МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ М. В. ЛОМОНОСОВА

Факультет Вычислительной Математики и Кибернетики

Отчет о выполнении задания практикума

«Система поддержки бронирования и заселения гостиницы»

выполнил:

Мажаров Иван

425 группа

Москва 2018

Оглавление

1. Уточненная постановка задачи

Небольшая гостиница содержит K номеров (20≤ K ≤ 30), различающихся по степени комфорта и стоимости: «люкс», «полулюкс», одноместные, простые двухместные, двухместные с раскладным диваном (например, 70 у.е. за день проживания в одноместном номере, 120 у.е. – за номер «люкс»).

Требуется создать компьютерную систему, автоматизирующую управление занятостью номеров гостиницы. Система обрабатывает входной поток заявок двух видов:

- ✓ заявки, бронирующие определенные типы номеров на определенный срок;
- ✓ заявки на заселение в текущий момент.

Система хранит информацию о фактической занятости всех номеров и о их занятости в ближайшие дни (учитываются уже оплаченные вперед дни), а также сведения о произведенной брони номеров, и использует все эти данные при обработке заявок. При бронировании номеров система автоматически формирует сообщение-подтверждение брони, а при выезде постояльцев она оформляет им счета.

Стратегия обработки заявок строится так, чтобы добиться максимальной занятости гостиницы с целью увеличения ее прибыли. Для этого система гибко распоряжается номерным фондом: в частности, при нехватке нужных номеров можно использовать пустующие номера большей комфортности (по меньшей цене), например, при нехватке одноместных номеров можно поселить одного человека в двухместный номер (за 70% его стоимости).

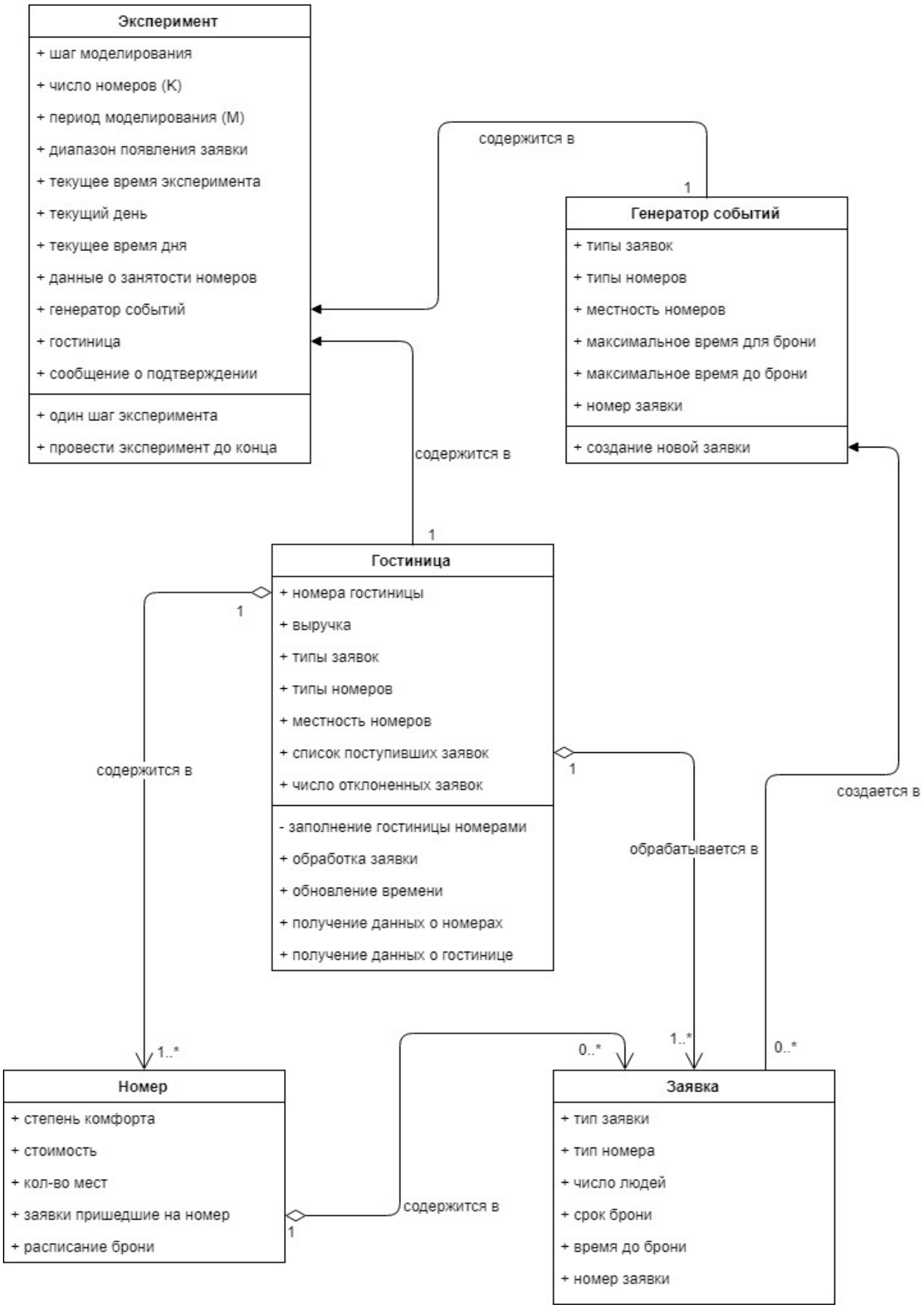
Для тестирования построенной системы необходимо смоделировать входной поток заявок на бронирование и поселение. Вид и параметры каждой заявки определяются случайным образом. Интервал между появлением двух заявок следует моделировать как случайную величину из определенного диапазона (например, от 1 до 5 часов).

Период моделирования – M дней (12≤ M ≤ 30), шаг – несколько часов. Цель моделирования – изучение стратегий обработки заявок на заселение. В параметры моделирования следует включить: числа K и M, количество номеров каждой категории, характеристики используемых случайных величин.

В ходе моделирования система должна предоставлять всю необходимую информацию о занятости номеров гостиницы. По окончании моделирования

выводится статистика заселения номеров, выполненных заявок, процент загруженности отдельных категорий номеров и гостиницы в целом.

2. Диаграмма классов



3. Текстовые спецификации классов

```
class Room:
    def __init__(self, class_, price, cap):
        # степень комфорта (тип) номера
        self.class_ = class_
        # стоимость номера
        self.price = price
        # местность номера
        self.cap = cap
        # заявки пришедшие на этот номер
        self.app_array = []
```

```
class Application:
    def __init__(self, uid, app_type, room_type, people, time_to_book,
time_before_booking=0):
        # тип заявки
        self.type = app_type
        # тип номера в заявке
        self.room_type = room_type
        # число человек на заселение
        self.people = people
        # время бронирования/заселения
        self.time_to_book = time_to_book
        # время до заселения (если заявка на бронь)
        self.time_before_booking = time_before_booking
        # уникальный номер заявки
        self.uid = uid
```

```
class EventGenerator:
    # конструктор
    def __init__(self, app_types, room_types, room_cap):
        # типы заявок
        self.app_types = app_types
        # типы номеров
        self.room_types = room_types
        # местность номеров
        self.room_cap = room_cap
        # максимальное время для брони
        self.max_time_to_book = 7
        # максимально время до брони
        self.max_time_before_booking = 7
        # номер заявки
        self.app_id = -1

    # создание новой заявки
    def get_new_application(self):
```

```python
class Hotel:
    # конструктор
    def __init__(self, n_rooms, app_types, room_types, room_cap):
        # номера гостницы
        self.rooms = self.__generate_rooms__(n_rooms)
        # выручка
        self.profit = 0
        # типы заявок
        self.app_types = app_types
        # типы номеров
        self.room_types = room_types
        # местность номеров
        self.room_cap = room_cap
        # список поступивших заявок
        self.all_apps = defaultdict()
        # число отклоненных заявок
        self.rejected = 0

    # заполнение гостиницы номерами
    def __generate_rooms__(self, n_rooms):

    # обработка заявки
    def process_application(self, application):

    # обновление времени
    def process_one_tick(self):

    # получение данных о занятости номеров
    def get_data(self, day, max_days, data):

    # получение данных о состоянии гостиницы
    def report(self):
```

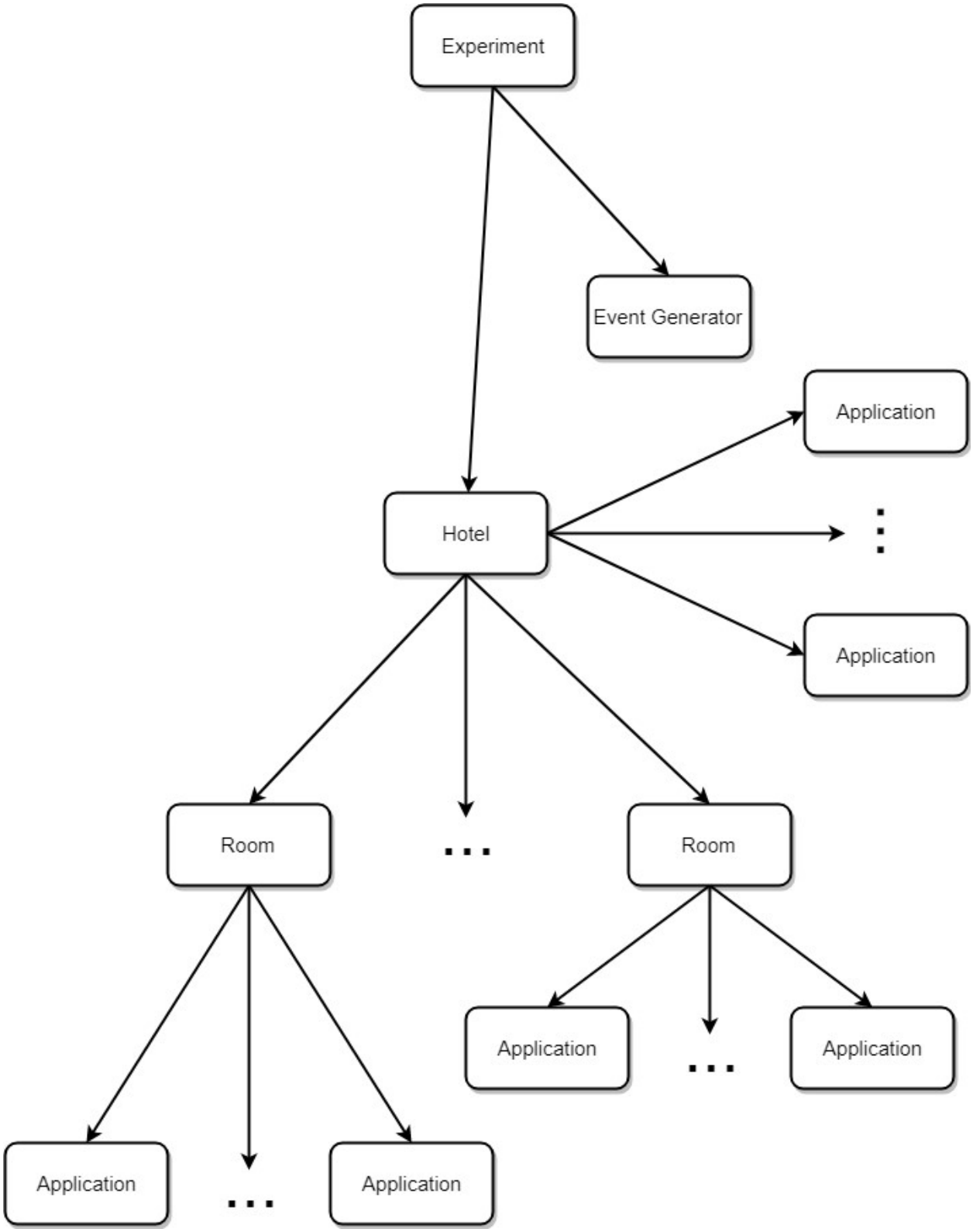
```python
class Experiment:
    # конструктор
    def __init__(self, step=5, n_rooms=24, simulation_period=30,
max_time_for_new_application=5):
        # шаг
        self.step = step
        # число номеров в гостнице
        self.n_rooms = n_rooms
        # период моделирования
        self.period = simulation_period
        # верхняя граница времени до новой заявки
        self.max_time = max_time_for_new_application
        # текущее время эксперимента
        self.curr_time = 0
        # текущий день эксперимента
        self.curr_day = 0
        # текущее время дня
```

```
        self.time = 0
        # время до поступления заявки
        self.time_for_new_application = -1
        # данные о занятости номеров
        self.data = []
        # генератор событий
        self.event_generator = EventGenerator(app_types, room_types, room_cap)
        # гостиница
        self.hotel = Hotel(self.n_rooms, app_types, room_types, room_cap)
        # сообщение о подтверждении брони
        self.message = ""

    # один шаг эксперимента
    def make_step(self):

    # провести эксперимент до конца
    def complete(self):
```

4. Диаграмма объектов



5. Инструментальные средства

Язык разработки – Python 3.6

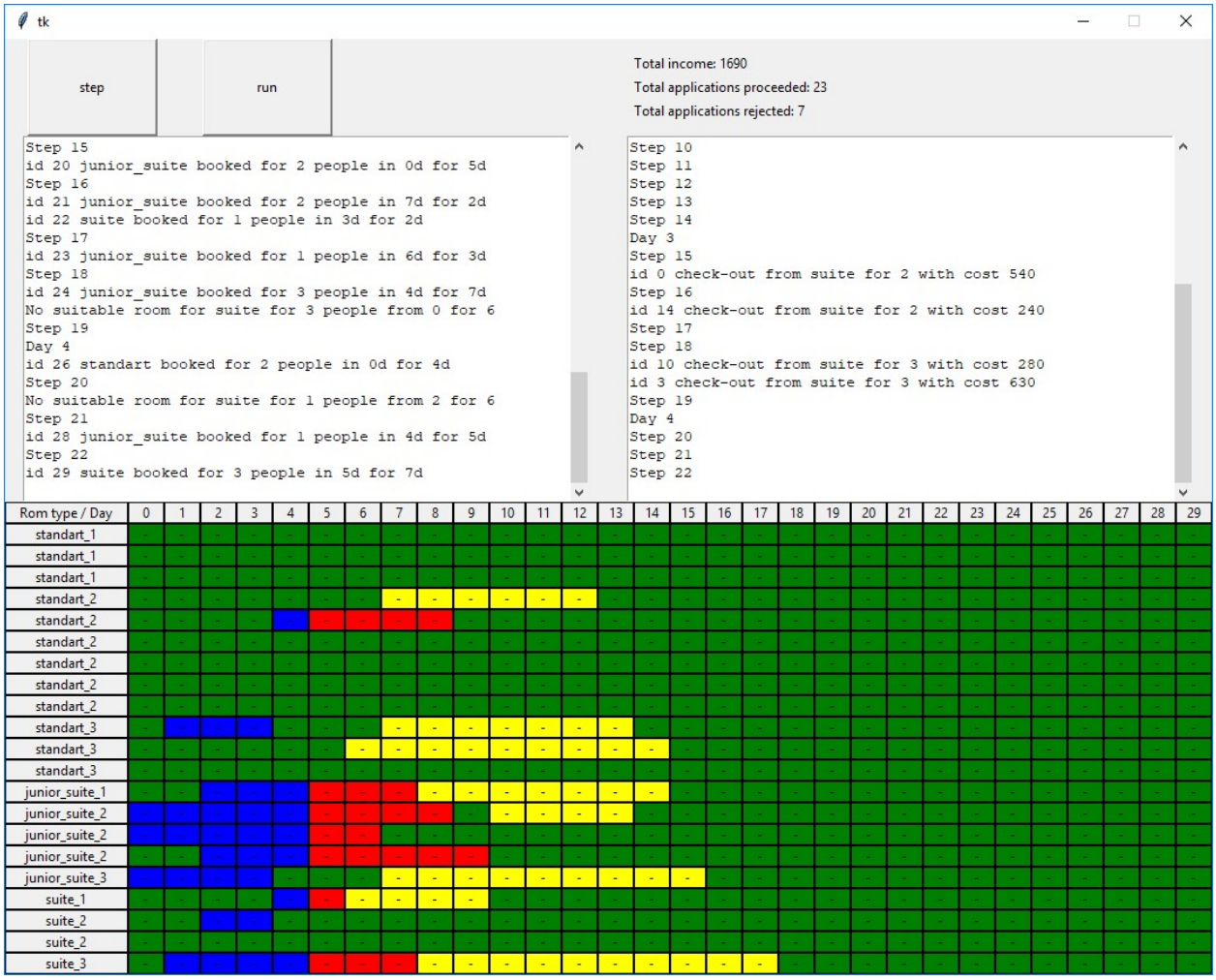
Среда разработки – Jupyter Notebook, Visual Studio Code

Используемые библиотеки – tkinter

6. Файловая структура

Application.py	- описание и реализация класса «Заявка»
EventGenerator.py	- описание и реализация класса «Генератор событий»
Experiment.py	- описание и реализация класса «Эксперимент»
Hotel.py	- описание и реализация класса «Гостиница»
Main.py	- основная программа
Room.py	- описание и реализация класса «Номер»

7. Пользовательский интерфейс



- дни, в которые номер был занят

- дни, в которые в номер заселены люди

- дни, в которые номер забронирован

- свободные дни