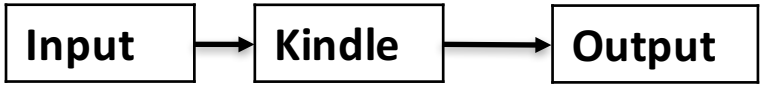
关键字： Kindle

Copyright © www.jiuzhang.com

Clarify

- What

关键字： Kindle

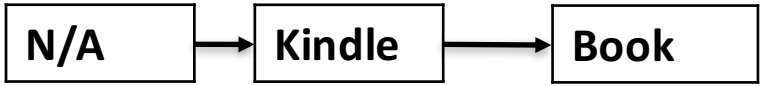


Copyright © www.jiuzhang.com

Clarify

- What

关键字： Kindle



Copyright © www.jiuzhang.com

Clarify

- What

关键字： Kindle, Book

Copyright © www.jiuzhang.com

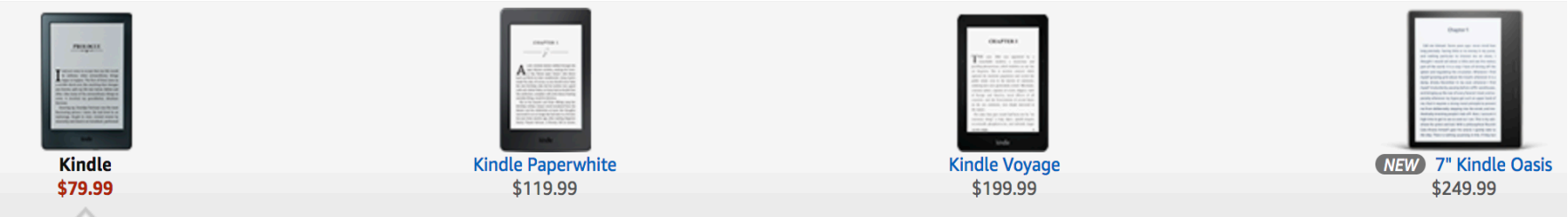
Clarify

关键字： Kindle

Copyright © www.jiuzhang.com

Clarify

关键字： Kindle

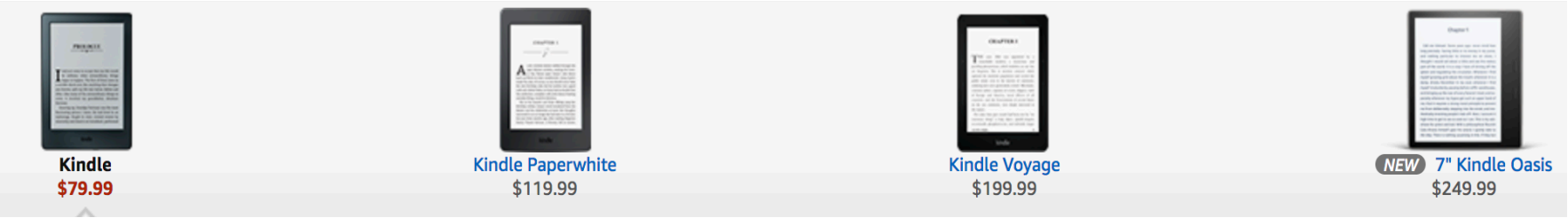


需不需要设计不同版本？

Copyright © www.jiuzhang.com

Clarify

关键字： Kindle



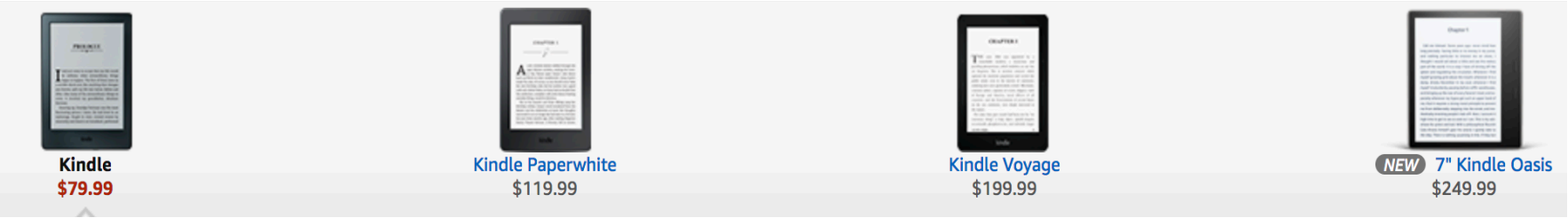
需不需要设计不同版本？

- Design: get price

Copyright © www.jiuzhang.com

Clarify

关键字： Kindle



需不需要设计不同版本？

- Design: get price
- Design: Memory difference

Copyright © www.jiuzhang.com

Clarify

关键字： Book

Copyright © www.jiuzhang.com

Clarify

关键字： Book

- 支持哪些格式的电子书？

Copyright © www.jiuzhang.com

Clarify

- 对于本题：

- 不需要考虑不同的版本
- 不需要考虑内存和书的大小
- 支持pdf, epub 和 mobi三种格式

Copyright © www.jiuzhang.com

Clarify

How

Copyright © www.jiuzhang.com

Clarify

如何获取电子书？

Copyright © www.jiuzhang.com

Clarify

如何获取电子书？

- 是否支持Upload
- 是否支持Download

Copyright © www.jiuzhang.com

Clarify

如何获取电子书？

- Upload
- Download

对于付费的电子书，提供哪些支付功能？

Copyright © www.jiuzhang.com

Clarify

如何获取电子书？

- Upload
- Download

对于付费的电子书，提供哪些支付功能？

Payment -> Strategy design pattern

Copyright © www.jiuzhang.com

Clarify

- 对于本题：支持上传，下载
- 对于本题：不需要考虑付费

Copyright © www.jiuzhang.com

Clarify

- Who && When

- N/A

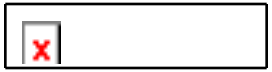
Copyright © www.jiuzhang.com

Core Object

Kindle

Copyright © www.jiuzhang.com

Core Object



Kindle

Book

Copyright © www.jiuzhang.com

Core Object

Kindle
- List<Book> library

Book

Copyright © www.jiuzhang.com

Use cases

- Kindle

Copyright © www.jiuzhang.com

Use cases

- Kindle

- Upload book

Copyright © www.jiuzhang.com

Use cases

- Kindle

- Upload book
- Download book

Copyright © www.jiuzhang.com

Use cases

- Kindle

- Upload book
- Download book
- Read book

Copyright © www.jiuzhang.com

Use cases

- Kindle

- Upload book
- Download book
- Read book
- Remove book

Copyright © www.jiuzhang.com

Classes

Kindle
- List<Book> library

Book

Use cases
Upload book
Download book
Read book
Remove book

Copyright © www.jiuzhang.com

Upload book

- Upload a file to kindle and store as a book

Copyright © www.jiuzhang.com

Classes

Kindle
- List<Book> library
+ void uploadBook(File f)

Book

Use cases
Upload book
Download book
Read book
Remove book

Copyright © www.jiuzhang.com

Classes

Kindle
- List<Book> library
+ void uploadBook(File f)

Book

UploadBookException

Use cases
Upload book
Download book
Read book
Remove book

Copyright © www.jiuzhang.com

Download book

- Download a book and put in library

Copyright © www.jiuzhang.com

Classes

Kindle
- List<Book> library
+ void uploadBook(File f) + void downloadBook(Book b)

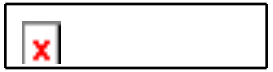
Book

UploadBookException

Use cases
Upload book
Download book
Read book
Remove book

Copyright © www.jiuzhang.com

Classes



Kindle
- List<Book> library
+ void uploadBook(File f) + void downloadBook(Book b)

Book

UploadBookException

DownloadBookException

Use cases
Upload book
Download book
Read book
Remove book

Copyright © www.jiuzhang.com

Read book

- Select a book and display it

Copyright © www.jiuzhang.com

Classes

Kindle
- List<Book> library
+ void uploadBook(File f) + void downloadBook(Book b) + void read(Book b)

Book

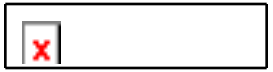
UploadBookException

DownloadBookException

Use cases
Upload book
Download book
Read book
Remove book

Copyright © www.jiuzhang.com

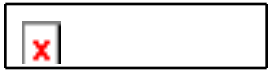
Remove book



- Remove a book from library

Copyright © www.jiuzhang.com

Classes



Kindle
- List<Book> library
+ void uploadBook(File f) + void downloadBook(Book b) + void read(Book b) + void remove(Book b)

Book

UploadBookException

DownloadBookException

Use cases
Upload book
Download book
Read book
Remove book

Copyright © www.jiuzhang.com

Challenge

- What about different book format?

Copyright © www.jiuzhang.com

Classes

Kindle
- List<Book> library
+ void uploadBook(File f) + void downloadBook(Book b) + void read(Book b) + void remove(Book b)

Book
- Format format

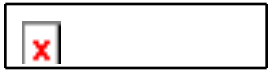
UploadBookException

DownloadBookException

Use cases
Upload book
Download book
Read book
Remove book

Copyright © www.jiuzhang.com

Classes



Kindle

- List<Book> library

+ void uploadBook(File f)
+ void downloadBook(Book b)
+ void read(Book b)
+ void remove(Book b)

Book

- Format format

UploadBookException

DownloadBookException

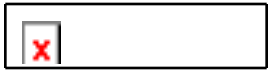
<<enumeration>>
Format

Use cases

- Upload book
- Download book
- Read book
- Remove book

Copyright © www.jiuzhang.com

Classes



Kindle

- List<Book> library

+ void uploadBook(File f)
+ void downloadBook(Book b)
+ void read(Book b)
+ void remove(Book b)

Book

- Format format

UploadBookException

DownloadBookException

<<enumeration>>
Format

PDF
EPUB
MOBI

Use cases

- Upload book
- Download book
- Read book
- Remove book

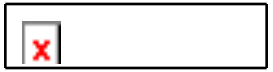
Copyright © www.jiuzhang.com

Challenge

- How would read book work?

Copyright © www.jiuzhang.com

Classes



Kindle

- List<Book> library

+ void uploadBook(File f)
+ void downloadBook(Book b)
+ void read(Book b)
+ void remove(Book b)

Book

- Format format

UploadBookException

DownloadBookException

<<enumeration>>
Format

PDF
EPUB
MOBI

Use cases

- Upload book
- Download book
- Read book
- Remove book

Copyright © www.jiuzhang.com

Challenge

- How would read book work?

```
public void read(Book book)
{
    if(book.getFormat == Format.PDF)
    {
        PDFReader reader = new PDFReader(book);
        reader.display();
    }
    else if(book.getFormat == Format.MOBI)
    {
        MOBIReader reader = new MOBIReader(book);
        reader.display();
    }
    else if(book.getFormat == Format.EPUB)
    {
        EPUBReader reader = new EPUBReader(book);
        reader.display();
    }
}
```

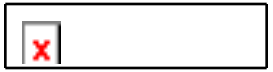
Copyright © www.jiuzhang.com

Challenge

- Solution: Factory design pattern

Copyright © www.jiuzhang.com

Factory design pattern



Kindle

- List<Book> library

+ void uploadBook(File f)
+ void downloadBook(Book b)
+ void read(Book b)
+ void remove(Book b)

Book

- Format format

UploadBookException

DownloadBookException

<<enumeration>>
Format

PDF
EPUB
MOBI

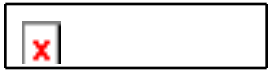
ReaderFactory

Use cases

- Upload book
- Download book
- Read book
- Remove book

Copyright © www.jiuzhang.com

Factory design pattern



Kindle

- List<Book> library

+ void uploadBook(File f)
+ void downloadBook(Book b)
+ void read(Book b)
+ void remove(Book b)

Book

- Format format

UploadBookException

DownloadBookException

<<enumeration>>
Format

PDF
EPUB
MOBI

ReaderFactory

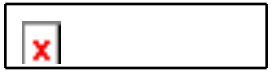
Reader

Use cases

- Upload book
- Download book
- Read book
- Remove book

Copyright © www.jiuzhang.com

Factory design pattern



Kindle

- List<Book> library

+ void uploadBook(File f)
+ void downloadBook(Book b)
+ void read(Book b)
+ void remove(Book b)

Book

- Format format

UploadBookException

DownloadBookException

<<enumeration>>
Format

PDF
EPUB
MOBI

ReaderFactory

Reader

PDFReader

MOBIReader

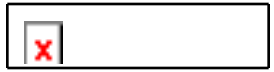
EPUBReader

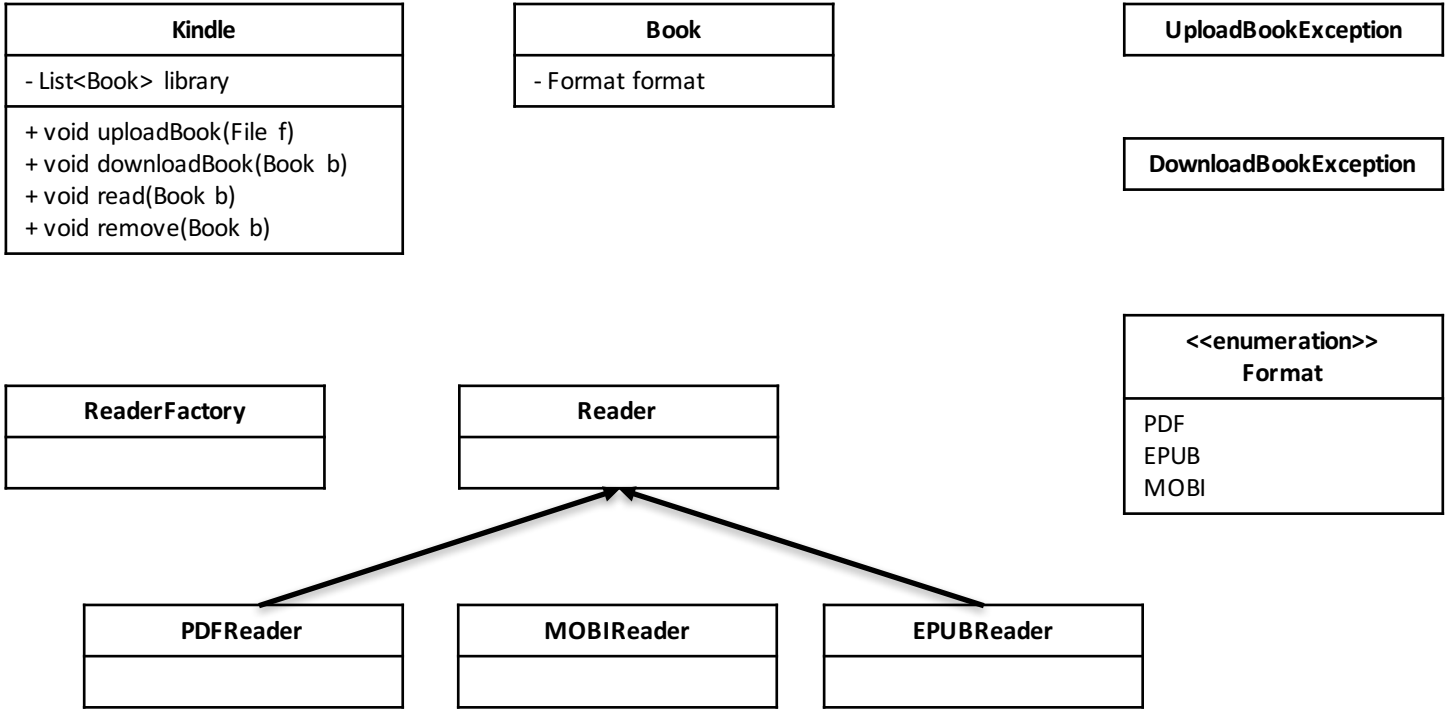
Use cases

- Upload book
- Download book
- Read book
- Remove book

Copyright © www.jiuzhang.com

Factory design pattern

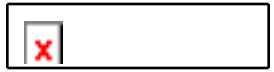


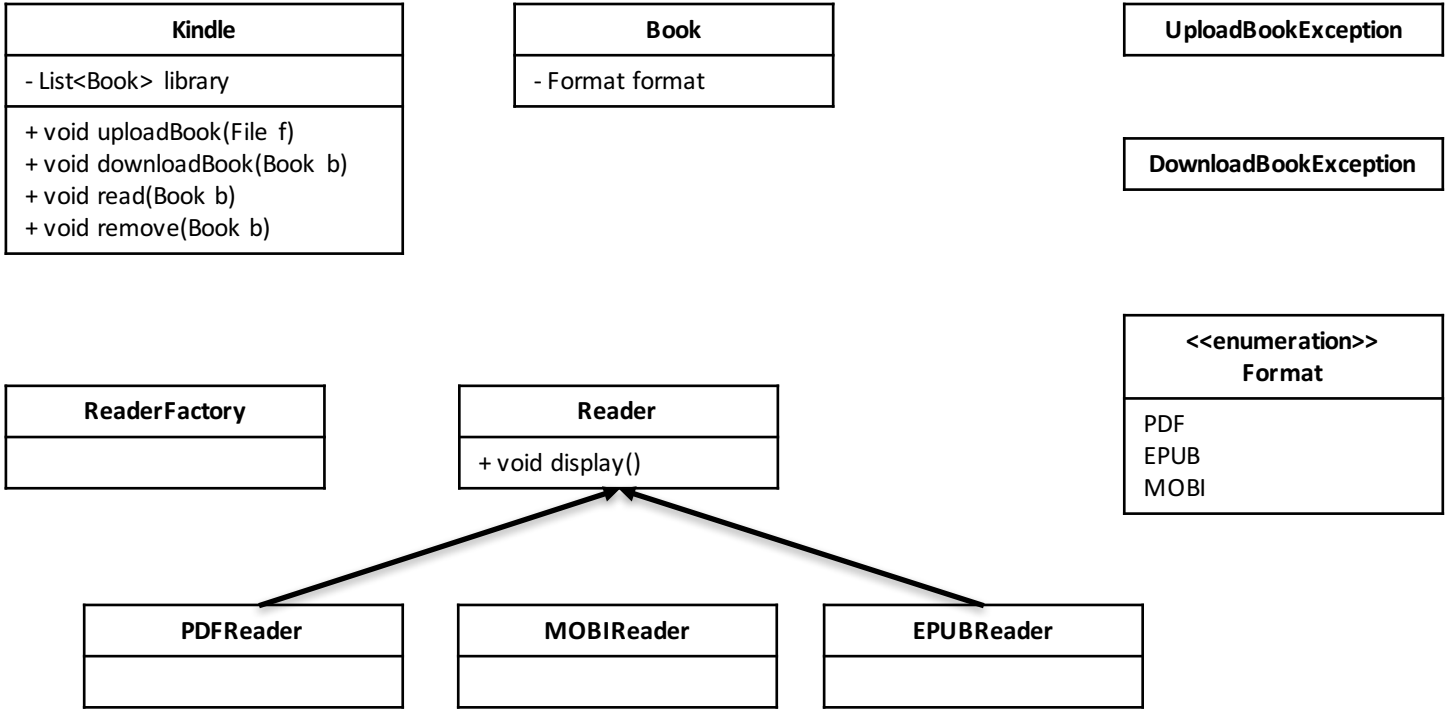


Use cases
Upload book
Download book
Read book
Remove book

Copyright © www.jiuzhang.com

Factory design pattern

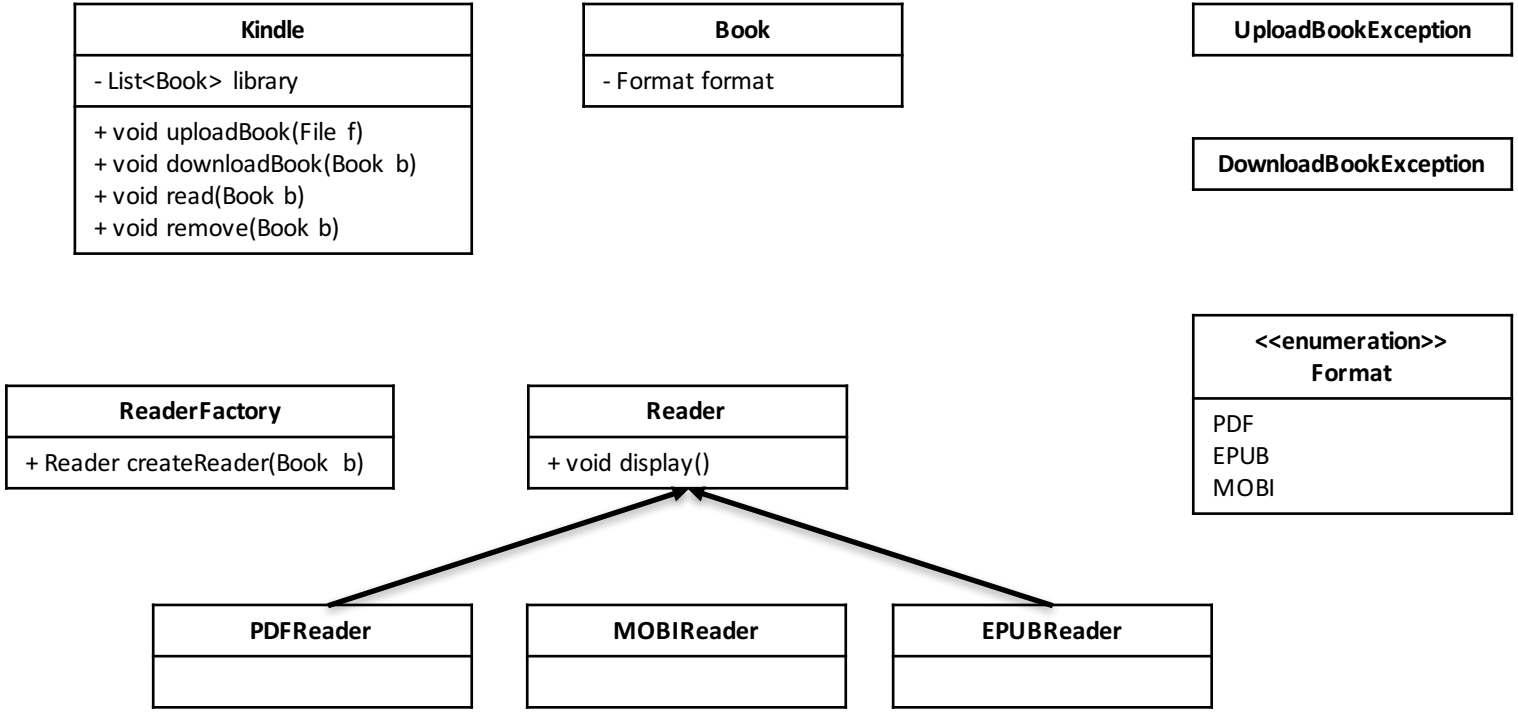




Use cases
Upload book
Download book
Read book
Remove book

Copyright © www.jiuzhang.com

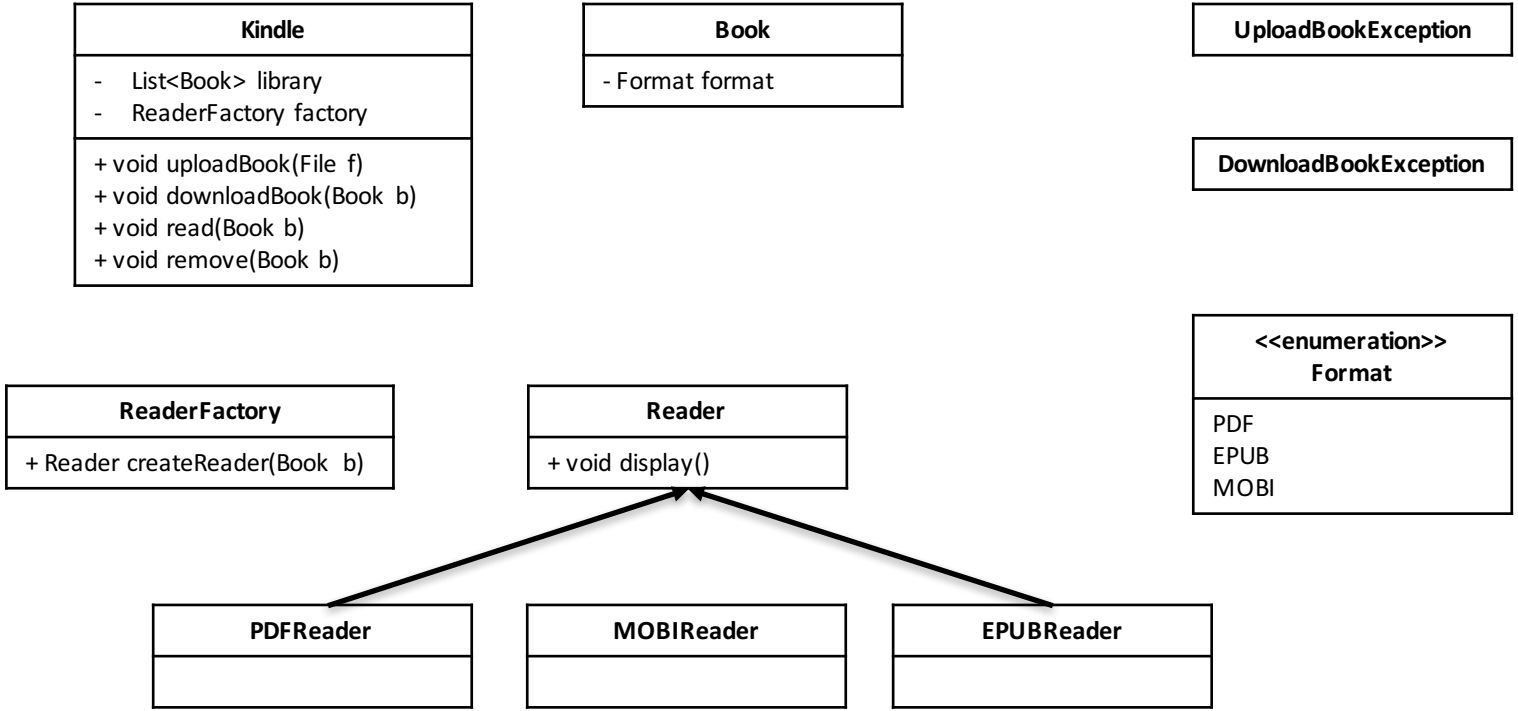
Factory design pattern



Use cases
Upload book
Download book
Read book
Remove book

Copyright © www.jiuzhang.com

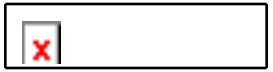
Factory design pattern



Use cases
Upload book
Download book
Read book
Remove book

Copyright © www.jiuzhang.com

Simple factory



```
public Reader createReader(Book book)
{
    if(book.getFormat == Format.PDF)
    {
        return new PDFReader(book);
    }
    else if(book.getFormat == Format.MOBI)
    {
        return new MOBIReader(book);
    }
    else if(book.getFormat == Format.EPUB)
    {
        return new EPUBReader(book);
    }
    retrun null;
}
```

```
Reader reader = factory.createReader(book);
reader.display();
```

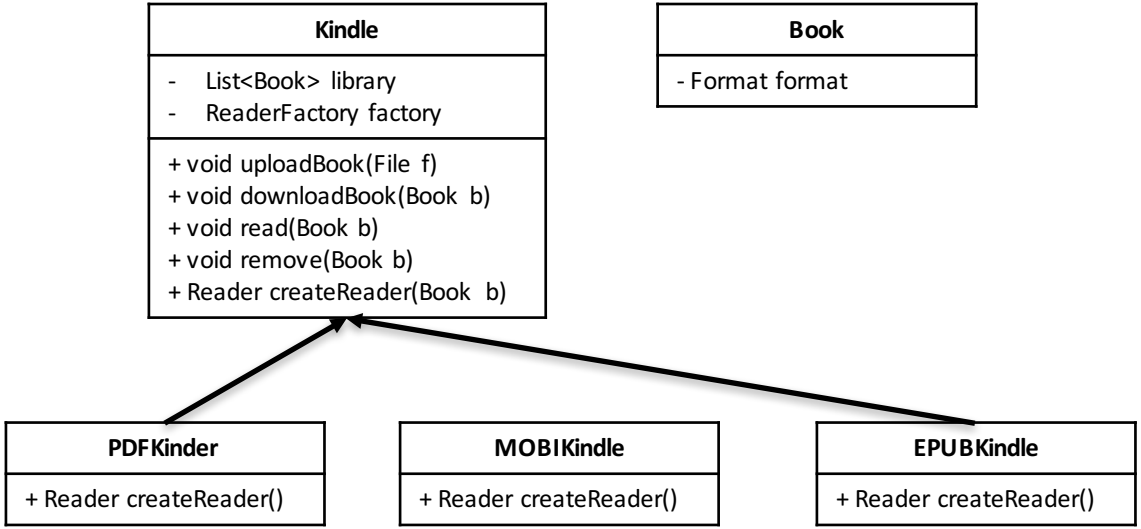
Copyright © www.jiuzhang.com

Factory design pattern

- Factory method
- Abstract factory

Copyright © www.jiuzhang.com

Factory method



UploadBookException

DownloadBookException

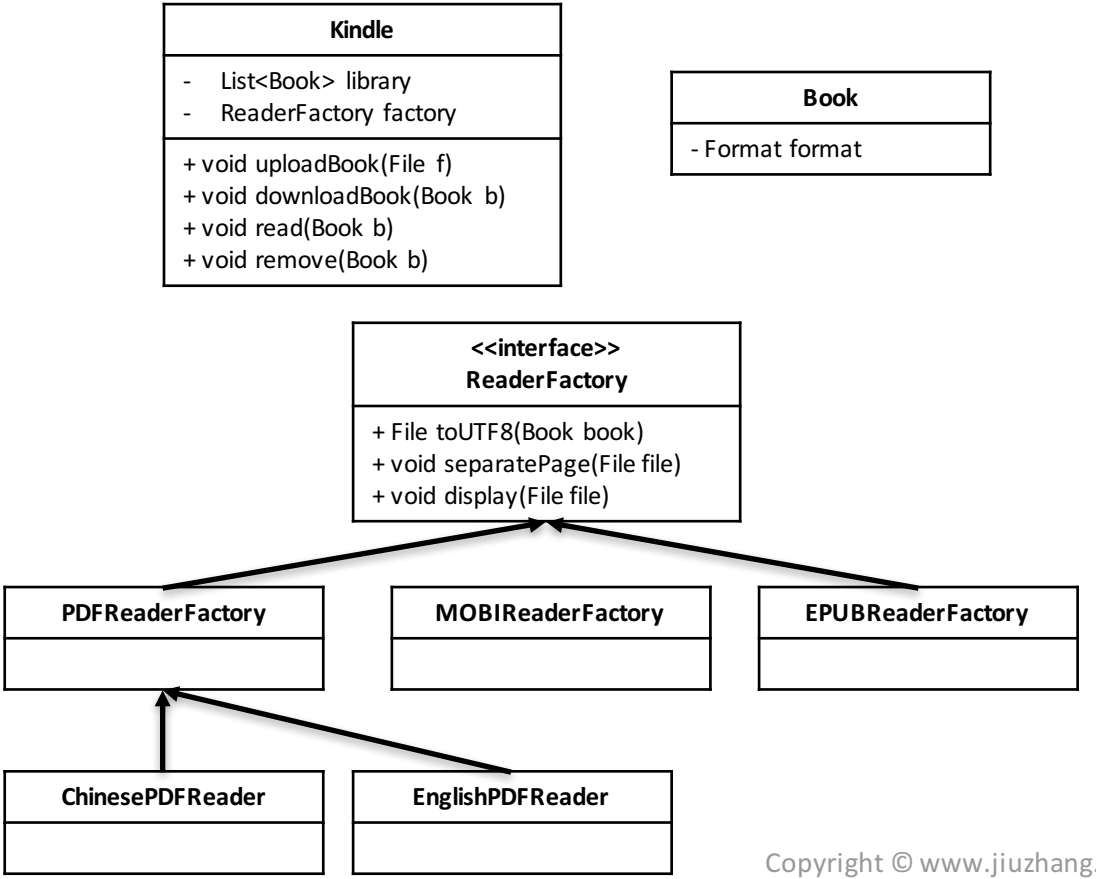
<<enumeration>>
Format

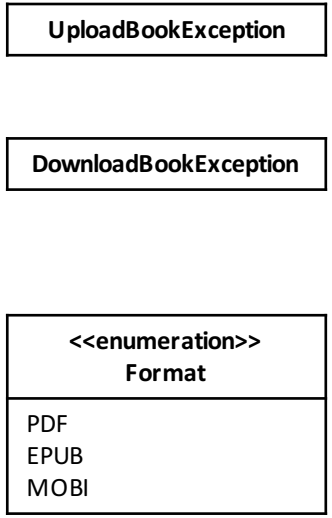
PDF
EPUB
MOBI

Use cases
Upload book
Download book
Read book
Remove book

Copyright © www.jiuzhang.com

Abstract factory





Use cases
Upload book
Download book
Read book
Remove book

Copyright © www.jiuzhang.com

Recap

- 常见的实物类面向对象设计

Copyright © www.jiuzhang.com

Recap

- 常见的实物类面向对象设计
- Input -> 题目主体 -> Output
- State design pattern
- Decorate design pattern
- Factory design pattern

Copyright © www.jiuzhang.com

问卷调查

- https://www.jiuzhang.com/course/10/questionnaire/

Copyright © www.jiuzhang.com

Q & A

The image part with relationship ID rId2 was not found in the file.

扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: ninechapter
知乎专栏： http://zhuanlan.zhihu.com/jiuzhang
微博: http://www.weibo.com/ninechapter
官网: www.jiuzhang.com

Copyright © www.jiuzhang.com