

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ М. В. ЛОМОНОСОВА  
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ

ОТЧЕТ ПО ЗАДАНИЮ  
**«Модель работы  
магазина или супермаркета»**

Выполнила:  
студентка 425 группы  
Телегина А. Д.

Преподаватель:  
Большакова Е. И.

Москва  
2020

# Содержание

Постановка задачи	2
Диаграмма классов программной системы	3
Текстовая спецификация классов	4
Диаграмма объектов программной системы	13
Инструментальные средства	13
Файловая структура системы	13
Пользовательский интерфейс	14

## Постановка задачи

В данной задаче рассматривается механизм работы супермаркета или обычного магазина.

В заведении работает несколько касс ( $1 \leq K \leq 7$ ), которые обслуживают приходящих покупателей. Покупатели приходят с интервалом, который является случайной величиной. Интервал зависит от дня недели, времени суток и количества скидок в магазине (в конце недели и в конце дня плотность потока клиентов выше). Длительность обслуживания и сумма покупки - также случайные величины в некотором диапазоне, причем эти величины не зависят от внешних факторов.

Одним из параметров модели является максимальная длина очереди у кассы - вновь прибывшие покупатели не смогут встать в очередь, если в ней уже скопилось максимально возможное количество человек. Если ни одной кассы, куда можно было бы встать, нет, то покупатели уходят.

Необходимо разработать эту модель обслуживания приходящих покупателей в магазине. Создаваемое приложение реализует эту модель. Пользователь приложения управляет работой магазина. Он моделирует работу в течение недели. Шаг моделирования пользователь тоже задает заранее (это число в интервале от 10 до 60 минут).

Перед началом эксперимента пользователю нужно задать параметры системы, которые включают в себя:

1. рабочие часы в будни;
2. рабочие часы в выходные;
3. количество работающих касс в будни;
4. количество работающих касс в выходные;
5. максимальная длина очереди в кассах;
6. размер скидок;

В течение игры пользователь получает ежедневную информацию о количестве обслуженных и потерянных покупателей, общей прибыли магазина, средней длины очереди и среднего времени ожидания клиента. Общая прибыль магазина за каждый день вычисляется с учётом выплаты зарплат кассирам магазина. Помимо ежедневной информации, собирается соответствующая статистика и за всю неделю.

Цель исследования работы магазина – определить оптимальные режимы его работы – режимы, при которых работающие кассы или продавцы всегда заняты, и увеличивается прибыль от продаж.

#

## Текстовая спецификация классов

```
1 class Supermarket(object):
2     """
3     Сингтон класс.
4     Главный класс магазина
5     """
6     __slots__ = '_max_queue', '_discount' \
7                 'currrent_stat' 'desks'
8
9     def __init__(self, max_queue, discount):
10         """
11         Создание основных параметров магазина,
12         настройка размера скидки
13         """
14
15     def get_discount(self):
16         """
17         Возвращает размер скидки
18         """
19
20     def open_day(self, desks):
21         """
22         Функция начала рабочего дня;
23         Открытие касс
24         """
25
26     def close_day(self):
27         """
28         Функция очищает очереди покупателей и закрывает кассы
29         """
30
31     def add_customer(self):
32         """
33         Функция добавления нового покупателя в кассу
34         с учётом максимальной длины очереди.
35         """
36
37     def get_info(self):
38         """
39         Функция подсчитывает размер очередей в кассах.
40         """
41
42     def update_cash_stat(self):
43         """
```

```

44     Функция обновляет статистику по очередям в кассах.
45     """
46
47     def update_cash_desks(self):
48         """
49         По прошествии 1 минуты функция уменьшает время
50         обслуживания текущего покупателя в кассе на 1 мину
51         ту,
52         используя локальный таймер каждой кассы.
53         """
54
55     def _adjust_discount(self, discount):
56         """
57         Функция устанавливает зависимость между размером
58         скидки магазина и увеличением потока покупателей.
59         """
60
61     def _find_cash_desk(self):
62         """
63         Функция находит кассу с минимальной очередью и возв
64         ращает
65         её номер и кол-во покупателей в ней.
66         """
67
68 class CashDesk(object):
69     """
70     Класс кассы магазина
71     """
72     __slots__ = '_clients', '_margin', '_start_time'
73     def __init__(self):
74         """
75         Создание объекта, инициализация основных параметров
76         значениями.
77         """
78
79     def get_clients(self):
80         """
81         Возвращает очередь покупателей.
82         """
83
84     def queue_length(self):
85         """
86         Возвращает длину текущей очереди покупателей.
87         """
88
89     def get_margin(self):

```

```

88         """
89         Возвращает прибыль кассы.
90         """
91
92     def push_customer(self, new_client):
93         """
94         Добавление нового покупателя в очередь.
95         """
96
97     def clear(self):
98         """
99         Очищение очереди кассы.
100        """
101
102     def decrease_timer(self):
103         """
104         Увеличивает локальный таймер кассы.
105         Проверятся, сколько времени осталось текущему покуп
106             ателю
107         до конца обслуживания.
108         """
109
110     def _pop_customer(self):
111         """
112         Убирает обслуженного покупателя из очереди.
113         """
114
115 class Statistics(object):
116     """
117     Класс собираемой статистики магазина.
118     """
119     __slots__ = '_desks', '_profit', '_lost_clients' \
120         '_done_clients', '_waiting', 'que_len'
121
122     def __init__(self, desks):
123         """
124         Создание класса статистики.
125         """
126
127     def get_profit(self):
128         """
129         Возвращает полученную прибыль.
130         """
131
132     def add_lost(self):
133         """

```

```

133         Добавление информации о потерянном покупателе в статистику.
134         """
135
136     def update_custom_stat(self, client):
137         """
138         Добавление информации о обслуженном покупателе в статистику.
139         """
140
141     def current_clients(self):
142         """
143         Возвращает текущую информацию об обслуженных и потерянных покупателях в статистику.
144         """
145
146
147     def update_waiting_time(self, cash_desk):
148         """
149         Собирает статистику о среднем времени ожидания покупателей.
150         """
151
152     def update_queue(self, lengths):
153         """
154         Собирает статистику о средней длине очереди в кассе.
155         """
156
157     def prepare_stat(self, work_time):
158         """
159         Готовит полную статистику о доходе за день с учётом зарплат, обслуженных и потерянных покупателей, средней длины очереди в кассах и среднем времени ожидания.
160         """
161
162
163 class Customer(object):
164     """
165     Класс покупателя.
166     Покупатель характеризуется суммой покупки и временем обслуживания.
167     """
168     __slots__ = '_sum', '_time'
169
170
171     def __init__(self, sum_value, time):
172         """

```



```

173         Создание объекта покупателя.
174         """
175
176     def get_sum(self):
177         """
178         Возвращает сумму покупки.
179         """
180
181     def get_time(self):
182         """
183         Возвращает время обслуживания.
184         """
185
186 class Visualizer(QWidget):
187     """
188     Класс визуализации, содержащий все основные
189     функции по отрисовке пользовательского интерфейса.
190     """
191     __slots__ = 'time', 'buttons', 'titles', '
192                paint_interval', \
193                'paused', 'size', 'day', 'lengths', 'total'
194     def __init__(self):
195         """
196         Создание объекта.
197         """
198
199     def start(self, market, desks, hours):
200         """
201         Начало эксперимента, обнуление таймера.
202         """
203
204     def _default_params(self):
205         """
206         Установка параметров по умолчанию,
207         обнуление статистик.
208         """
209
210     def _start_window(self):
211         """
212         Создание и визуализация главного окна приложения.
213         """
214
215     def _init_stat_text(self):
216         """
217         Установка и настройка текста для вывода статистики.

```

```

218 def _init_day_scores(self):
219     """
220     Инициализация и настройка полей для вывода
221     ежедневной статистики.
222     """
223
224 def _init_total_scores(self):
225     """
226     Инициализация и настройка полей для вывода
227     недельной статистики.
228     """
229
230 def _init_label(self, text, setting, coord):
231     """
232     Создает надпись с заданным текстом и координатами.
233     """
234
235 def _init_discount_slider(self):
236     """
237     Создает ползунок для установки количества скидки.
238     """
239
240 def _init_button(self, text, size, coord, function=None
241 ):
242     """
243     Создает кнопку с заданным текстом и размером;
244     Расположена в точках с заданными координатами.
245     """
246
247 def _init_title_edit(self, coord, text, mode=None):
248     """
249     Создает поля для ввода параметров.
250     Поле расположено в заданных координатах
251     и заданным текстом по умолчанию.
252     """
253
254 def _init_market_settings(self):
255     """
256     Создает заголовки и подписи к полям ввода параметро
257     в.
258     """
259
260 def _set_functions(self):
261     """
262     Определяет функции обработки событий
263     для соответствия формату PyQt.

```

```

262         """
263
264     def _reset_timer(self):
265         """
266         Сброс таймера при закрытии рабочего дня
267         или при завершении эксперимента.
268         """
269
270     def _update_score(self):
271         """
272         Обновление вывода ежедневной статистики на экран.
273         """
274
275     def _update_total_score(self, start=False):
276         """
277         Обновление вывода недельной статистики на экран.
278         """
279
280     def _reset_stat_scores(self):
281         """
282         Обнуление статистики при выводе на экран.
283         """
284
285     def _close_event(self, event):
286         """
287         Выход из приложения.
288         """
289
290     def _kill(self):
291         """
292         Начало нового эксперимента.
293         """
294
295     def _paint_event(self, e):
296         """
297         Функция, отвечающая за отрисовку процесса.
298         """
299
300     def _pause(self):
301         """
302         Постановка эксперимента на паузу.
303         """
304
305     def _paint_paddle(self, painter):
306         """
307         Отрисовка касс и покупателей.

```

```

308         """
309
310     def _timer_event(self):
311         """
312         Таймер для отрисовки.
313         """
314
315 class World:
316     """
317     Класс внешнего мира.
318     """
319     __slots__ = 'market', 'cash_desk', 'work_hours' \
320               'paramters', 'visualizer'
321     def __init__(self):
322         """
323         Создание объекта и инициализация параметров по умол
324             чанию
325         """
326
327     def start_game(self):
328         """
329         Начало визуализации, установка параметров по умолча
330             нию.
331         """
332
333     def _adapt_functions(self):
334         """
335         Установка обработчиков на поля для ввода параметров
336             .
337         """
338
339     def _start_experiment(self):
340         """
341         Начало эксперимента, проверка необходимых параметро
342             в системы.
343         """
344
345     def _start_working(self):
346         """
347         Создание объекта магазина и непосредственно начало
348             работы.
349         """
350
351     def _cash_desk_weekday(self, text):
352         """
353         Обработчик на ввод параметров эксперимента.

```

```

349         """
350
351     def _cash_desk_weekend(self, text):
352         """
353         Обработчик на ввод параметров эксперимента.
354         """
355
356     def _work_hours_weekday(self, number):
357         """
358         Обработчик на ввод параметров эксперимента.
359         """
360
361     def _work_hours_weekend(self, number):
362         """
363         Обработчик на ввод параметров эксперимента.
364         """
365
366     def _set_interval(self, number):
367         """
368         Обработчик на ввод параметров эксперимента.
369         """
370
371     def _max_queue_len(self, number):
372         """
373         Обработчик на ввод параметров эксперимента.
374         """
375
376     def _adjust_discount(self, value):
377         """
378         Обработчик на ввод параметров эксперимента.
379         """
380
381     def _check_config(self, config):
382         """
383         Проверка введённых данных на корректность.
384         """

```

## Диаграмма объектов программной системы

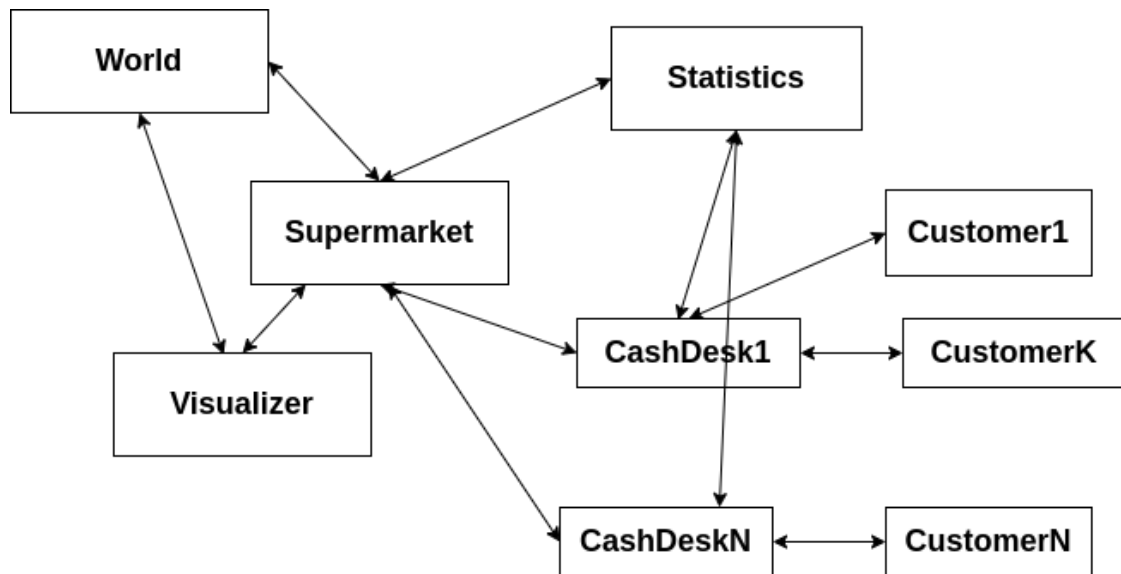


Рис. 2: Диаграмма объектов программной системы

## Инструментальные средства

1. Язык программирования: **Python 3.6**
2. Среда разработки: **Sublime Text**
3. Используемые библиотеки: **PyQt5, random, math**

## Файловая структура системы

1. **main.py** - файл, отвечающий за запуск программы
2. **World.py** - файл, содержащий описание класса внешнего мира
3. **Visualizer.py** - файл с описанием класса визуализации, отвечает за отрисовку пользовательского интерфейса
4. **Supermarket.py** - файл с описанием класса магазина
5. **CashDesk.py** - файл с описанием класса кассы
6. **Statistics.py** - файл с описанием класса статистики
7. **Customer.py** - файл с описанием класса покупателя
8. **Utils.py** - вспомогательные утилиты
9. **Constant.py** - файл со вспомогательными константами

## Пользовательский интерфейс

При запуске программы открывается основное окно игры, в котором пользователю нужно сначала ввести параметры моделирования (интервал моделирования, количество работающих касс, количество скидок и максимальную длину очереди). Так же можно отредактировать количество рабочих часов магазина (по умолчанию для будних дней - 8, для выходных - 11).

После ввода необходимых параметров для запуска эксперимента нужно нажать кнопку **Start experiment**.

После нажатия начнется симуляция работы магазина: откроется заданное количество касс и начнётся прием покупателей. В центре окна приложения будет визуализироваться информация о текущем состоянии магазина в заданном пользователем интервале.

В окне приложения также выводится некоторая информация о текущем состоянии магазина (ежедневная статистика на данный момент времени).

Во время симуляции нет возможности менять параметры моделирования, есть возможность только приостановить эксперимент, нажав на кнопку **Pause/Continue**.

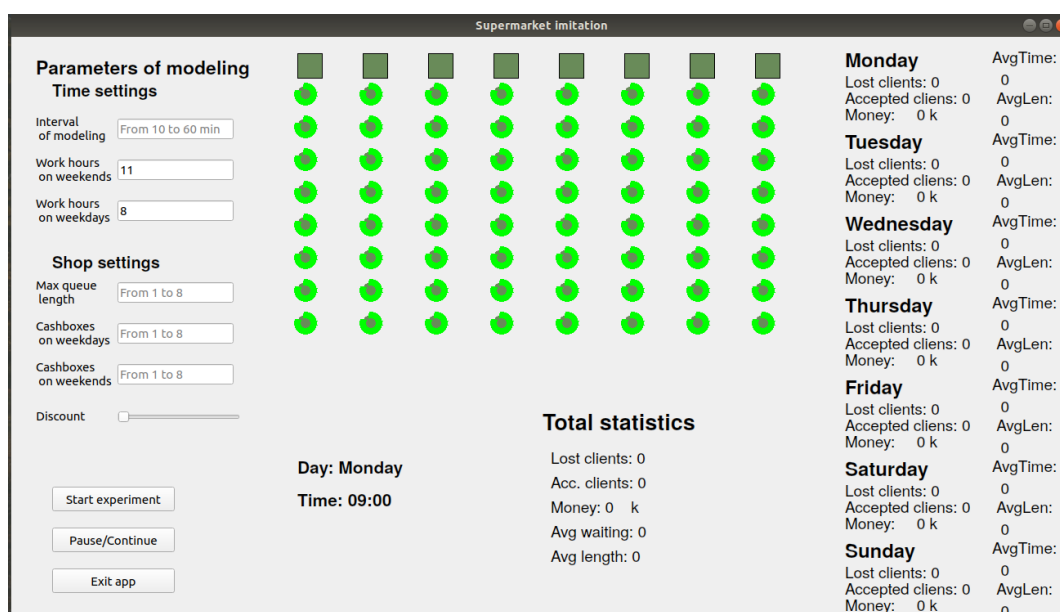


Рис. 3: Основное окно приложения

Базовые настройки приложения можно менять либо после окончания текущего эксперимента, либо после нажатия клавиши **Pause/Continue**. И в том, и в другом случае, при смене каких-либо параметров, начнётся новый эксперимент, и вся предыдущая статистика обнулится.

Для выхода из приложения нужно нажать кнопку **Exit app**.