Assignment 6

Designer: ZHU Yueming
Tester: JIANG Chuan

Description

As you know that for some hotels, they contain different type of rooms. Everyday, several rooms have been checked in and several rooms have been checked out, during the process the hotel also have incomes accordingly. Then you are asked to help a hotel owner to design a room management system. The system is very simple, and it only contains two types of rooms: **Ordinary Room** and **Luxury Room**.

General describe your task

Your task is to design following classes

1. **Room**.
    - Do not modify or remove any methods or fields that have been already defined.
    - You can add methods or fields that you think are necessary.

2. **LuxuryRoom** and **OrdinaryRoom.** Those are the subclasses of the class Room.

    - LuxuryRoom: adding a private data field named **addBed (boolean)** with an initial value **false**
    - OrdinaryRoom: adding a private data field name **breakfastCount (int)** with an initial value **0**.
    - toString method: Override **toString** method in each subclasses. Here the return format must strictly follow the requirement.
    - **checkOut** and **checkIn** method: Override the abstract method in each subclasses.
    - You can add other methods or attributes that you think are necessary.

3. The enum class **Day**. It is an enumeration class that is used to record everyday information including the **name** of the current day, the **rate** of basic price and the **gift** our hotel would given. When you design the enumeration class, you need to use following seven constants directly. However only adding following statements into enum class are not enough, you need to complement all missing code (you think are necessary) in it.

```java
MONDAY("MON", 0.8, "Fruits"),
TUESDAY("TUE", 0.75, "Drinks"),
WEDNESDAY("WED", 0.71, "GYM CARD"),
THURSDAY("THU", 0.68, "Fruits"),
FRIDAY("FRI", 1, "GYM CARD"),
SATURDAY("SAT", 1, "HOT SPRINGS"),
SUNDAY("SUN", 0.95, "SWIMMING");
private String name;
private double rate;
private String gift;
```

4. **ConcreteHotel.** It is a concrete class of the interface Hotel, in which you need to implement all the abstract methods that are declared in the interface Hotel. In

concrete Hotel, the following one important parameter must be defined. Beyond that, you can define any attributes that you think are important.

- **rooms**: It is a List<Room> type, which contains all rooms, including two different types, in concrete hotel.
  ```
  private List<Room> rooms = new ArrayList<>();
  ```

5. Other import parameters:
However, there are several initial parameters needs to mention to you, where to define those parameters are determined to your design.

- The price of **breakfast** for one person in Ordinary Room is **180**.
- The price of **adding a bed** in Luxury Room is **250**.
- The initial **basic price** of Luxury Room is **1200** per night (the breakfasts are included).
- The initial **basic price** of Ordinary Room is **500** per night (adding bed are not allowed)
- The calculation of total price for one room per night is
  For Luxury Room:
  Basic price of Luxury Room * the rate of price in current day + the price of add bed (if need)
  For ordinary Room:
  Basic price of Ordinary Room * the rate of price in current day + the price of breakfast * the number of breakfast.

- The initial value of **day** is **Day.MONDAY**

Your tasks

For the whole project, you needn't input anything. The class Client.java, being composited by several static methods, serves as the sample input. Please read carefully about what we would do in Client (The official test case is another Client.java class, which is similar to this one), and the sample output for those tasks. Other than that, you also need to read carefully about the annotations in Hotel.java, from which you can understand what to do in each abstract method.

```
task001_getRoomInfo();
```

```
task002_getSubClassInfo();
```

Explanation 1: The format of toString() method in class Room
Luxury Room: "L" RoomNumber whether add bed
Ordinary Room: "O" RoomNumber the count of breakfast

```
task01_addLuxuryRoom();
L R1 false
```

```
task02_addOrdinaryRoom();
O R2 0
```

```
task03_addMoreRoom();
L R1 false
O R2 0
L R3 false
L R4 false
L R5 false
L R6 false
L R7 false
O R8 0
O R9 0
O R10 0
O R11 0
O R12 0
O R13 0
O R14 0
O R15 0
O R16 0
O R17 0
L R18 false
L R19 false
```

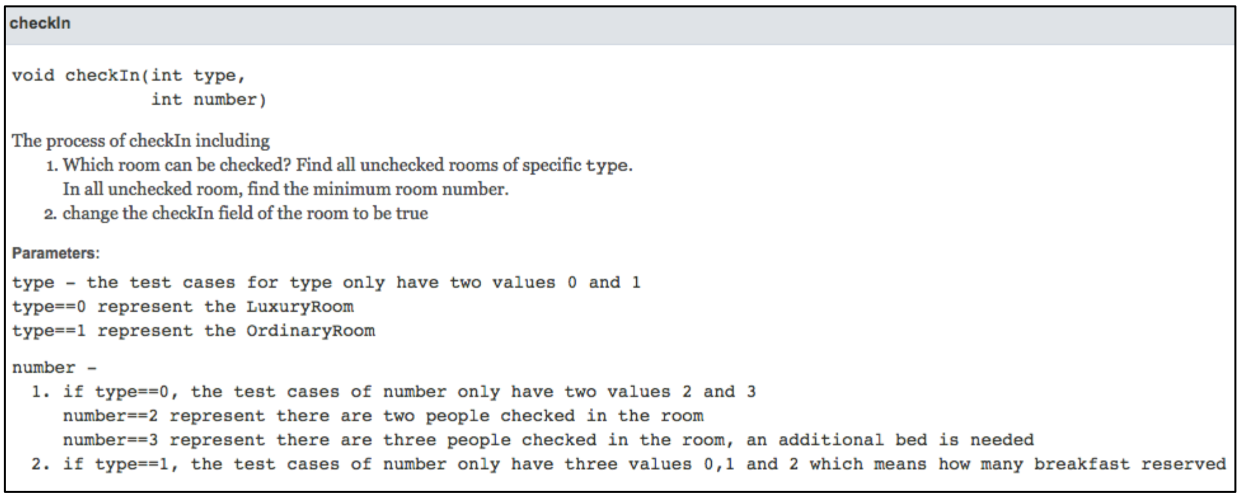
```
task04_getTwoTypeRoomPrice();
1200 500
```

```
task05_setTwoTypeRoomPrice();
1300 600
```

Explanation 2: The format of toString() method in enum Day
Day{name='name', rate='rate', gift='gift'}

```
task06_displayEveryDayInfo();
Day{name='MON', rate=0.8, gift='Fruits'}
Day{name='TUE', rate=0.75, gift='Drinks'}
Day{name='WED', rate=0.71, gift='GYM CARD'}
Day{name='THU', rate=0.68, gift='Fruits'}
Day{name='FRI', rate=1.0, gift='GYM CARD'}
Day{name='SAT', rate=1.0, gift='HOT SPRINGS'}
Day{name='SUN', rate=0.95, gift='SWIMMING'}
```

Explanation 3: Check in process



checkIn

```
void checkIn(int type,
             int number)
```

The process of checkIn including

1. Which room can be checked? Find all unchecked rooms of specific type.
   In all unchecked room, find the minimum room number.
2. change the checkIn field of the room to be true

Parameters:

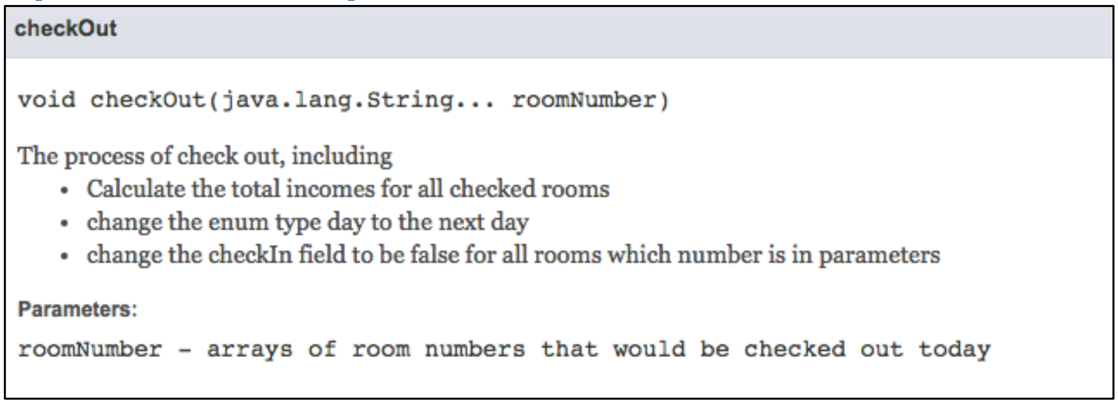
```
type - the test cases for type only have two values 0 and 1
type==0 represent the LuxuryRoom
type==1 represent the OrdinaryRoom

number -
  1. if type==0, the test cases of number only have two values 2 and 3
     number==2 represent there are two people checked in the room
     number==3 represent there are three people checked in the room, an additional bed is needed
  2. if type==1, the test cases of number only have three values 0,1 and 2 which means how many breakfast reserved
```

task07_checkInForFirstDay();

R1 R2 R3 R4 R8 R9

Explanation 4: Check out process



checkOut

```
void checkOut(java.lang.String... roomNumber)
```

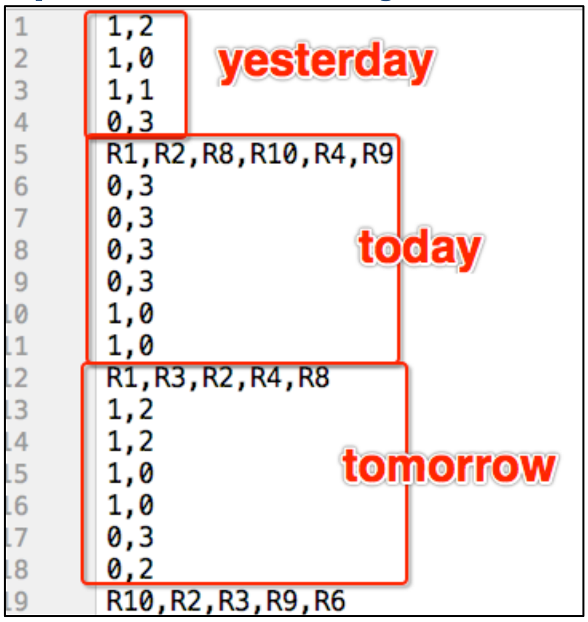
The process of check out, including

- Calculate the total incomes for all checked rooms
- change the enum type day to the next day
- change the checkIn field to be false for all rooms which number is in parameters

Parameters:

```
roomNumber - arrays of room numbers that would be checked out today
```

Explanation 5: How can we get to tomorrow?



```
1,2
1,0
1,1
0,3
R1,R2,R8,R10,R4,R9
0,3
0,3
0,3
0,3
1,0
1,0
R1,R3,R2,R4,R8
1,2
1,2
1,0
1,0
0,3
0,2
R10,R2,R3,R9,R6
```

task08_checkOutForSecondDay();

```
R3 R4 R9

task09_displayYesterdayIncome();
5420

task10_displayTodayInfo();
Day{name='TUE', rate=0.75, gift='Drinks'}

task11_displayTotalPriceForOneRoom();
1225


task12_operateForSeveralDays();

task13_displayAllCheckedRoomNumber();
R2 R4 R5 R9 R11 R19

task14_displayYesterdayIncome();
9082

task15_displayTodayInfo();
Day{name='FRI', rate=1.0, gift='GYM CARD'}

task16_displayRecentDaysIncome();
53024
```

Rules

1. You need to submit those ".java" files including
   Hotel.java
   ConcreteHotel.java
   Day.java
   Room.java
   LuxuryRoom.java
   OrdinaryRoom.java.
2. Do not modify or remove any methods or fields that have been already defined in these two files **Hotel.java** and **Room.java**.
3. No Chinese characters are allowed to appear in your code.
4. No package included.
5. The output must strictly follow the description and the sample of each task, and you can only get points for each task only when your output are completely the same as the output of test case.
6. Please submit your assignment on the SAKAI site of your lab section. Marks will be deducted if you submit later than the deadline. If you submit your assignment within 24 hours after the deadline (grace period), your score will be half of the score you could get if the submission was made before the deadline. Assignments submitted after the grace period will not be graded (meaning you will get a zero for the assignment).