Московский государственный университет
имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Отчет по заданию

«Модель работы магазина или супермаркета»

Выполнила:
студентка 424 группы
Телегина А. Д.

Преподаватель:
Большакова Е. И.

Москва
2020

Содержание

Постановка задачи

В данной задаче рассматривается механизм работы супермаркета или обычного магазина.

В заведении работает несколько касс ($1 <= K <= 7$), 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.

Диаграмма классов программной системы

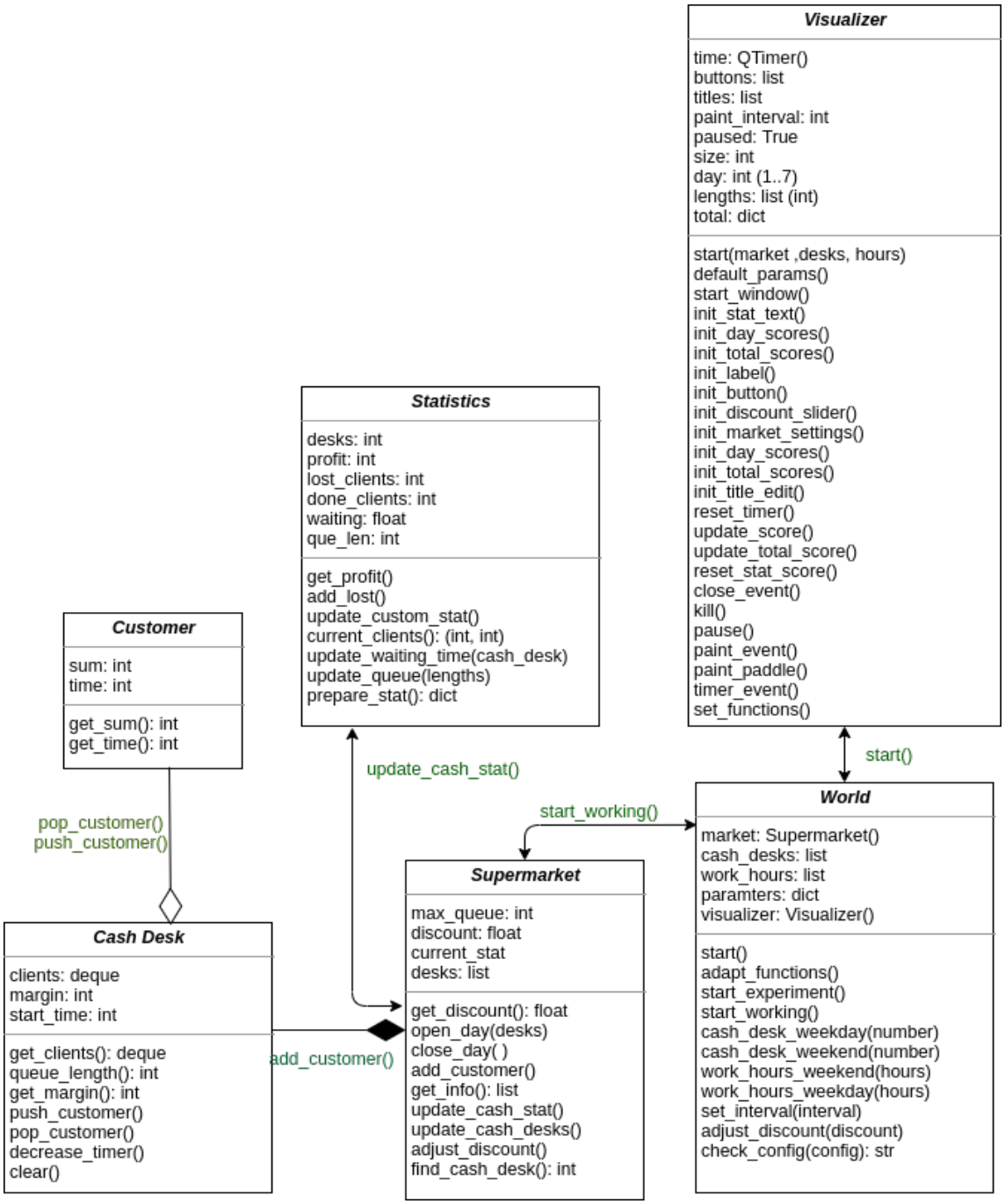

Рис. 1: Диаграмма классов программной системы

Текстовая спецификация классов

```
1  class Supermarket(object):
2      """
3      Синглтон 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 кас
               сы
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          обслуживанися текущего покупателя в кассе на 1 мину
                ту,
51          используя локальный таймер каждой кассы.
52          """
53
54      def _adjust_discount(self, discount):
55          """
56          Функция устанавливает зависимость между размером
57          скидки магазина и увеличением потока покупателей.
58          """
59
60      def _find_cash_desk(self):
61          """
62          Функция находит кассу с минимальной очередью и возв
                ращает
63              её номер и кол-во покупателей в ней.
64          """
65
66  class CashDesk(object):
67      """
68      Класс кассы магазина
69      """
70      __slots__ = '_clients', '_margin', '_start_time'
71      def __init__(self):
72          """
73          Создание объекта, инициализация основных параметров
74              значениями.
75          """
76
77      def get_clients(self):
78          """
79          Возвращает очередь покупателей.
80          """
81
82      def queue_length(self):
83          """
84          Возвращает длину текущей очереди покупателей.
85          """
86
87      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
                ателю
106         до конца обслуживания.
107         """
108
109     def _pop_customer(self):
110         """
111         Убирает обслуженного покупателя из очереди.
112         """
113
114 class Statistics(object):
115     """
116     Класс собираемой статистики магазина.
117     """
118     __slots__ = '_desks', '_profit', '_lost_clients' \
119             '_done_clients', '_waiting', 'que_len'
120
121     def __init__(self, desks):
122         """
123         Создание класса статистики.
124         """
125
126     def get_profit(self):
127         """
128         Возвращает полученную прибыль.
129         """
130
131     def add_lost(self):
132         """
```

```
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
144             и потерянных покупателях в статистику.
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 кассах.
155             """
156
157         def prepare_stat(self, work_time):
158             """
159             Готовит полную статистику о доходе за день с учётом зарплат, обслуженных и потерянных покупателях,
160                 средней длине очереди в кассах и среднем времени ожидания.
161             """
162
163 class Customer(object):
164     """
165     Класс покупателя.
166     Покупатель характеризуется
167     суммой покупки и временем обслуживания.
168     """
169     __slots__ = '_sum', '_time'
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', '
            paint_interval', \
192                 'paused', 'size', 'day', 'lengths', 'total'
193         def __init__(self):
194             """
195             Создание объекта.
196             """
197         def start(self, market, desks, hours):
198             """
199             Начало эксперимента, обнуление таймера.
200             """
201
202         def _default_params(self):
203             """
204             Установка параметров по умолчанию,
205             обнуление статистик.
206             """
207
208         def _start_window(self):
209             """
210             Создание и визуализация главного окна приложения.
211             """
212
213         def _init_stat_text(self):
214             """
215             Установка и настройка текста для вывода статистики.
216             """
217
```

```
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         Создает кнопку с заданным текстом и размером;
243         Расположена в точках с заданными координатами.
244         """
245 
246     def _init_title_edit(self, coord, text, mode=None):
247         """
248         Создает поля для ввода параметров.
249         Поле расположено в заданных координатах
250         и заданным текстом по умолчанию.
251         """
252 
253     def _init_market_settings(self):
254         """
255         Создает заголовки и подписи к полям ввода параметро
            в.
256         """
257 
258     def _set_functions(self):
259         """
260         Определяет функции обработки событий
261             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
                чанию
324             """
325
326         def start_game(self):
327             """
328             Начало визуализации, установка параметров по умолча
                нию.
329             """
330
331         def _adapt_functions(self):
332             """
333             Установка обработчиков на поля для ввода параметров
                .
334             """
335
336         def _start_experiment(self):
337             """
338             Начало эксперимента, проверка необходимых параметро
                в системы.
339             """
340
341         def _start_working(self):
342             """
343             Создание объекта магазина и непосредственно начало
                работы.
344             """
345
346         def _cash_desk_weekday(self, text):
347             """
348             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

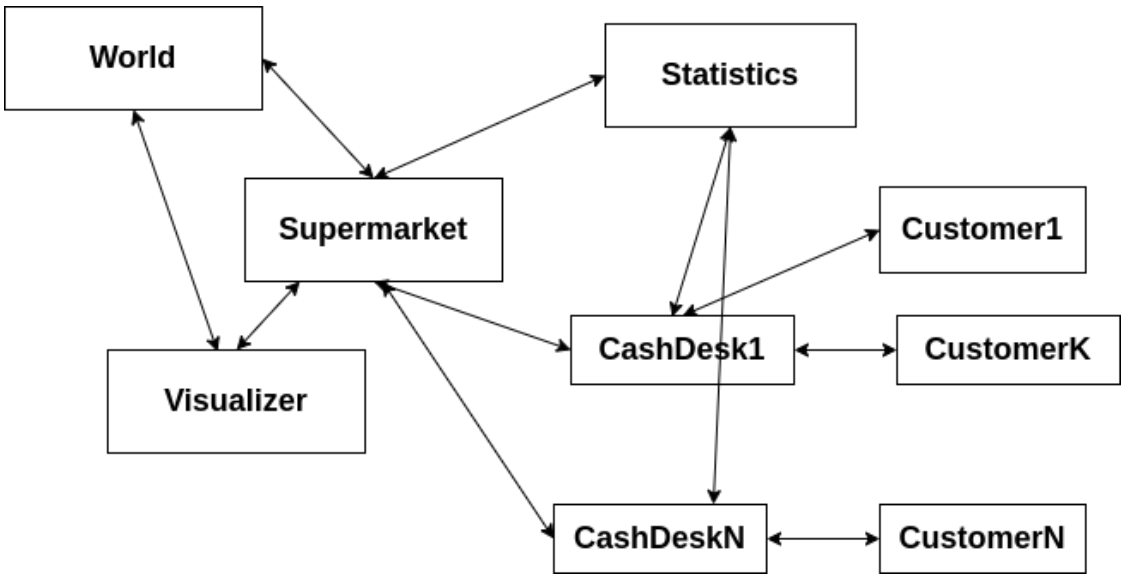

Рис. 2: Диаграмма объектов программной системы

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**.

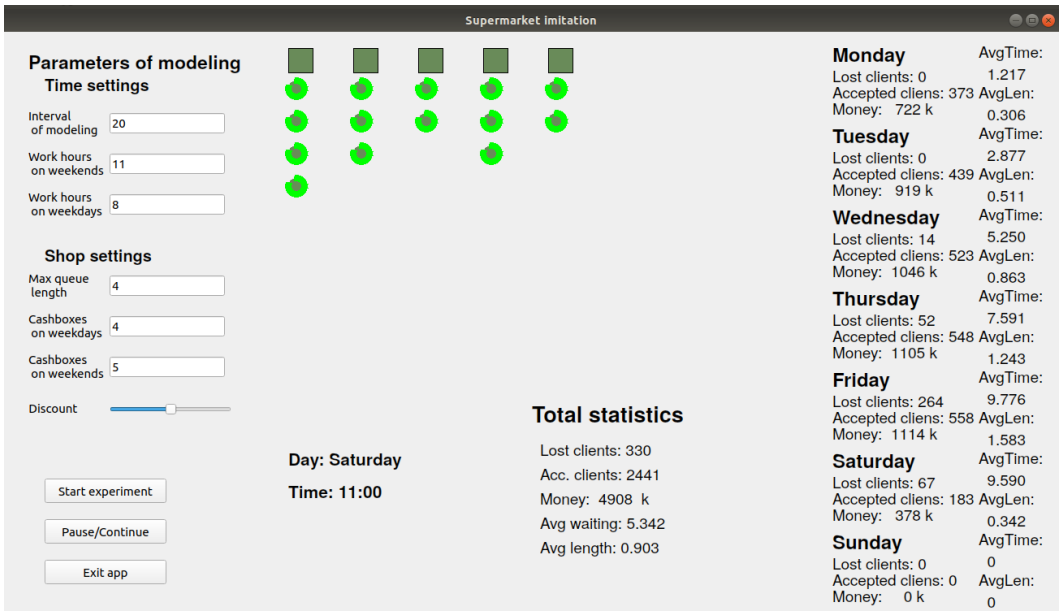

Рис. 3: Основное окно приложения

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**.