```
import sys
import turtle
from time import sleep
import random


class NoParkingSpacesException(Exception):
    pass


class User:
    def __init__(self, start_time, end_time):
        self.start_time = start_time
        self.end_time = end_time


class Park:
    def __init__(self):
        # 停车位可用时间列表，一开始都是（0,24）
        self.useful_time_list = [(0, 24)]


class SharedParkingSpaceMS:
    """
    1. 计算并紧凑户主的使用时间，以减少碎片，得到没有外界车辆加入时的可用时间队列
    2. 有外界车辆申请加入时，判断有没有能满足该车辆使用时间的停车位
    3. 如果有，允许停车并修改可用时间队列
    """
    def __init__(self):
        # 假设有五位业主，五个车位
        self.user_list = [User(7, 12),
                          User(8, 19),
                          User(12, 16),
                          User(5, 7),
                          User(10, 15)]
        self.sc_list = []
        self.parking_list = [Park() for i in range(len(self.user_list))]

    def draw(self):
        """
        画格子，x轴方向为时间（/h），y轴方向为车位号
        :return:
        """
        turtle.screensize(1000, 800, "#e9e7e5")
        turtle.pensize(3)
        turtle.pencolor('#8f8f8f')
        turtle.speed(10)
        turtle.hideturtle()
        turtle.delay(delay=0)
        s = [int(300 - (600 / (len(self.user_list))) * y) for y in
range(len(self.user_list) + 1)]
        print(s)
```

```
        for y in s:
            turtle.penup()
            turtle.goto(-400, y)
            turtle.right(90)
            turtle.pendown()
            turtle.goto(400, y)
        t = [int(-400 + (800 / 24) * y) for y in range(25)]
        for x in t:
            turtle.penup()
            turtle.goto(x, 300)
            turtle.right(90)
            turtle.pendown()
            turtle.goto(x, -300)

    @staticmethod
    def draw_useful(park_index, start_time, end_time, flag):
        """
        将停车场的占用情况画出来
        :param park_index: 车位号
        :param start_time: 停车开始时间
        :param end_time: 离开时间
        :param flag: 1 为户主占用的时间，用红色标记，2为外界车辆占用的时间
        """
        color = ['#8f8f8f', '#edd6b0', '#55ecd0', '#ed8211', '#5f9928', '#2559d3',
'#94d8cb', '#5b5237']
        user_color = ['#ed3c3c', '#77061c', '#f07d85', '#e33a1c', '#b61f04']
        if flag == 2:
            col = random.choice(color)
        if flag == 1:
            col = random.choice(user_color)
        turtle.pencolor(col)
        turtle.fillcolor(col)
        turtle.pensize(1)
        turtle.begin_fill()
        turtle.penup()
        turtle.goto(-400 + (800 / 24) * start_time, 280 - (120 * park_index))
        turtle.pendown()
        turtle.goto(-400 + (800 / 24) * end_time, 280 - (120 * park_index))
        turtle.goto(-400 + (800 / 24) * end_time, 280 - (120 * park_index) - 80)
        turtle.goto(-400 + (800 / 24) * start_time, 280 - (120 * park_index) - 80)
        turtle.goto(-400 + (800 / 24) * start_time, 280 - (120 * park_index))
        turtle.end_fill()

    @staticmethod
    def draw_no_useful():
        """
        没有停车位时，显式文字
        :return:
        """
        turtle.penup()
        turtle.goto(-100, 0)
        turtle.pencolor('red')
        turtle.write("没用可用车位", font=('Arial', 40, 'normal'))
        # sleep(0.5)
```

```
        turtle.undo()

    def get_useful_list(self, user: User):
        """
        判断某个新来的用户能不能停车，如果没有车位，抛出 NoParkingSpacesException 异常
        :param user: User(start_time, end_time)
        :return: 如果有车位，返回车位号
        """
        for park_index, park in enumerate(self.parking_list):
            for index, useful in enumerate(park.useful_time_list):
                if (user.start_time >= useful[0]) and (user.end_time <=
useful[1]):
                    park.useful_time_list.pop(index)
                    if useful[0] < user.start_time:
                        park.useful_time_list.append((useful[0], user.start_time))
                    if user.end_time < useful[1]:
                        park.useful_time_list.append((user.end_time, useful[1]))
                    return park_index
        raise NoParkingSpacesException

    def request(self, new_user: User):
        """
        请求一个车位
        :param new_user: User(start_time, end_time)
        :return:
        """
        print(f'{new_user.start_time}:00 , 一个外来车辆请求停车位，需要使用：
{new_user.end_time - new_user.start_time} 小时')
        try:
            park_index = self.get_useful_list(new_user)
            print(f'能满足请求，允许停车,停放在 {park_index} 号车位')
            self.draw_useful(park_index, new_user.start_time, new_user.end_time,
2)
        except NoParkingSpacesException:
            sys.stderr.write("不能满足请求！！！\n")
            self.draw_no_useful()
        self.sc_list.append(new_user)

    def run(self):
        self.draw()
        # 先将户主的占用时间加入可用队列，保证户主有可用的车位
        for i in self.user_list + self.sc_list:
            park_index = self.get_useful_list(i)
            self.draw_useful(park_index, i.start_time, i.end_time, 1)
        sleep(1)
        # 随机产生社会车辆
        for i in range(30):
            start = random.randint(0, 24)
            end = random.randint(0, 24)
            if start < end:
                user = User(start, end)
                self.request(user)
                sleep(2)
```

```
if __name__ == '__main__':
    SharedParkingSpaceMS().run()
    turtle.done()
```