Написать класс для бронирования комнат в гостинице.

1. У класса должен быть статический атрибут *list_id* – список идентификаторов постояльцев всех гостиниц.
2. У класса должны быть динамические атрибуты *name* – название гостиницы и *amount_position* – общее количество комнат, задаваемые при инициализации объекта.
3. У класса должен быть динамический метод *spot_list* – список комнат, заполняющийся при инициализации объекта в зависимости от переданного атрибута *amount_position*
4. У класса должен быть динамический атрибут *residentId_list* – список идентификаторов постояльцев конкретной гостиницы.
5. У класса должен быть динамический атрибут *book_dict* – словарь забронированных комнат, представляющий собой пары: {id постояльца : забронированный номер}
6. У класса должен быть динамический метод *bookIt* для бронирования номера. На вход метода подается *spot_number* - целое число – предполагаемый номер для брони. Логика работы метода:
    - Если комнаты с таким номером не существует (номера нет в *spot_list)*, вывод сообщения: «Такого номера нет в базе»
    - Если комната с таким номером существует в *spot_list*, проверяется не занята ли комната:
        - Если номер комнаты есть в *book_dict* – вывод сообщения: «Номер занят! Выберите другой»
        - Если номера комнаты нет в *book_dict* – вывод сообщения: «Вы успешно забронировали номер!». Генерируется случайное целое число – *residentId* идентификатор постояльца, число должно генерироваться до тех пор, пока не станет уникальным (такого числа еще нет в *list_id)*. Сгенерированный id заносится в список *list_id* и в список *residentId_list*. В динамический атрибут *book_dict* добавляется запись формата { *residentId* : *spot_number* }

Пример:

Входные данные:

```
Hostel1 = Booking(name='Sweet Home', amount_position=10)     # Создаем хостел (всего 10 комнат)
Hostel1.bookIt(1)                                            # Бронируем комнату № 1
Hostel1.bookIt(2)                                            # Бронируем комнату № 2
Hostel1.bookIt(2)                                            # Бронируем комнату № 2
Hostel1.bookIt(11)                                           # Бронируем комнату № 11
Hostel1.bookIt(5)                                            # Бронируем комнату № 5
print(Hostel1.book_dict)                                     # Смотрим занятые комнаты
```

Выходные данные:

```
Вы успешно забронировали номер!              # Бронируем комнату № 1

Вы успешно забронировали номер!              # Бронируем комнату № 2

Номер занят! Выберите другой                 # Бронируем комнату № 2

Такого номера нет в базе                     # Бронируем комнату № 11

Вы успешно забронировали номер!              # Бронируем комнату № 5

{9507: 1, 8556: 2, 1344: 5}                  # Смотрим занятые комнаты
```

Предварительно написать вспомогательный класс «еда/продукт» (Food). У этого класса достаточно определить два динамический атрибута: *name* – название продукта и *volume* – объем продукта. Оба атрибута задаются при инициализации объекта.

Написать класс «холодильник»:

1. У класса должен быть динамический атрибут *volume* – объем холодильника, задаваемый при инициализации объекта.
2. У класса должен быть динамический атрибут *fullness* – заполненность холодильника, при инициализации объекта *fullness* = 0.
3. У класса должен быть динамический атрибут *food_list* – массив в котором хранится информация о продуктах в холодильнике.
4. У класса должен быть динамический метод *putFood* (положить еду), принимающий на вход объект вспомогательного класса *Food* – продукт, который складывается в холодильник. Метод должен выводить: текущий свободный объем холодильника; название и объем продукта, который пользователь хочет положить. Если объем продукта больше, чем свободное пространство, должно выводится сообщение: «Для этого продукта в холодильнике нет места!». Если места достаточно – заполненность холодильника *fullness* должна увеличиваться на *volume* продукта, а сам продукт добавляться в *food_list*.
5. У класса должен быть динамический метод *takeFood* (взять еду), принимающий на вход переменную *food_name* строку – название продукта, которое пользователь хочет взять из холодильника. Метод должен выводить сообщение «Вы хотите взять {*food_name*}», далее метод должен проверить, что продукт с таким названием есть в холодильнике (*food_list*). Если продукт с таким названием есть в холодильнике, должно выводиться сообщение: «{*food_name*} есть в холодильнике!», заполненность холодильника *fullness* должна уменьшаться на объем забираемого продукта и продукт должен удаляться из *food_list*. Учесть, что в холодильнике может быть несколько продуктов с одинаковым *name*, метод должен удалять только один продукт! Если в холодильнике не оказалось продукта с *name = food_name* должно выводится сообщение: «{food_name} нет в холодильнике!»

Пример:

Входные данные:

```
# Создаем несколько экземпляров (объектов) класса Food
meat1 = Food(name='Мясо', volume=4)
bread = Food('Хлеб', 2)
milk = Food('Молоко', 1)
eggs = Food('Яйца', 1)
meat2 = Food('Мясо', 2)

myFridge = Fridge(8) # Создаем холодильник объемом 8
myFridge.putFood(meat2) # Складываем в холодильник
второй кусок мяса
myFridge.putFood(meat1) # Складываем в холодильник
первый кусок мяса
myFridge.putFood(milk) # Складываем в холодильник молоко
myFridge.putFood(eggs) # Складываем в холодильник яйца
myFridge.putFood(bread) # Складываем в холодильник хлеб

myFridge.takeFood('Торт') # Хотим достать торт
myFridge.takeFood('Мясо') # Хотим достать мясо
print(myFridge.food_list) # Проверяем сколько продуктов
осталось
print(myFridge.fullness) # Проверяем заполненность
```

Выходные данные:

```
Свободный объем холодильника: 8
Объем продукта Мясо: 2
Свободный объем холодильника: 6
Объем продукта Мясо: 4
Свободный объем холодильника: 2
Объем продукта Молоко: 1
Свободный объем холодильника: 1
Объем продукта Яйца: 1
Свободный объем холодильника: 0
Объем продукта Хлеб: 2
Для этого продукта в холодильнике нет места!
Вы хотите взять Торт
Торт нет в холодильнике!
Вы хотите взять Мясо
Мясо есть в холодильнике!
[<__main__.Food object at 0x0000019225759450>,
<__main__.Food object at 0x00000192257594D0>,
<__main__.Food object at 0x0000019225759510>]
6
```

Написать класс «электронный дневник»

1. У класса должен быть статический атрибут *list_students_id*, хранящий информацию об используемых идентификаторах.
2. У класса должен быть динамический атрибут *student_surname* – фамилия ученика, задаваемая при инициализации
3. У класса должен быть динамический атрибут *student_id* – уникальный идентификатор учащегося. Этот атрибут должен генерироваться случайным образом при инициализации. *student_id* должен быть целым числом от 0 до 9 и не должен совпадать с уже существующими числами из *list_students_id*. Не забудьте добавить сгенерированный *id* к списку существующих.
4. У класса должен быть динамический атрибут *student_report_card* – представляющий собой словарь, в котором ключи – названия предметов, а параметры – списки оценок. (Например: {'Математика': [4, 5, 4, 4, 3, 4, 5], 'Химия': [3, 3, 4, 4, 3, 4, 3], 'Литература': [5, 5, 4, 5, 4, 5, 4]})
5. У класса должен быть динамический метод *rate_student*, принимающий на вход название предмета и оценку. Метод должен добавлять оценку в *student_report_card* для соответствующего предмета или, если предмета еще нет, создавать соответствующую запись с оценкой в *student_report_card*.

Пример:

Входные данные:

```
# Просто формируем табель успеваемости:
_math = {"Математика":[4, 5, 4, 4, 3, 4, 5]}
_chem = {"Химия":[3, 3, 4, 4, 3, 4, 3]}
_lit = {"Литература":[5, 5, 4, 5, 4, 5, 4]}
_math.update(_chem)
_card = _math
_card.update(_lit)

Harry_diary = StudentDiary('Potter', _card)          # Первый дневник
Ron_diary = StudentDiary('Weasley', _card)           # Второй дневник
Hermione_diary = StudentDiary('Granger', _card)      # Третий дневник
print(StudentDiary.list_students_id)                 # Проверяем id учеников
print(Harry_diary.student_report_card)               # Проверяем табель первого ученика
Harry_diary.rate_student('Математика', 3)            # Ставим первому ученику 3 по математике
print(Harry_diary.student_report_card)               # Проверяем табель первого ученика
Ron_diary.rate_student('Физика', 4)                  # Ставим второму ученику 4 по физике
print(Ron_diary.student_report_card)                 # Проверяем табель второго ученика
```

Выходные данные:

```
[1, 2, 5]

{'Математика': [4, 5, 4, 4, 3, 4, 5], 'Химия': [3, 3, 4, 4, 3, 4, 3], 'Литература': [5, 5, 4, 5, 4, 5, 4]}

{'Математика': [4, 5, 4, 4, 3, 4, 5, 3], 'Химия': [3, 3, 4, 4, 3, 4, 3], 'Литература': [5, 5, 4, 5, 4, 5, 4]}

{'Математика': [4, 5, 4, 4, 3, 4, 5, 3], 'Химия': [3, 3, 4, 4, 3, 4, 3], 'Литература': [5, 5, 4, 5, 4, 5, 4], 'Физика': [4]}
```