预定类面向对象设计

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

Restaurant

- Can you design a restaurant?

Clarify

- What
- How

Clarify

- What

关键字：Restaurant

Clarify

- What

关键字：Restaurant

In / Out：？

Clarify

- What

关键字：Restaurant

In / Out ：

Party / Table

Clarify

- What

关键字：Restaurant, Party, Table

What

关键字：Restaurant

属性：？

What

关键字：Restaurant

What

关键字：Restaurant

- 吧台

- 大堂

- 包间

What

关键字：Restaurant

- 桌子的规格不一样，能坐的人数区别
- 吧台里，party的区分
- 收费？

Clarify

- 针对本题

大堂，所有桌子都一样, 暂不考虑人数限制

Clarify

- How

规则？

Clarify

- How

- 是否能够预约？

Clarify

- How

- 是否能够送外卖？

Clarify

- How

- 每个Order需要区分是Dine-in还是Dine-out

Clarify

- 针对本题

没有Reservation
没有Dine-out

Clarify

思考模式1：

Party 进入餐馆 -> Host指引到空桌 (find table) ->
一个waiter负责这桌客人 (assign waiter) -> 客人点菜 (ta
Chief 拿到order，按顺序做菜 (cook by order) -> Order
(serve order) -> 客人吃完后付钱 (check out)

Clarify

思考模式2：

1. 客人进入餐馆，餐馆返回一个Table
2. 客人点菜，餐馆返回一桌菜
3. 客人付账，餐馆清空Table

Clarify

思考模式1：

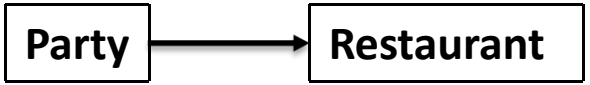
Clarify

思考模式2：

Core Object

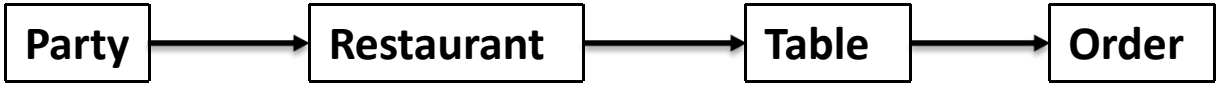
Restaurant

Core Object



Core Object

Core Object



Core Object

Core Object

Party

Restaurant
- List<Table> tables

Table

Order

Meal

Core Object

Party

Restaurant
- List<Table> tables - List<Meal> menu

Table

Order

Meal

Core Object

Party

Restaurant
- List<Table> tables - List<Meal> menu

Table
- Party p

Order

Meal

Core Object

Party

Restaurant
- List<Table> tables - List<Meal> menu

Table

Order

Meal

Core Object

Party

Restaurant
- List<Table> tables - List<Meal> menu

Table

Order
- List<Meal> meals

Meal

Cases

- Party

Cases

- Party

- Make order ?

Cases

- Restaurant

Cases

- Restaurant

- Find table

Cases

- Restaurant

- Find table
- Take Order

Cases

- Restaurant

- Find table
- Take Order
- Checkout

Cases

- Table

- N/A

Cases

- Order

- N/A

Cases

- Meal

- N/A

Use case summary

- Mangement 类常见use case

Reserve:

Serve:

Checkout:

Use case summary

- Mangement 类常见use case

Reserve: 暂不考虑

Use case summary

- Mangement 类常见use case

Reserve: 暂不考虑

Serve: Find table, Take order

Use case summary

- Mangement 类常见use case

Reserve: 暂不考虑

Serve: Find table, Take order

Checkout: checkout

Class

Party

Restaurant

- List<Table> tables
- List<Meal> menu

Table

Order

- List<Meal> meals

Meal

Use cases

Find table

Take order

Checkout

Class

- Use case: Find table

- Restaurant find an available table, an change the table to be unavailable

Class

Party

Restaurant

- List<Table> tables
- List<Meal> menu

+ Table findTable()

Table

Order

- List<Meal> meals

Meal

Use cases

Find table

Take order

Checkout

Class

Party

Restaurant

- List<Table> tables
- List<Meal> menu

+ Table findTable()

Table

- Boolean available

Order

- List<Meal> meals

Meal

Use cases

Find table

Take order

Checkout

Class

Party

Restaurant

- List<Table> tables
- List<Meal> menu

+ Table findTable()

Table

- Boolean available

+ boolean isAvailable()

Order

- List<Meal> meals

Meal

Use cases

Find table

Take order

Checkout

Class

Party

Restaurant

- List<Table> tables
- List<Meal> menu

+ Table findTable()

Table

- Boolean available

+ boolean isAvailable()
+ void changeStatus()

Order

- List<Meal> meals

Meal

Use cases

Find table

Take order

Checkout

Best practice

Party

Restaurant

- List<Table> tables
- List<Meal> menu

+ Table findTable()

Table

- Boolean available

+ boolean isAvailable()
+ void changeStatus()

Order

- List<Meal> meals

Meal

Use cases

Find table

Take order

Checkout

Best practice

Party

Restaurant

- List<Table> tables
- List<Meal> menu

+ Table findTable()

Table

- Boolean available

+ boolean isAvailable()
+ void markUnavailable()

Order

- List<Meal> meals

Meal

Use cases

Find table

Take order

Checkout

Class

- Use case: Take order

- Restaurant takes an order

Class

Party

Restaurant

- List<Table> tables
- List<Meal> menu

+ Table findTable()
+ void takeOrder(Order o)

Table

- Boolean available

+ boolean isAvailable()
+ void markUnavailable()

Order

- List<Meal> meals

Meal

Use cases

Find table

Take order

Checkout

Class

- Use case: Check out

- Restaurant checks out a table/order, mark the table available again
- Calculate order price

Class

Party

Restaurant

- List<Table> tables
- List<Meal> menu

+ Table findTable()
+ void takeOrder(Order o)

Table

- Boolean available

+ boolean isAvailable()
+ void markUnavailable()

Order

- List<Meal> meals

Meal

Use cases
Find table
Take order
Checkout

Class

Party

Restaurant

- List<Table> tables
- List<Meal> menu

+ Table findTable()
+ void takeOrder(Order o)

Table

- Boolean available

+ boolean isAvailable()
+ void markUnavailable()

Order

- List<Meal> meals
- Table t

Meal

Use cases

Find table

Take order

Checkout

Class

Party

Restaurant

- List<Table> tables
- List<Meal> menu

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)

Table

- Boolean available

+ boolean isAvailable()
+ void markUnavailable()

Order

- List<Meal> meals
- Table t

Meal

Use cases

Find table

Take order

Checkout

Class

Party

Restaurant

- List<Table> tables
- List<Meal> menu

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)

Table

- Boolean available

+ boolean isAvailable()
+ void markUnavailable()
+ void markAvailable()

Order

- List<Meal> meals
- Table t

Meal

Use cases

Find table

Take order

Checkout

Class

Party

Restaurant

- List<Table> tables
- List<Meal> menu

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)

Table

- Boolean available

+ boolean isAvailable()
+ void markUnavailable()
+ void markAvailable()

Order

- List<Meal> meals
- Table t

Meal

- Float price

+ float getPrice()

Use cases

Find table

Take order

Checkout

Class

Party

Restaurant

- List<Table> tables
- List<Meal> menu

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)

Table

- Boolean available

+ boolean isAvailable()
+ void markUnavailable()
+ void markAvailable()

Order

- List<Meal> meals
- Table t

+ float getPrice()

Meal

- Float price

+ float getPrice()

Use cases

Find table

Take order

Checkout

Class

Party

Restaurant

- List<Table> tables
- List<Meal> menu

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)

Table

- Boolean available

+ boolean isAvailable()
+ void markUnavailable()
+ void markAvailable()

Order

- List<Meal> meals
- Table t
- Party p

+ float getPrice()

Meal

- Float price

+ float getPrice()

Use cases

Find table

Take order

Checkout

Challenge

- Share table?

Class

Party

Restaurant

- List<Table> tables
- List<Meal> menu
- Map<Table, List<Order>> orders

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)

Table

- Boolean available

+ boolean isAvailable()
+ void markUnavailable()
+ void markAvailable()

Order

- List<Meal> meals
- Table t
- Party p

+ float getPrice()

Meal

- Float price

+ float getPrice()

Use cases

Find table

Take order

Checkout

Class

Party

Restaurant

- List<Table> tables
- List<Meal> menu
- Map<Table, List<Order>> orders

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)

Table

- Boolean available
- Int availableSeats

+ boolean isAvailable()

Order

- List<Meal> meals
- Table t
- Party p

+ float getPrice()

Meal

- Float price

+ float getPrice()

Use cases

Find table

Take order

Checkout

Class

Party

- Int size

Restaurant

- List<Table> tables
- List<Meal> menu
- Map<Table, List<Order>> orders

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)

Table

- Boolean available
- Int availableSeats

+ boolean isAvailable()

Order

- List<Meal> meals
- Table t
- Party p

+ float getPrice()

Meal

- Float price

+ float getPrice()

Use cases

Find table

Take order

Checkout

Class

Party

- Int size

Restaurant

- List<Table> tables
- List<Meal> menu
- Map<Table, List<Order>> orders

+ Table findTable(Party p)
+ void takeOrder(Order o)
+ void checkout(Order o)

Table

- Boolean available
- Int availableSeats

+ boolean isAvailable()

Order

- List<Meal> meals
- Table t
- Party p

+ float getPrice()

Meal

- Float price

+ float getPrice()

Use cases

Find table

Take order

Checkout

Correctness

Party

Restaurant

- List<Table> tables
- List<Meal> menu
- Map<Table, List<Order>> orders

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)

Table

- Boolean available

+ boolean isAvailable()
+ void markUnavailable()
+ void markAvailable()

Order

- List<Meal> meals
- Table t
- Party p

+ float getPrice()

Meal

- Float price

+ float getPrice()

NoTableException

Use cases

Find table

Take order

Checkout

Challenge

- How can you change your design to support reservation in your restaurant?

课程大纲

九章算法

- 预定类OOD题型
- 预定类OOD解题思路
- Hotel reservation system
- Booking.com

预定类题库

九章算法

- Restaurant reservation system

预定类题库

- Restaurant reservation system
- Hotel reservation system

预定类题库

九章算法

- Restaurant reservation system
- Hotel reservation system
- Flight/Bus/Train reservation system

预定类题目特点

- 频率：中

预定类题目特点

- 频率：中
- 难度：高

预定类题目解题思路

九章算法

- What

- 考虑预定的东西

预定类题目解题思路

九章算法

- What

- 考虑预定的东西

例子：机票

预定类题目解题思路

九章算法

- What

- 考虑预定的东西

例子：机票

机舱/座位号/…

预定类题目解题思路

九章算法

- Use case

- Search
- Select

- Cancel

预定类题目解题思路

九章算法

- Use case

Search criteria -> Search() -> List<Result> -> Select() -> Receipt

Reservation

- 有哪些需要和面试官统一的contract?

Reservation

- 有哪些需要和面试官统一的contract?

Search criteria -> Search() -> List<Result> -> Select() -> Receipt

Reservation

- Search criteria

Reservation

- Search criteria

Make a reservation

2 people | Oct 10, 2017 | 7:00 PM | Find a Table

Reservation

- Search criteria

Make a reservation

2 people | Oct 10, 2017 | 7:00 PM | Find a Table

- 人数： 无拼桌，每张桌子大小相同，不会有超过桌子大小的人数

Reservation

- Search criteria

Make a reservation

2 people | Oct 10, 2017 | 7:00 PM | Find a Table

- 人数：无拼桌，每张桌子大小相同，不会有超过桌子大小的人数
- 日期：是否允许预定多日以后的？

Reservation

- Search criteria

Make a reservation

2 people | Oct 10, 2017 | 7:00 PM | Find a Table

- 人数：无拼桌，每张桌子大小相同，不会有超过桌子大小的人数
- 日期：是否允许预定多日以后的？
- 时间： 是否所有时间都允许预定？

Reservation

- Search criteria

Make a reservation

2 people | Oct 10, 2017 | 7:00 PM | Find a Table

- 人数：无拼桌，每张桌子大小相同，不会有超过桌子大小的人数
- 日期：是否允许预定多日以后的？ - 允许
- 时间： 是否所有时间都允许预定？ - 24/7

Reservation

- Search criteria

Make a reservation

2 people | Oct 10, 2017 | 7:00 PM | Find a Table

- 人数：无拼桌，每张桌子大小相同，不会有超过桌子大小的人数
- 日期：是否允许预定多日以后的？ - 允许
- 时间： 是否所有时间都允许预定？ - 24/7

- Design: FindTableForReservation(Timeslot t)

Reservation

- Search criteria

Make a reservation

2 people | Oct 10, 2017 | 7:00 PM | Find a Table

- 人数：无拼桌，每张桌子大小相同，不会有超过桌子大小的人数
- 日期：是否允许预定多日以后的？ - 允许
- 时间： 是否所有时间都允许预定？ - 24/7

- Design: FindTableForReservation(Timeslot t)
- Timeslot contains Date and time

Reservation

- List<Result>

Reservation

- List<Result>

-当选择的时间段可以/不行时，系统应该给出什么反馈？

Reservation

- 做法一：

Make a reservation

2 people | Nov 2, 2017 | 7:30 PM | Find a Table

Booked 22 times today

5:30 PM | 6:00 PM | 7:30 PM | 8:00 PM | 8:30 PM

See affiliated restaurants ›

Make a reservation

2 people | Oct 10, 2017 | 7:00 PM | Find a Table

Booked 22 times today

You're in luck! We still have 1 timeslot left

9:30 PM

See affiliated restaurants ›

Reservation

- List<Result>

Result == Timeslot

Reservation

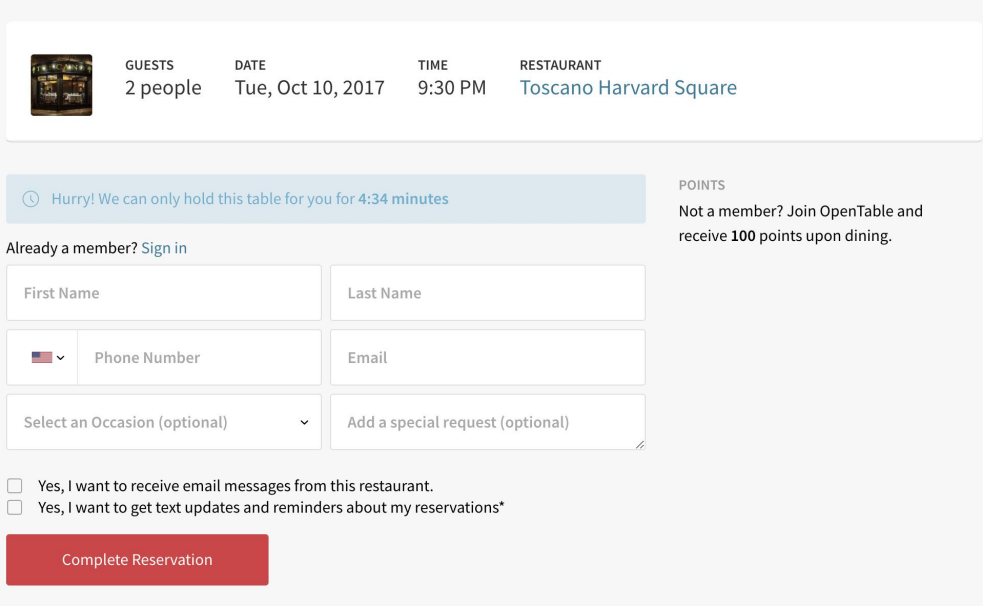
- List<Result>

Result == Timeslot

- Design: List<Timeslot> findTableForReservation(Timeslot t)
- Possible Challenge: 跟面试官讨论如何获得这个List

Reservation

- 做法二：
  - 可以预定：直接进入Confirm阶段
  - 不能预定：Throw exception / Show message



Reservation

- 做法二：
  - 可以预定：直接进入Confirm阶段
  - 不能预定：Throw exception / Show message

- Design: Pair<Table, Timeslot> findTableForReservation(Time slot) throws NoTableForReservationException

Reservation

- 做法二：
  - 可以预定：直接进入Confirm阶段
  - 不能预定：Throw exception / Show message

- Design: Pair<Table, Timeslot> findTableForReservation(Time slot) throws NoTableForReservationException
- Design: void confirmReservation(Pair<Table, Timeslot> reservation)

Reservation

- 为什么我们可以跳过List<Result>这个步骤？

Reservation

- 为什么我们可以跳过List<Result>这个步骤？

因为Table是一样的，用户不用选择也不会知道是订1号桌还是2号桌

Restaurant

Party

- Int capacity

+ int getCapacity()

Restaurant

- List<Table> tables
- List<Meal> menu
- Map<Table, List<Order>> orders

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)

Table

- Boolean available
- Int capacity

+ boolean isAvailable()
+ void markUnavailable()
+ void markAvailable()
+ get capacity()

Order

- List<Meal> meals
- Table t
- Party p

+ float getPrice()

Meal

- Float price

+ float getPrice()

NoTableException

Use cases

Find table

Take order

Checkout

Restaurant

九章算法

Party

- Int capacity

+ int getCapacity()

Restaurant

- List<Table> tables
- List<Meal> menu
- Map<Table, List<Order>> orders

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)

Table

- Boolean available
- Int capacity

+ boolean isAvailable()
+ void markUnavailable()
+ void markAvailable()
+ get capacity()

Order

- List<Meal> meals
- Table t
- Party p

+ float getPrice()

Meal

- Float price

+ float getPrice()

Reservation

- Table table
- Timeslot timeslot

NoTableException

Use cases

Find table

Take order

Checkout

Restaurant

Party

- Int capacity

+ int getCapacity()

Table

- Boolean available
- Int capacity

+ boolean isAvailable()
+ void markUnavailable()
+ void markAvailable()
+ get capacity()

Order

- List<Meal> meals
- Table t
- Party p

+ float getPrice()

Meal

- Float price

+ float getPrice()

Restaurant

- List<Table> tables
- List<Meal> menu
- Map<Table, List<Order>> orders

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)
+ Reservation findTableForReservation(Timeslot t)

Reservation

- Table table
- Timeslot timeslot

NoTableException

Use cases

Find table

Take order

Checkout

Restaurant

Party

- Int capacity

+ int getCapacity()

Table

- Boolean available
- Int capacity

+ boolean isAvailable()
+ void markUnavailable()
+ void markAvailable()
+ get capacity()

Order

- List<Meal> meals
- Table t
- Party p

+ float getPrice()

Meal

- Float price

+ float getPrice()

Restaurant

- List<Table> tables
- List<Meal> menu
- Map<Table, List<Order>> orders

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)
+ Reservation findTableForReservation(Timeslot t)

Reservation

- Table table
- Timeslot timeslot

NoTableException

NoTableForReservationExceptio n

Use cases

Find table

Take order

Checkout

Restaurant

Party

- Int capacity

+ int getCapacity()

Table

- Boolean available
- Int capacity

+ boolean isAvailable()
+ void markUnavailable()
+ void markAvailable()
+ get capacity()

Order

- List<Meal> meals
- Table t
- Party p

+ float getPrice()

Meal

- Float price

+ float getPrice()

Restaurant

- List<Table> tables
- List<Meal> menu
- Map<Table, List<Order>> orders

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)
+ Reservation findTableForReservation(Timeslot t)
+ void confirmReservation(Reservation r)

Reservation

- Table table
- Timeslot timeslot

NoTableException

NoTableForReservationExceptio n

Use cases

Find table

Take order

Checkout

Any problems?

Any problems?

- How to know if a table is open for reservation for a timeslot?

Any problems?

- How to know if a table is open for reservation for a timeslot?

要知道每张桌子的预定情况

Table
- Boolean available - Int capacity - List<Timeslot> reservations
+ boolean isAvailable() + void markUnavailable() + void markAvailable() + get capacity()

Any problems?

- How to know if a table is open for reservation for a timeslot?

把时间点转换为时间段

Table
- Boolean available - Int capacity - List<TimePeriod> reservations
+ boolean isAvailable() + void markUnavailable() + void markAvailable() + get capacity()

Restaurant

Party

- Int capacity

+ int getCapacity()

Table

- Boolean available
- Int capacity

+ boolean isAvailable()
+ void markUnavailable()
+ void markAvailable()
+ get capacity()

Order

- List<Meal> meals
- Table t
- Party p

+ float getPrice()

Meal

- Float price

+ float getPrice()

Restaurant

- List<Table> tables
- List<Meal> menu
- Map<Table, List<Order>> orders

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)
+ Reservation findTableForReservation(Timeslot t)
+ void confirmReservation(Reservation r)

Reservation

- Table table
- Timeslot timeslot

NoTableException

NoTableForReservationExceptio n

Use cases

Find table

Take order

Checkout

Restaurant

Party

- Int capacity

+ int getCapacity()

TimePeroid

- Time start
- Time end

Table

- Boolean available
- Int capacity

+ boolean isAvailable()
+ void markUnavailable()
+ void markAvailable()
+ get capacity()

Order

- List<Meal> meals
- Table t
- Party p

+ float getPrice()

Meal

- Float price

+ float getPrice()

Restaurant

- List<Table> tables
- List<Meal> menu
- Map<Table, List<Order>> orders

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)
+ Reservation findTableForReservation(Timeslot t)
+ void confirmReservation(Reservation r)

Reservation

- Table table
- Timeslot timeslot

NoTableException

NoTableForReservationExceptio n

Use cases

Find table

Take order

Checkout

Restaurant

九章算法

Party

- Int capacity

+ int getCapacity()

TimePeroid

- Time start
- Time end

Table

- Boolean available
- Int capacity

+ boolean isAvailable()
+ void markUnavailable()
+ void markAvailable()
+ get capacity()

Order

- List<Meal> meals
- Table t
- Party p

+ float getPrice()

Meal

- Float price

+ float getPrice()

Restaurant

- List<Table> tables
- List<Meal> menu
- Map<Table, List<Order>> orders

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)
+ Reservation findTableForReservation(Timeslot t)
+ void confirmReservation(Reservation r)

Reservation

- Table table
- TimePeroid timePeriod

NoTableException

NoTableForReservationExceptio n

Use cases

Find table

Take order

Checkout

Restaurant

Party

- Int capacity

+ int getCapacity()

TimePeroid

- Time start
- Time end

Table

- Boolean available
- Int capacity
- List<TimePeroid> reservations

+ boolean isAvailable()
+ void markUnavailable()
+ void markAvailable()
+ get capacity()

Order

- List<Meal> meals
- Table t
- Party p

+ float getPrice()

Meal

- Float price

+ float getPrice()

Restaurant

- List<Table> tables
- List<Meal> menu
- Map<Table, List<Order>> orders

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)
+ Reservation findTableForReservation(Timeslot t)
+ void confirmReservation(Reservation r)

Reservation

- Table table
- TimePeroid timePeriod

NoTableException

NoTableForReservationExceptio n

Use cases

Find table

Take order

Checkout

Restaurant

Party

- Int capacity

+ int getCapacity()

TimePeroid

- Time start
- Time end

Table

- Boolean available
- Int capacity

+ boolean isAvailable()
+ void markUnavailable()
+ void markAvailable()
+ get capacity()

Order

- List<Meal> meals
- Table t
- Party p

+ float getPrice()

Meal

- Float price

+ float getPrice()

Restaurant

- List<Table> tables
- List<Meal> menu
- Map<Table, List<Order>> orders
- Map<Table, List<TimePeroid>> reservation

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)
+ Reservation findTableForReservation(Timeslot t)
+ void confirmReservation(Reservation r)

NoTableException

NoTableForReservationExceptio n

Reservation

- Table table
- TimePeroid timePeroid

Use cases

Find table

Take order

Checkout

Cancel reservation

- Restaurant takes a cancel reservation request, and ask the table to free that timeslot.

Restaurant

Party

- Int capacity

+ int getCapacity()

TimePeroid

- Time start
- Time end

Table

- Boolean available
- Int capacity

+ boolean isAvailable()
+ void markUnavailable()
+ void markAvailable()
+ get capacity()

Order

- List<Meal> meals
- Table t
- Party p

+ float getPrice()

Meal

- Float price

+ float getPrice()

Restaurant

- List<Table> tables
- List<Meal> menu
- Map<Table, List<Order>> orders
- Map<Table, List<TimePeroid>> reservation

+ Table findTable()
+ void takeOrder(Order o)
+ void checkout(Order o)
+ Reservation findTableForReservation(Timeslot t)
+ void confirmReservation(Reservation r)
+ void cancelReservation(Reservation r)

NoTableException

NoTableForReservationExceptio n

Reservation

- Table table
- TimePeroid timePeroid

Use cases

Find table

Take order

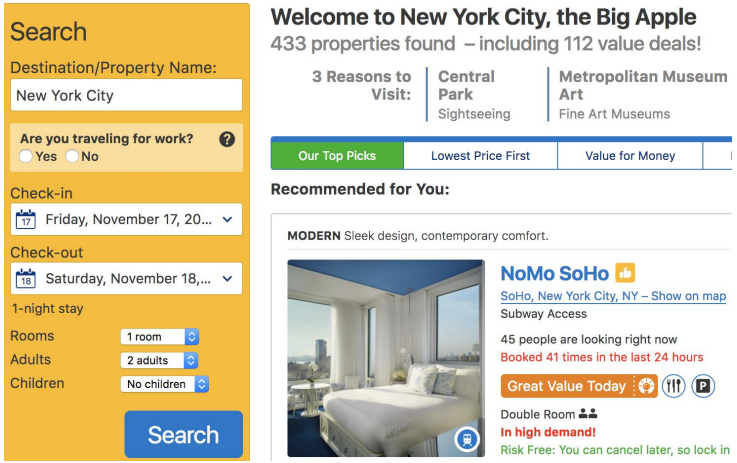
Checkout

Hotel reservation system

- Can you design a hotel reservation system?

Hotel reservation system

- Can you design a hotel reservation system?



VS.

Clarify

- What

是为一间酒店设计预定房间系统，还是先选择酒店的系统？

Design上会有哪些区别？

Clarify

- What

是为一间酒店设计预定房间系统，还是先选择酒店的系统？

- 搜索条件区别

人数 + 时间 VS. 人数 + 时间 + 地址

Clarify

- What

是为一间酒店设计预定房间系统，还是先选择酒店的系统？

- 返回结果区别

Rooms VS. Hotels

Clarify

九章算法

- What

是为一间酒店设计预定房间系统，还是先选择酒店的系统？

- 针对本题：

先设计一间酒店，再设计选择酒店的系统

Clarify

- What

Search criteria -> Search() -> List<Result> -> Select() -> Receipt

Clarify

- What

Search criteria -> Search() -> List<Result> -> Select() -> Receipt

除了考虑题目中的名词之外，还需要从上述的三处考虑，What类型的提问主要针对List<Result>

Clarify

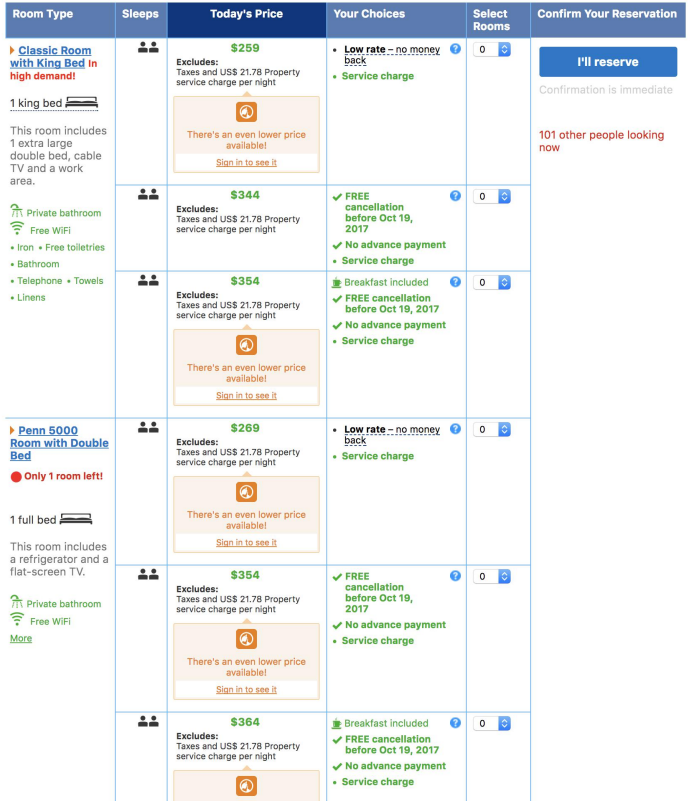
- What

关键字：Room

Clarify

- What

关键字：Room



Clarify

- What

针对本题：房间的人数和价格可能会不同

Challenge

针对本题：房间的人数和价格可能会不同。

如何设计房间类？

Challenge

如何设计房间类？

Room
- int capacity - float price

Challenge

如何设计房间类？

Room
- int capacity - float price

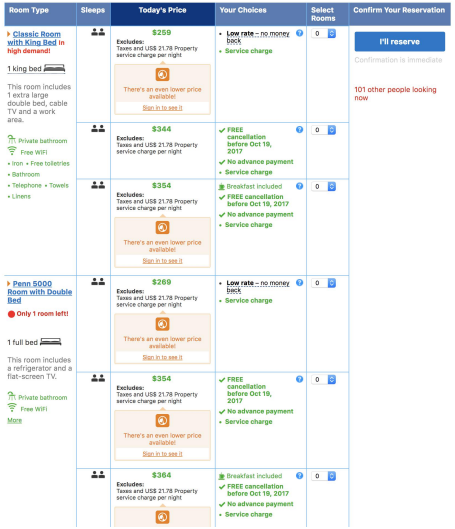
List<Result> -> List<Room>

Room_1	Room_2	Room_3	Room_4	Room_5	Room_6
Capacity : 2 Price: 198	Capacity : 2 Price: 198	Capacity : 2 Price: 198	Capacity : 2 Price: 198	Capacity : 1 Price: 128	Capacity : 1 Price: 128

Challenge

如何设计房间类？

Room
- int capacity - float price



List<Result> -> List<Room>

Room_1
Capacity : 2 Price: 198

Room_2
Capacity : 2 Price: 198

Room_3
Capacity : 2 Price: 198

Room_4
Capacity : 2 Price: 198

Room_5
Capacity : 1 Price: 128

Room_6
Capacity : 1 Price: 128

Clarify

- How

规则？

Clarify

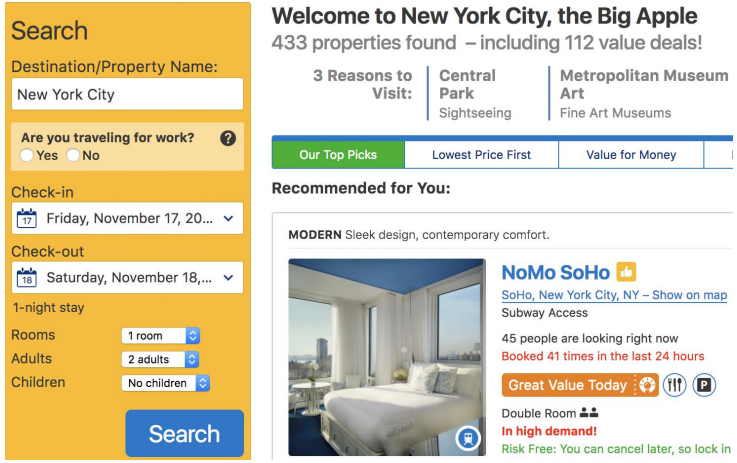
- 预定类规则：

Search criteria

Clarify

- 预定类规则：

Search criteria



Core object

Core object

Hotel

Core object

Request

Hotel

Core object

Request

Hotel

RoomType

Core object

Request

Hotel

RoomType

Room

Core object

Request

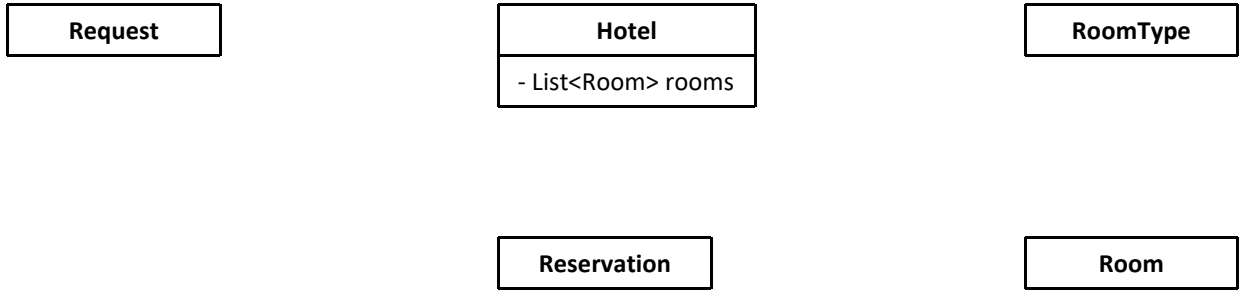
Hotel

RoomType

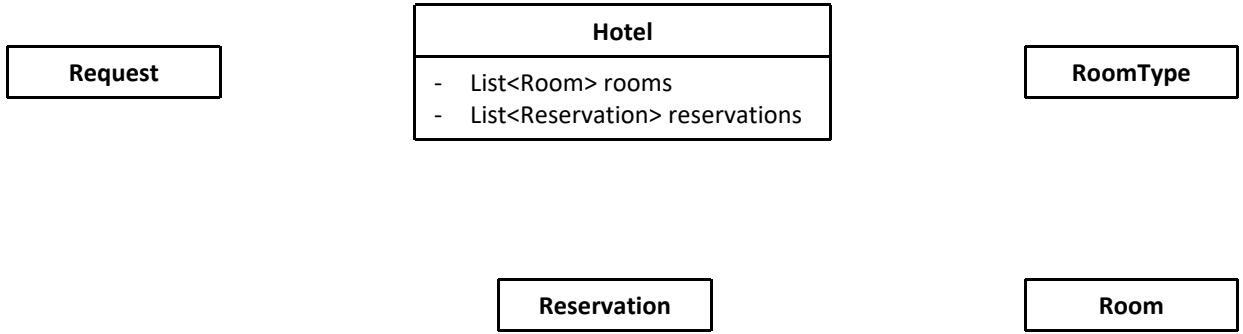
Reservation

Room

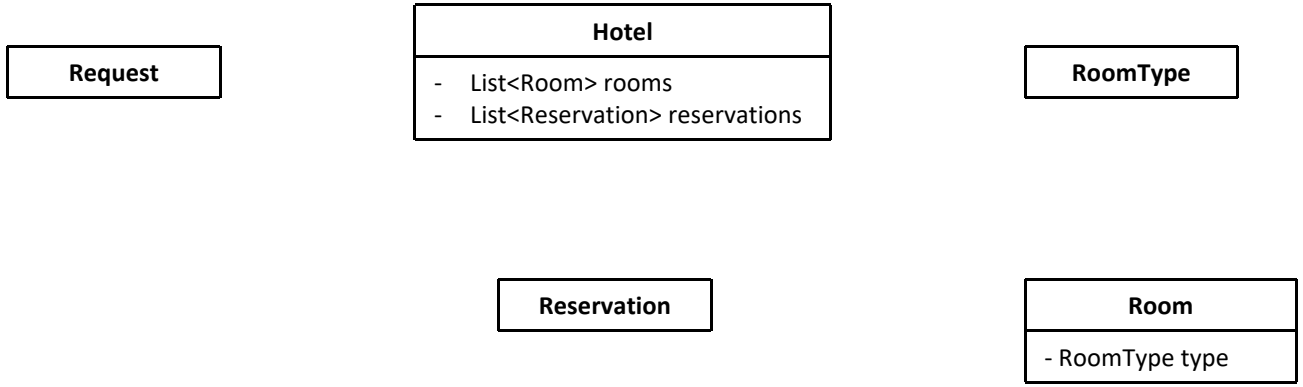
Core object



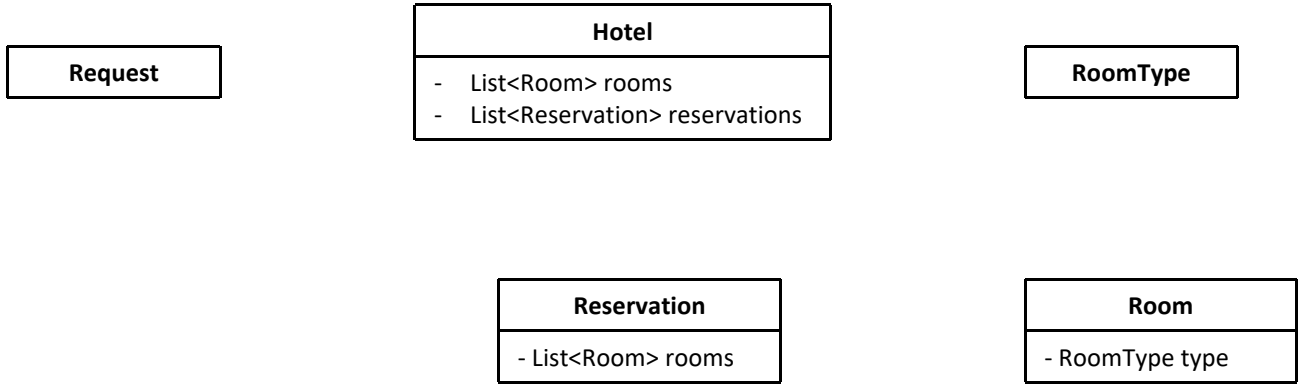
Core object



Core object



Core object



Use cases

Hotel

Use cases

Hotel:

- Search for available rooms

Use cases

Hotel:

- Search for available rooms
- Make reservation

Use cases

Hotel:

- Search for available rooms
- Make reservation
- Cancel reservation

Classes

九章算法

Request

Hotel

- List<Room> rooms
- List<Reservation> reservations

RoomType

Room

- RoomType type

Reservation

- List<Room> rooms

Use cases

Search for available rooms

Make reservation

Cancel reservation

Use cases

Search for available rooms

1: Based on search criteria
2: Go through rooms to check availability
3: list available room types and available count

Use cases

Search for available rooms

1: Based on search criteria

Classes

Request
- Date startDate

Hotel
- List<Room> rooms - List<Reservation> reservations

RoomType

Room
- RoomType type

Reservation
- List<Room> rooms

Use cases
Search for available rooms
Make reservation
Cancel reservation

Classes

九章算法

Request
- Date startDate - Date endDate

Hotel
- List<Room> rooms - List<Reservation> reservations

RoomType

Room
- RoomType type

Reservation
- List<Room> rooms

Use cases
Search for available rooms
Make reservation
Cancel reservation

Classes

Request
- Date startDate - Date endDate

Hotel
- List<Room> rooms - List<Reservation> reservations
+ Map<RoomType, int> handleSearchRequest(Request r)

RoomType

Room
- RoomType type

Reservation
- List<Room> rooms

Use cases
Search for available rooms
Make reservation
Cancel reservation

Use cases

Search for available rooms

2: Go through rooms to check availability

Classes

九章算法

Request
- Date startDate - Date endDate

Hotel
- List<Room> rooms - List<Reservation> reservations
+ Map<RoomType, int> handleSearchRequest(Request r)

RoomType

Room
- RoomType type - List<Date> reservations

Reservation
- List<Room> rooms

Use cases
Search for available rooms
Make reservation
Cancel reservation

Classes

九章算法

Request
- Date startDate - Date endDate

Hotel
- List<Room> rooms - List<Reservation> reservations - Map<Room, List<Date>> roomReservations
+ Map<RoomType, int> handleSearchRequest(Request r)

RoomType

Room
- RoomType type

Reservation
- List<Room> rooms

Use cases
Search for available rooms
Make reservation
Cancel reservation

Classes

九章算法

Request
- Date startDate - Date endDate

Hotel
- List<Room> rooms - List<Reservation> reservations - Map<Room, List<Date>> roomReservations
+ Map<RoomType, int> handleSearchRequest(Request r) - boolean isRequestAvailable(Request r, List<Date> dates)

RoomType

Room
- RoomType type

Reservation
- List<Room> rooms

Use cases
Search for available rooms
Make reservation
Cancel reservation

Classes

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)

RoomType

Room

- RoomType type
- Boolean available

Reservation

- List<Room> rooms

Use cases

Search for available rooms

Make reservation

Cancel reservation

Classes

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)

RoomType

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms

Use cases

Search for available rooms

Make reservation

Cancel reservation

Use cases

Search for available rooms

3: list available room types and counts

Classes

九章算法

Request
- Date startDate - Date endDate

Hotel
- List<Room> rooms - List<Reservation> reservations - Map<Room, List<Date>> roomReservations
+ Map<RoomType, int> handleSearchRequest(Request r) - boolean isRequestAvailable(Request r, List<Date> dates)

RoomType
SINGLE DOUBLE

Room
- RoomType type - Boolean available
+ boolean isAvailable()

Reservation
- List<Room> rooms

Use cases
Search for available rooms
Make reservation
Cancel reservation

Classes

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms

Use cases

Search for available rooms
Make reservation
Cancel reservation

Use cases

Make reservation

1: Add RoomType and number of rooms in a request
2: Send request to Hotel
3: If there is enough room left, confirm the reservation
4: If there isn’t enough room left, throw exception

Use cases

Make reservation

1: Add RoomType and number of rooms in a request

Classes

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms

Use cases

Search for available rooms
Make reservation
Cancel reservation

Classes

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms

ReservationRequest

Use cases

Search for available rooms
Make reservation
Cancel reservation

Classes

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms

ReservationRequest

- Date startDate
- Date endDate

Use cases

Search for available rooms
Make reservation
Cancel reservation

Classes

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

Use cases

Search for available rooms
Make reservation
Cancel reservation

Use cases

Make reservation

2: Send request to Hotel

Classes

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

Use cases

Search for available rooms
Make reservation
Cancel reservation

Classes

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

Use cases

Search for available rooms
Make reservation
Cancel reservation

Use cases

Make reservation

3: If there is enough room left, confirm the reservation

Classes

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms
- Date startDate
- Date endDate

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

Use cases

Search for available rooms
Make reservation
Cancel reservation

Use cases

Make reservation

4: If there isn’t enough room left, throw exception

Classes

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms
- Date startDate
- Date endDate

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

NotEnoughRoomForReservationException

Use cases

Search for available rooms
Make reservation
Cancel reservation

Challenge

九章算法

Search for available rooms

1: Based on search criteria
2: Go through rooms to check availability
3: list available room types and room count

Make reservation

1: Add RoomType and number of rooms in a request
2: Send request to Hotel
3: If there is enough room left, confirm the reservation
4: If there isn't enough room left, throw exception

Challenge

```java
Map<RoomType, List<Room>> map = new HashMap<>();

for(Entry<Room, List<Date>> entry : roomReservations.entrySet())
{
    Room r = entry.getKey();
    List<Date> roomBooked = entry.getValue();

    if(isRequestAvailable(roomBooked))
    {
        if(map.containsKey(r.getRoomType()))
        {
            List<Room> roomList = map.get(r.getRoomType());
            roomList.add(r);
            map.put(r.getRoomType(), roomList);
        }
        else
        {
            List<Room> roomList = new ArrayList<>();
            roomList.add(r);
            map.put(r.getRoomType(), roomList);
        }
    }
}
```

Challenge

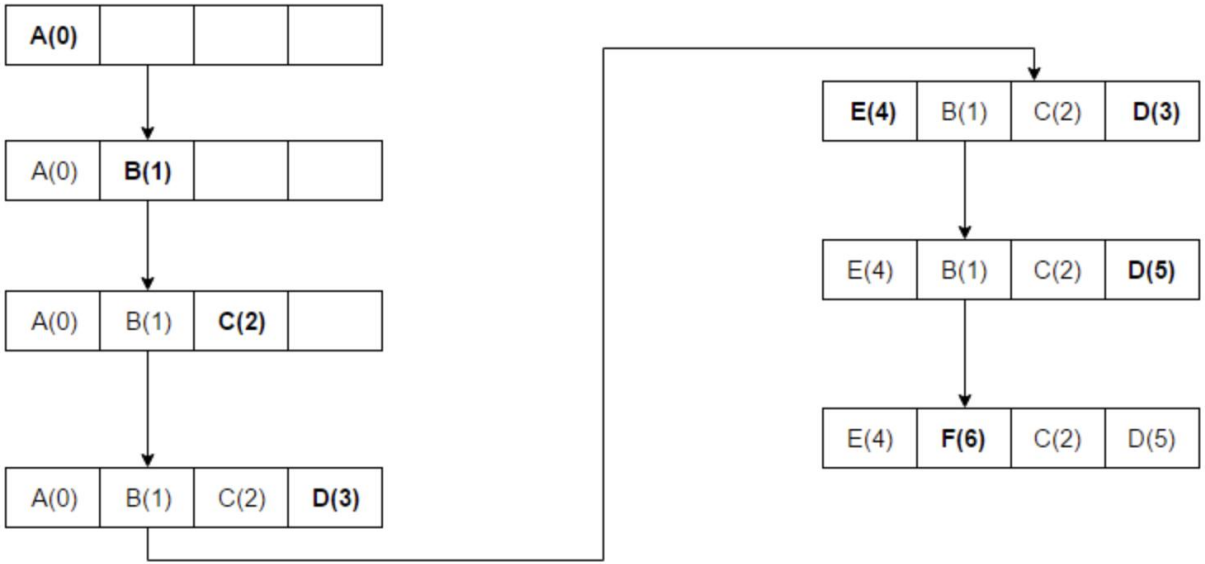
- Map<RoomType, List<Room>> map

- Go through rooms to check availability
- If there is enough room left, confirm the reservation
- If there isn’t enough room left, throw exception

Challenge

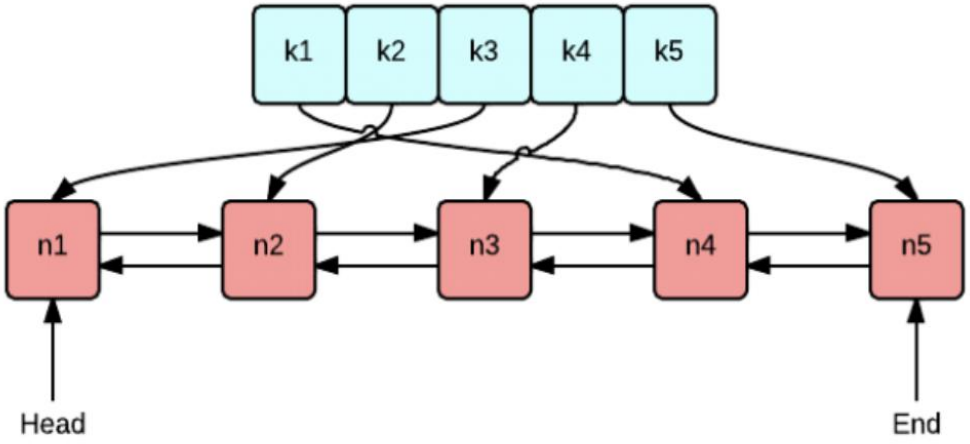
- LRU Cache

The access sequence for the below example is A B C D E D F.



Challenge

- LRU Cache



Challenge

九章算法

```java
class LRUCache extends LinkedHashMap<Request, Map<RoomType, List<Room>>>
{
    private int capacity;

    public LRUCache(int capacity)
    {
        super(capacity);
        this.capacity = capacity;
    }

   @Override
   protected boolean removeEldestEntry(Map.Entry<Request, Map<RoomType, List<Room>>> eldest){
      return size() > this.capacity;
   }
}
```

Classes

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms
- Date startDate
- Date endDate

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

NotEnoughRoomForReservationException

Use cases

Search for available rooms
Make reservation
Cancel reservation

Use cases

Cancel reservation

1: Hotel system takes a reservation, and cancel it.

Classes

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms
- Date startDate
- Date endDate

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

NotEnoughRoomForReservationException

Use cases

Search for available rooms

Make reservation

Cancel reservation

Classes – Final view

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms
- Date startDate
- Date endDate

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

NotEnoughRoomForReservationException

Use cases

Search for available rooms

Make reservation

Cancel reservation

Challenge

面试官：

Can you extend your design to a booking.com like system?

Challenge

Search criteria -> Search() -> List<Result> -> Select() -> Receipt

Challenge

Search criteria -> Search() -> List<Result> -> Select() -> Receipt

Search criteria A -> Search() -> List<Result_A> -> Select() -> Search criteria B -> Search() -> List<Result_B> -> Select() -> Receipt

Challenge

Search criteria -> Search() -> List<Result> -> Select() -> Receipt

Search criteria A -> Search() -> List<Result_A> -> Select() -> Search criteria B -> Search() -> List<Result_B> -> Select() -> Receipt

Search criteria A : 10/30 -11/5 Boston

Challenge

Search criteria -> Search() -> List<Result> -> Select() -> Receipt

Search criteria A -> Search() -> List<Result_A> -> Select() -> Search criteria B -> Search() -> List<Result_B> -> Select() -> Receipt

Search criteria A : 10/30 -11/5 Boston

BookingSystem.Search()

Challenge

Search criteria -> Search() -> List<Result> -> Select() -> Receipt

Search criteria A -> Search() -> List<Result_A> -> Select() -> Search criteria B -> Search() -> List<Result_B> -> Select() -> Receipt

Search criteria A : 10/30 -11/5 Boston

BookingSystem.Search()

List<Result_A>: Hotel_1, Hotel_2,…

Challenge

Search criteria -> Search() -> List<Result> -> Select() -> Receipt

Search criteria A -> Search() -> List<Result_A> -> Select() -> Search criteria B -> Search() -> List<Result_B> -> Select() -> Receipt

Search criteria A : 10/30 -11/5 Boston

BookingSystem.Search()

List<Result_A>: Hotel_1, Hotel_2,…

BookingSystem.Select()

Challenge

Search criteria -> Search() -> List<Result> -> Select() -> Receipt

Search criteria A -> Search() -> List<Result_A> -> Select() -> Search criteria B -> Search() -> List<Result_B> -> Select() -> Receipt

Search criteria A : 10/30 -11/5 Boston

Search criteria B : 10/30 -11/5

BookingSystem.Search()

List<Result_A>: Hotel_1, Hotel_2,…

BookingSystem.Select()

Challenge

Search criteria -> Search() -> List<Result> -> Select() -> Receipt

Search criteria A -> Search() -> List<Result_A> -> Select() -> Search criteria B -> Search() -> List<Result_B> -> Select() -> Receipt

Search criteria A : 10/30 -11/5 Boston

BookingSystem.Search()

List<Result_A>: Hotel_1, Hotel_2,…

BookingSystem.Select()

Search criteria B : 10/30 -11/5

Hotel_1.Search()

Challenge

Search criteria -> Search() -> List<Result> -> Select() -> Receipt

Search criteria A -> Search() -> List<Result_A> -> Select() -> Search criteria B -> Search() -> List<Result_B> -> Select() -> Receipt

Search criteria A : 10/30 -11/5 Boston

BookingSystem.Search()

List<Result_A>: Hotel_1, Hotel_2,…

BookingSystem.Select()

Search criteria B : 10/30 -11/5

Hotel_1.Search()

List<Result_A>: RoomType_1, …

Challenge

Search criteria -> Search() -> List<Result> -> Select() -> Receipt

Search criteria A -> Search() -> List<Result_A> -> Select() -> Search criteria B -> Search() -> List<Result_B> -> Select() -> Receipt

Search criteria A : 10/30 -11/5 Boston

BookingSystem.Search()

List<Result_A>: Hotel_1, Hotel_2,…

BookingSystem.Select()

Search criteria B : 10/30 -11/5

Hotel_1.Search()

List<Result_A>: RoomType_1, …

BookingSystem.Select()

Clarify

What

Clarify

What

Search criteria A

List<Result_A>

Search criteria B

List<Result_B>

Receipt

Clarify

What

Search criteria A

List<Result_A>

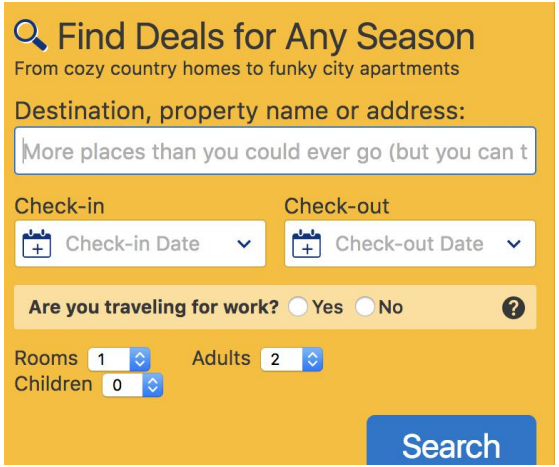
Search criteria B

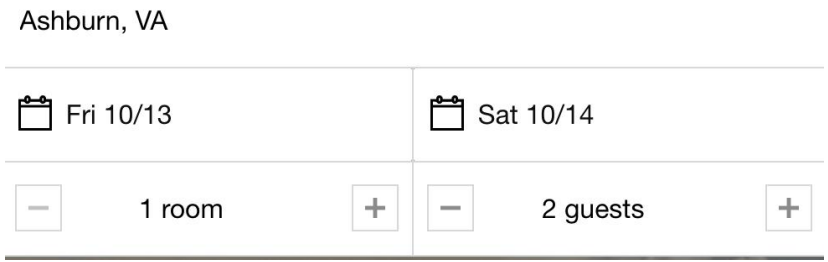
List<Result_B>

Receipt

Clarify

What





What are the search criteria for booking system ?

Clarify

针对本题：

用Start Date, End Date, Group size, City

Clarify

How

Search()

Select()

Search()

Select()

Clarify

How

Search()

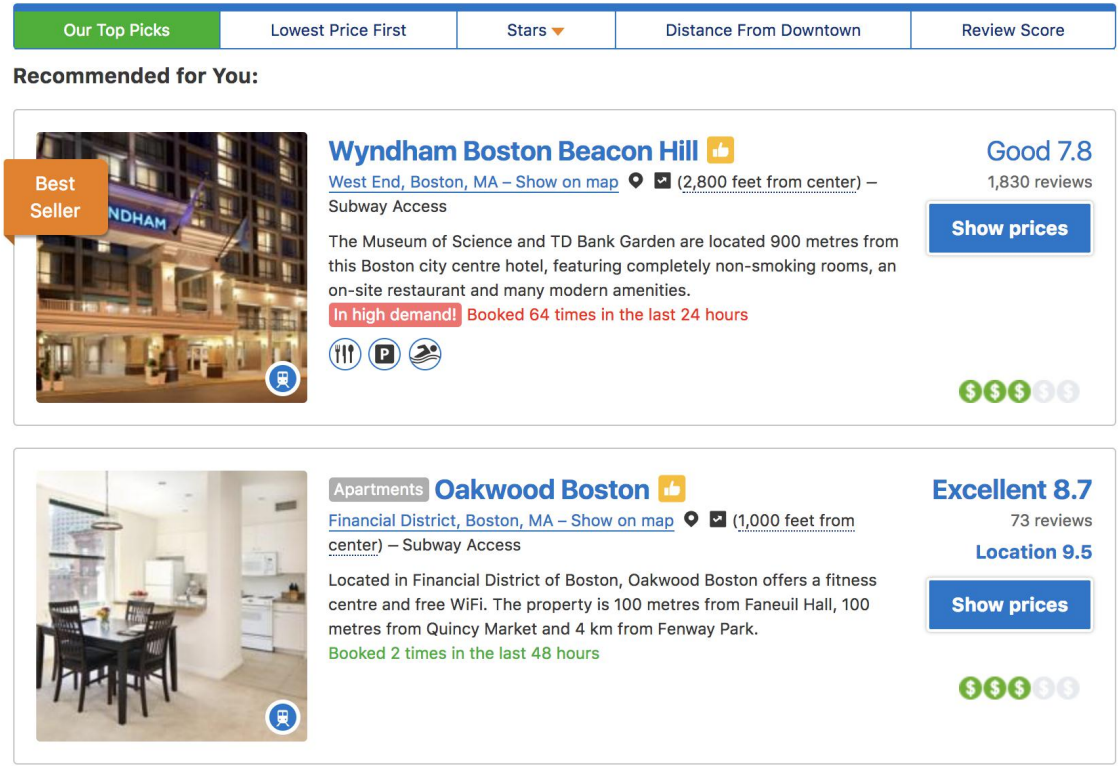
Select()

Search()

Select()

Clarify

How



- 哪些Hotel可以被放进List里？
- 按照什么顺序排序？

Clarify

九章算法

How

针对本题：

- 在同样的City就能放进List里
- Hotel在List里的顺序不重要

Core Object

九章算法

How

针对本题：

- 在同样的City就能放进List里
- Hotel在List里的顺序不重要

Core object

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms
- Date startDate
- Date endDate

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

NotEnoughRoomForReservationException

Use cases

Search for available rooms
Make reservation
Cancel reservation

Core object

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

NotEnoughRoomForReservationException

Use cases

Search for available rooms

Make reservation

Cancel reservation

Core object

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

NotEnoughRoomForReservationException

Use cases

Search for available rooms

Make reservation

Cancel reservation

Core object

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

NotEnoughRoomForReservationException

Use cases

Search for available rooms
Make reservation
Cancel reservation

Use cases

- BookingSystem

Use cases

- BookingSystem

- Search for hotels

Use cases

- BookingSystem

- Search for hotels
- Make reservation

Use cases

- BookingSystem

- Search for hotels
- Make reservation
- Cancel reservation

Class

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

NotEnoughRoomForReservationException

Use cases

- Search for available rooms
- Make reservation
- Cancel reservation
- Search for hotels
- Make reservation (Booking System)
- Cancel reservation (Booking System)

Classes

- Search for hotels

- Based on request (start date + end date + city + group size)
- Check for all hotels in this city
- For such hotels, if this is enough capacity for group size during dates
- List all hotels that satisfy above criteria

Classes

- Search for hotels

- Based on request (start date + end date + city + group size)

Class

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

NotEnoughRoomForReservationException

Use cases

- Search for available rooms
- Make reservation
- Cancel reservation
- Search for hotels
- Make reservation (Booking System)
- Cancel reservation (Booking System)

Classes

- Search for hotels

- Based on request (start date + end date + city + group size)
- Check for all hotels in this city

Class

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache
- String city

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

NotEnoughRoomForReservationException

Use cases

Search for available rooms
Make reservation
Cancel reservation
Search for hotels
Make reservation (Booking System)
Cancel reservation (Booking System)

Class

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache
- String city

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)
+ boolean isSameCity(string city)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

NotEnoughRoomForReservationException

Use cases

- Search for available rooms
- Make reservation
- Cancel reservation
- Search for hotels
- Make reservation (Booking System)
- Cancel reservation (Booking System)

Classes

- Search for hotels

- Based on request (start date + end date + city + group size)
- Check for all hotels in this city
- For such hotels, if this is enough capacity for group size during dates

Class

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache
- String city

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)
+ boolean isSameCity(string city)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

- Int capacity

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

NotEnoughRoomForReservationException

Use cases

- Search for available rooms
- Make reservation
- Cancel reservation
- Search for hotels
- Make reservation (Booking System)
- Cancel reservation (Booking System)

Class

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache
- String city

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)
+ boolean isSameCity(string city)
+ boolean isValidRequest(SearchHotelRequest r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

- Int capacity

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

NotEnoughRoomForReservationException

Use cases

Search for available rooms
Make reservation
Cancel reservation
Search for hotels
Make reservation (Booking System)
Cancel reservation (Booking System)

Classes

- Search for hotels

- Based on request (start date + end date + city + group size)
- Check for all hotels in this city
- For such hotels, if this is enough capacity for group size during dates
- List all hotels that satisfy above criteria

Class

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache
- String city

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)
+ boolean isSameCity(string city)
+ boolean isValidRequest(SearchHotelRequest r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

- Int capacity

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels

+ List<Hotel> searchHotel(SearchHotelRequest r)

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

NotEnoughRoomForReservationException

Use cases

- Search for available rooms
- Make reservation
- Cancel reservation
- Search for hotels
- Make reservation (Booking System)
- Cancel reservation (Booking System)

Use case

- Make reservation

- Pick a hotel
- Send reservation request to that hotel

Class

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache
- String city

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)
+ boolean isSameCity(string city)
+ boolean isValidRequest(SearchHotelRequest r)

<<Enumeration>> RoomType

SINGLE
DOUBLE

- Int capacity

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels

+ List<Hotel> searchHotel(SearchHotelRequest r)
+ Reservation makeReservation(Hotel h, ReservationRequest r)

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

NotEnoughRoomForReservationException

Use cases

- Search for available rooms
- Make reservation
- Cancel reservation
- Search for hotels
- Make reservation (Booking System)
- Cancel reservation (Booking System)

Classes

- Cancel reservation

- Cancel reservation for a hotel

Class

九章算法

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache
- String city

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)
+ boolean isSameCity(string city)
+ boolean isValidRequest(SearchHotelRequest r)

<<Enumeration>> RoomType

SINGLE
DOUBLE

- Int capacity

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- Hotel hotel
- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels

+ List<Hotel> searchHotel(SearchHotelRequest r)
+ Reservation makeReservation(Hotel h, ReservationRequest r)
+ void cancelReservation(Reservation r)

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

NotEnoughRoomForReservationException

Use cases

Search for available rooms
Make reservation
Cancel reservation
Search for hotels
Make reservation (Booking System)
Cancel reservation (Booking System)

Class – Final view

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache
- String city

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)
+ boolean isSameCity(string city)
+ boolean isValidRequest(SearchHotelRequest r)

<<Enumeration>> RoomType

SINGLE
DOUBLE

- Int capacity

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- Hotel hotel
- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels

+ List<Hotel> searchHotel(SearchHotelRequest r)
+ Reservation makeReservation(Hotel h, ReservationRequest r)
+ void cancelReservation(Reservation r)

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

NotEnoughRoomForReservationException

Use cases

Search for available rooms
Make reservation
Cancel reservation
Search for hotels
Make reservation (Booking System)
Cancel reservation (Booking System)

Challenge

- Can you add payment in your system?

Challenge

- Can you add payment in your system?

- Need to get price for each reservation
- Need to take a payment method

Challenge

- Can you add payment in your system?

- Need to get price for each reservation

Class

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache
- String city

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)
+ boolean isSameCity(string city)
+ boolean isValidRequest(SearchHotelRequest r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

- Int capacity
- Float price

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- Hotel hotel
- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels

+ List<Hotel> searchHotel(SearchHotelRequest r)
+ Reservation makeReservation(Hotel h, ReservationRequest r)
+ void cancelReservation(Reservation r)

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

NotEnoughRoomForReservationException

Use cases

Search for available rooms
Make reservation
Cancel reservation
Search for hotels
Make reservation (Booking System)
Cancel reservation (Booking System)

Enum code sample

九章算法

```
public enum RoomType
{
    SINGLE(1, 129),
    DOUBLE(2, 199);

    private int capacity;
    private float price;

    RoomType(int capacity, float price)
    {
        this.capacity = capacity;
        this.price = price;
    }

    public int getCapacity()
    {
        return capacity;
    }

    public float getPrice()
    {
        return price;
    }
}
```

Class

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache
- String city

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)
+ boolean isSameCity(string city)
+ boolean isValidRequest(SearchHotelRequest r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

- Int capacity
- Float price

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- Hotel hotel
- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels

+ List<Hotel> searchHotel(SearchHotelRequest r)
+ Reservation makeReservation(Hotel h, ReservationRequest r)
+ void cancelReservation(Reservation r)
+ void payByPaypal (Reservation r)

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

NotEnoughRoomForReservationException

Use cases

Search for available rooms
Make reservation
Cancel reservation
Search for hotels
Make reservation (Booking System)
Cancel reservation (Booking System)

Class

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache
- String city

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)
+ boolean isSameCity(string city)
+ boolean isValidRequest(SearchHotelRequest r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

- Int capacity
- Float price

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- Hotel hotel
- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels

+ List<Hotel> searchHotel(SearchHotelRequest r)
+ Reservation makeReservation(Hotel h, ReservationRequest r)
+ void cancelReservation(Reservation r)
+ void payByPaypal (Reservation r)
+ void payByCreditCard (Reservation r)

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

NotEnoughRoomForReservationException

Use cases

Search for available rooms
Make reservation
Cancel reservation
Search for hotels
Make reservation (Booking System)
Cancel reservation (Booking System)

Good practice:

- Strategy Pattern

Class

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache
- String city

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)
+ boolean isSameCity(string city)
+ boolean isValidRequest(SearchHotelRequest r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

- Int capacity
- Float price

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- Hotel hotel
- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels

+ List<Hotel> searchHotel(SearchHotelRequest r)
+ Reservation makeReservation(Hotel h, ReservationRequest r)
+ void cancelReservation(Reservation r)
+ void payByPaypal (Reservation r)
+ void payByCreditCard (Reservation r)

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

NotEnoughRoomForReservationException

Use cases

Search for available rooms
Make reservation
Cancel reservation
Search for hotels
Make reservation (Booking System)
Cancel reservation (Booking System)

Class

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache
- String city

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)
+ boolean isSameCity(string city)
+ boolean isValidRequest(SearchHotelRequest r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

- Int capacity
- Float price

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- Hotel hotel
- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels

+ List<Hotel> searchHotel(SearchHotelRequest r)
+ Reservation makeReservation(Hotel h, ReservationRequest r)
+ void cancelReservation(Reservation r)

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

<<interface>>
PaymentStrategy

NotEnoughRoomForReservationException

Use cases

Search for available rooms
Make reservation
Cancel reservation
Search for hotels
Make reservation (Booking System)
Cancel reservation (Booking System)

Class

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache
- String city

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)
+ boolean isSameCity(string city)
+ boolean isValidRequest(SearchHotelRequest r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

- Int capacity
- Float price

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- Hotel hotel
- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels

+ List<Hotel> searchHotel(SearchHotelRequest r)
+ Reservation makeReservation(Hotel h, ReservationRequest r)
+ void cancelReservation(Reservation r)

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

<<interface>>
PaymentStrategy

+ void pay(Reservation r)

Use cases

Search for available rooms
Make reservation
Cancel reservation
Search for hotels
Make reservation (Booking System)
Cancel reservation (Booking System)

NotEnoughRoomForReservationException

Class

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache
- String city

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)
+ boolean isSameCity(string city)
+ boolean isValidRequest(SearchHotelRequest r)

<<Enumeration>>
RoomType

SINGLE
DOUBLE

- Int capacity
- Float price

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- Hotel hotel
- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels
- PaymentStrategy strategy

+ List<Hotel> searchHotel(SearchHotelRequest r)
+ Reservation makeReservation(Hotel h, ReservationRequest r)
+ void cancelReservation(Reservation r)

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

<<interface>>
PaymentStrategy

+ void pay(Reservation r)

Use cases

Search for available rooms
Make reservation
Cancel reservation
Search for hotels
Make reservation (Booking System)
Cancel reservation (Booking System)

NotEnoughRoomForReservationException

Class

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache
- String city

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)
+ boolean isSameCity(string city)
+ boolean isValidRequest(SearchHotelRequest r)

<<Enumeration>> RoomType

SINGLE
DOUBLE

- Int capacity
- Float price

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- Hotel hotel
- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels
- PaymentStrategy strategy

+ List<Hotel> searchHotel(SearchHotelRequest r)
+ Reservation makeReservation(Hotel h, ReservationRequest r)
+ void cancelReservation(Reservation r)
+ void setStrategy(Payment strategy)

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

<<interface>> PaymentStrategy

+ void pay(Reservation r)

NotEnoughRoomForReservationException

Use cases

- Search for available rooms
- Make reservation
- Cancel reservation
- Search for hotels
- Make reservation (Booking System)
- Cancel reservation (Booking System)

Class

Request

- Date startDate
- Date endDate

Hotel

- List<Room> rooms
- List<Reservation> reservations
- Map<Room, List<Date>> roomReservations
- LRUCache<Request, Map<RoomType, Set<Room>>> cache
- String city

+ Map<RoomType, int> handleSearchRequest(Request r)
- boolean isRequestAvailable(Request r, List<Date> dates)
+ Reservation makeReservation(ReservationRequest r)
+ void cancelReservation(Reservation r)
+ boolean isSameCity(string city)
+ boolean isValidRequest(SearchHotelRequest r)

<<Enumeration>> RoomType

SINGLE
DOUBLE

- Int capacity
- Float price

Room

- RoomType type
- Boolean available

+ boolean isAvailable()

Reservation

- Hotel hotel
- List<Room> rooms
- Date startDate
- Date endDate

BookingSystem

- List<Hotel> hotels
- PaymentStrategy strategy

+ List<Hotel> searchHotel(SearchHotelRequest r)
+ Reservation makeReservation(Hotel h, ReservationRequest r)
+ void cancelReservation(Reservation r)
+ void setStrategy(Payment strategy)
+ void makePayment(Reservation r)

ReservationRequest

- Date startDate
- Date endDate
- Map<RoomType, int> roomsNeeded

SearchHotelRequest

- Date startDate
- Date endDate
- String city
- Int groupSize

<<interface>> PaymentStrategy

+ void pay(Reservation r)

Use cases

Search for available rooms
Make reservation
Cancel reservation
Search for hotels
Make reservation (Booking System)
Cancel reservation (Booking System)

NotEnoughRoomForReservationException

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