Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления (ИУ)»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии (ИУ7)»

ОТЧЕТ

по Лабораторной работе №6

по курсу «Моделирование»

на тему: «Моделирование работы электронной очереди»

Студент ИУ7-73Б _____ К.Э. Ковалец
(Группа) (Подпись, дата) (И. О. Фамилия)

Преподаватель _____ И.В. Рудаков
(Подпись, дата) (И. О. Фамилия)

2022 г.

Содержание

1 Моделируемая модель

1.1 Задание

В данной лабораторной работе моделируется следующая система. В пункт получения документов приходят клиенты с заданным интервалом времени, которые сначала подходят к терминалу выдачи талонов. У каждого терминала формируется своя очередь. Клиент выбирает очередь с минимальной длиной. Терминалы обслуживают клиентов за заданный интервал времени. Далее клиент отправляется к окну, в котором его обслужат. У каждого окна формируется своя очередь. Клиента отправляют к окну с минимальной очередью. В окне клиента обслуживают за заданный интервал времени. Количество клиентов задается.

1.2 Схема модели

На рисунке 1.1 представлена структурная схема модели.

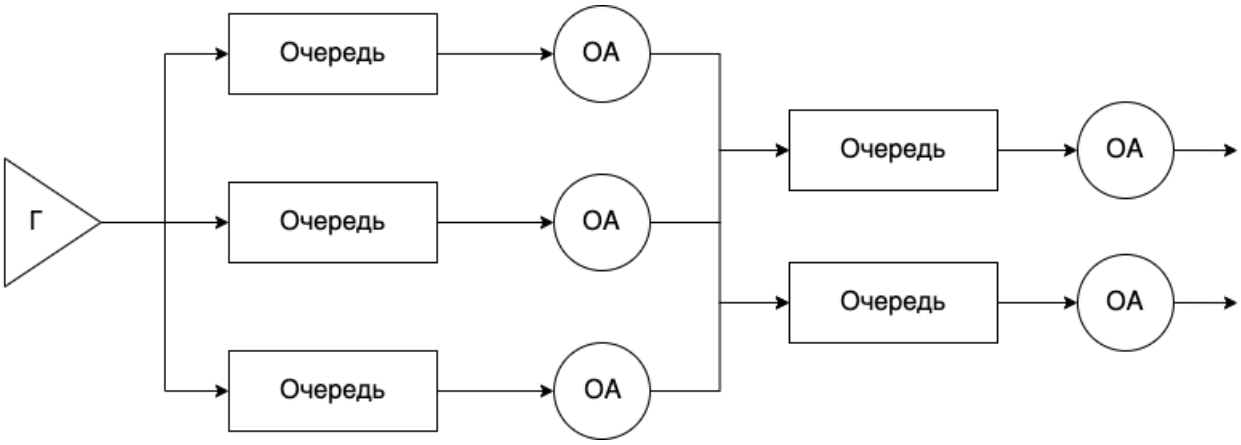

Рисунок 1.1 – Структурная схема модели

2 Результаты работы

2.1 Листинги программы

В листинге 2.1 представлена реализация генератора.

Листинг 2.1 — Реализация генератора

```
class Generator:
    def __init__(self, distribution, countClients):
        self.distribution = distribution
        self.receivers = []
        self.numbRequests = countClients
        self.next = 0

    def nextTime(self):
        return self.distribution.generate()

    def generateRequest(self):
        self.numbRequests -= 1
        receiverMin = self.receivers[0]

        for receiver in self.receivers:
            if receiver.currentQueueSize < receiverMin.currentQueueSize:
                receiverMin = receiver

        receiverMin.receiveRequest()
        return receiverMin

    def setReceivers(self, receivers):
        self.receivers = receivers
```

В листинге 2.2 представлена реализация канала обслуживания.

Листинг 2.2 — Реализация канала обслуживания

```
from generator import Generator

class Processor(Generator):
    def __init__(self, distribution):
        self.distribution = distribution
        self.maxQueueSize = 0
        self.currentQueueSize  = 0
        self.processedRequests = 0
        self.receivedRequests  = 0
        self.next = 0
        self.receivers = []
```

```
12
13      # Обработка запроса при его наличии
14      def processRequest(self):
15          if self.currentQueueSize > 0:
16              self.processedRequests += 1
17              self.currentQueueSize  -= 1
18
19          if len(self.receivers) != 0:
20              receiverMin = self.receivers[0]
21              for receiver in self.receivers:
22                  if receiver.currentQueueSize < receiverMin.currentQueueSize:
23                      receiverMin = receiver
24
25              receiverMin.receiveRequest()
26              receiverMin.next = self.next + receiverMin.nextTime()
27
28      # Добавление реквеста в очередь
29      def receiveRequest(self):
30          self.currentQueueSize += 1
31          self.receivedRequests += 1
32
33          if self.maxQueueSize < self.currentQueueSize:
34              self.maxQueueSize = self.currentQueueSize
35
36      def nextTime(self):
37          return self.distribution.generate()
38
39      def setReceivers(self, receivers):
40          self.receivers = receivers
```

В листинге 2.3 представлена реализация моделирования работы электронной очереди.

Листинг 2.3 — Реализация моделирования работы электронной очереди

```
1   from processor import Processor
2
3   class EventModel:
4       def __init__(self, generator, terminals, windows):
5           self.generator = generator
6           self.terminals = terminals
7           self.windows = windows
8
9       def run(self):
10          generator = self.generator
11          generator.next = generator.nextTime()
```

```
        self.terminals[0].next = self.terminals[0].nextTime()

        blocks = [generator] + self.windows + self.terminals

        numbRequests = generator.numbRequests
        count = 0
        while count < numbRequests:
            # Находим наименьшее время
            currentTime = generator.next
            for block in blocks:
                if 0 < block.next < currentTime:
                    currentTime = block.next

            for block in blocks:
                # Событие наступило для этого блока
                if currentTime == block.next:
                    if not isinstance(block, Processor): # для генератора
                        # Проверяем, может ли оператор обработать
                        nextGenerator = generator.generateRequest()
                        if nextGenerator is not None:
                            nextGenerator.next = currentTime +
                            ↪  nextGenerator.nextTime()

                        generator.next = currentTime + generator.nextTime()
                    else:
                        block.processRequest()
                        if block.currentQueueSize == 0:
                            block.next = 0
                        else:
                            block.next = currentTime + block.nextTime()

            count = 0
            for computer in self.windows:
                count += computer.processedRequests

        data = []
        for i in range(len(self.terminals)):
            data.append(["Терминал " + str(i + 1),
                self.terminals[i].maxQueueSize,
                self.terminals[i].processedRequests])

        for i in range(len(self.windows)):
            data.append(["Окно обслуживания " + str(i + 1),
                self.windows[i].maxQueueSize,
                self.windows[i].processedRequests])

        return currentTime, data
```

2.2 Демонстрация работы программы

На рисунке 2.1 представлен пример работы программы.

Рисунок 2.1 – Результат работы программы