|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**ОТЧЕТ**

*к лабораторной работе №6*

*По курсу: «Моделирование»*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ***ИУ7И-76Б*** |  |  | **Нгуен Ф. С.** |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | |  | | --- | | **Рудаков И.В.** | |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

*Москва, 2021 г.*

Оглавление

[**I.** **Задача** 3](#_Toc91729505)

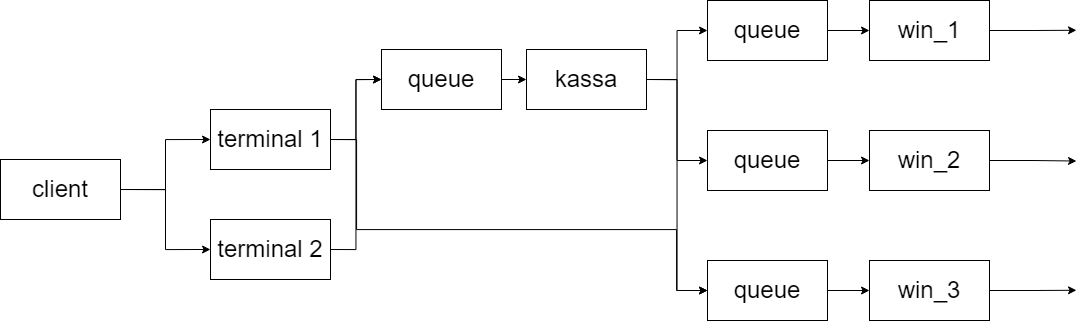
[**II.** **Экспериментальная часть** 3](#_Toc91729506)

[**III.** **Код программы:** 4](#_Toc91729507)

1. Задача

В пункте выдачи приходят клиенты каждые 5±2 минуты. Каждому клиенту необходимо получить талон на одном из двух терминалов. Каждый терминал выдает талон с интервалом в 2±1 минуты. Если в очереди к терминалу находится больше 4 человек, клиент уходит. С вероятность 10% терминал не будет работать из-за технических неполадок, и тогда клиенту будет отказано. Клиенты приходят по равномерному распределению.

Если клиент не заплатил за заказ, он идет в кассу, а затем в окно выдачи. Если клиент уже заплатил, нет необходимости проходить кассу, а идти прямо к окну выдачи. . На кассе время работы с клиентом составляет 10 ± 5 минуты. вероятность невыполнения платежа (из-за ошибки клиента или технической ошибки) 0.05, тогда клиент отклоняется. Всего есть 3 окон, которые работают 10±5, 15±5, 15±10 минут. Вероятность отсутствия товара на складе 0.05, тогда клиент отклоняется.



1. Экспериментальная часть

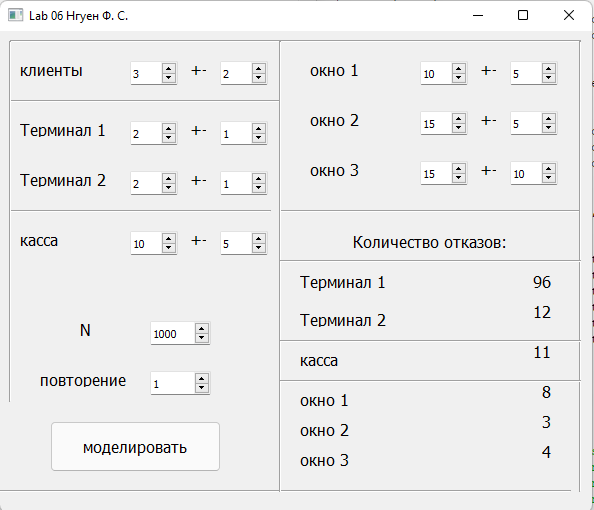


Рисунок 1. пример работы программы

1. **Код программы:**

from numpy import random as nr

from random import shuffle, random

from PyQt5 import QtWidgets, uic

from PyQt5.QtWidgets import QTableWidgetItem

from PyQt5.QtGui import QPen, QColor, QImage, QPixmap, QPainter

from PyQt5.QtCore import Qt, QTime, QCoreApplication, QEventLoop, QPointF

class RandomGenerator:

def \_\_init\_\_(self, begin, delta=0):

self.begin = begin

self.d = delta

def generate(self):

if (self.d == 0):

return self.begin

return nr.uniform(self.begin - self.d, self.begin + self.d)

class GenerateRequest:

def \_\_init\_\_(self, generator, name, count):

self.random\_generator = generator

self.num\_requests = count

self.receivers = []

self.next = 0

self.name = name

def generate\_request(self):

self.num\_requests -= 1

for receiver in self.receivers:

if receiver.receive\_request():

return receiver

return None

def delay(self):

return self.random\_generator.generate()

class ProcessRequest:

def \_\_init\_\_(self, generator, name, ban\_probality=0, max\_queue\_size=-1, end=False):

self.random\_generator = generator

self.queue = 0

self.received = 0

self.max\_queue = max\_queue\_size

self.processed = 0

self.next = 0

self.receivers = []

self.end = end

self.name = name

self.ban\_probality = ban\_probality

def receive\_request(self):

if self.max\_queue == -1 or self.max\_queue > self.queue:

self.queue += 1

self.received += 1

return True

return False

def process\_request(self):

if nr.sample() < self.ban\_probality:

return "ERR"

if self.queue > 0:

self.queue -= 1

self.processed += 1

shuffle(self.receivers)

for receiver in self.receivers:

if receiver.receive\_request():

return receiver

return None

def delay(self):

return self.random\_generator.generate()

class Model:

def \_\_init\_\_(self, clients, terminals, kassa, wins):

self.clients = clients

self.terminals = terminals

self.kassa = kassa

self.wins = wins

def event\_mode(self):

clients = self.clients

clients.receivers = self.terminals.copy()

self.terminals[0].receivers = [self.kassa, self.wins[0], self.wins[1], self.wins[2]]

self.terminals[1].receivers = [self.kassa, self.wins[0], self.wins[1], self.wins[2]]

self.kassa.receivers = [self.wins[0], self.wins[1], self.wins[2]]

clients.next = clients.delay()

self.terminals[0].next = self.terminals[0].delay()

self.terminals[1].next = self.terminals[1].delay()

blocks = []

blocks += [clients]

blocks += self.terminals

blocks += [self.kassa]

blocks += self.wins

refusals = {}

for block in blocks:

refusals[block.name] = 0

while clients.num\_requests >= 0:

current\_time = clients.next

for block in blocks:

if 0 < block.next < current\_time:

current\_time = block.next

for block in blocks:

if current\_time == block.next:

if not isinstance(block, ProcessRequest):

next\_clients = clients.generate\_request()

if next\_clients is not None:

next\_clients.next = current\_time + next\_clients.delay()

else:

refusals[block.name] += 1

clients.next = current\_time + clients.delay()

else:

next\_process = block.process\_request()

if block.queue == 0:

block.next = 0

else:

block.next = current\_time + block.delay()

if next\_process == "ERR":

refusals[block.name] += 1

continue

if block.end:

continue

if next\_process is not None:

next\_process.next = \

current\_time + next\_process.delay()

else:

refusals[block.name] += 1

return refusals

class Window(QtWidgets.QMainWindow):

def \_\_init\_\_(self):

QtWidgets.QWidget.\_\_init\_\_(self)

uic.loadUi("window.ui", self)

self.BtnModeling.clicked.connect(lambda: startModeling(self))

def startModeling(win):

print("Getting Data")

clients\_number = win.n.value()

repeat = win.repeat.value()

#client

time\_client = win.time\_client.value()

d\_time\_client = win.d\_time\_client.value()

print("Get Client OK {} {}".format(time\_client, d\_time\_client))

#terminal

time\_ter\_1 = win.time\_ter\_1.value()

d\_time\_ter\_1 = win.d\_time\_ter\_1.value()

print("Test")

time\_ter\_2 = win.time\_ter\_2.value()

d\_time\_ter\_2 = win.d\_time\_ter\_2.value()

print("Get Terminate OK")

#kassa

time\_kassa = win.time\_kassa.value()

d\_time\_kassa = win.d\_time\_kassa.value()

print("Get Kasssa OK")

#wins

time\_win\_1 = win.time\_win\_1.value()

d\_time\_win\_1 = win.d\_time\_win\_1.value()

time\_win\_2 = win.time\_win\_2.value()

d\_time\_win\_2 = win.d\_time\_win\_2.value()

time\_win\_3 = win.time\_win\_3.value()

d\_time\_win\_3 = win.d\_time\_win\_3.value()

print("Get Data OK")

clients = GenerateRequest(RandomGenerator(time\_client, d\_time\_client), "entry", clients\_number)

terminals = [

ProcessRequest(RandomGenerator(time\_ter\_1, d\_time\_ter\_1), "terminal1", ban\_probality=0.1, max\_queue\_size=3),

ProcessRequest(RandomGenerator(time\_ter\_2, d\_time\_ter\_2), "terminal2", ban\_probality=0.1, max\_queue\_size=3),

]

kassa = ProcessRequest(RandomGenerator(time\_kassa, d\_time\_kassa), "kassa", 0.05, max\_queue\_size=10)

wins = [

ProcessRequest(RandomGenerator(time\_win\_1, d\_time\_win\_1), "win1", 0.05, end=True),

ProcessRequest(RandomGenerator(time\_win\_2, d\_time\_win\_2), "win2", 0.05, end=True),

ProcessRequest(RandomGenerator(time\_win\_3, d\_time\_win\_3), "win3", 0.05, end=True),

]

model = Model(clients, terminals, kassa, wins)

result = model.event\_mode()

win.res\_ter\_1.setText("{}".format(result['terminal1']))

win.res\_ter\_2.setText("{}".format(result['terminal2']))

win.res\_win\_1.setText("{}".format(result['win1']))

win.res\_win\_2.setText("{}".format(result['win2']))

win.res\_win\_3.setText("{}".format(result['win3']))

win.res\_kassa.setText("{}".format(result['kassa']))

#print(result)

print()

print('Количество отказов:')

print('Терминал 1: %d' % (result['terminal1']))

print('Терминал 2: %d' % (result['terminal2']))

print('касса : %d' % (result['kassa']))

print('окно 1: %d' % (result['win1']))

print('окно 2: %d' % (result['win2']))

print('окно 3: %d' % (result['win3']))

if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QtWidgets.QApplication(sys.argv)

w = Window()

w.show()

sys.exit(app.exec\_())