

Московский государственный университет имени М. В. Ломоносова

Факультет вычислительной математики и кибернетики

Отчет по заданию практикума

## **Моделирование обслуживания в филиале банка**

студент 4 курса 424 группы

*Зуев Кирилл Александрович*

Москва, 2018

# Содержание

1	Уточнение постановки задачи	2
2	Диаграмма основных классов	4
3	Спецификации интерфейса	5
4	Диаграмма объектов	11
5	Инструментальные средства	12
6	Описание файловой структуры системы	12
7	Пользовательский интерфейс	13

# 1 Уточнение постановки задачи

Необходимо создать компьютерную модель обслуживания потока заявок, поступающих от клиентов банка, несколькими клерками ( $2 \leq N \leq 7$ ) в одном из филиалов банка. Известно недельное расписание работы филиала банка: 5 дней по 8 часов и один день — 6 часов, возможны перерывы на обед.

При моделировании работы заявки на обслуживание (т.е. приход клиентов) поступают случайным образом. Случайной величиной является отрезок времени между последовательным появлением двух заявок (например, от 0 до 10 минут). Длительность обслуживания каждой заявки — также случайное число в некотором диапазоне (например, от 2 до 30 минут), но длительность не зависит от входного потока заявок. Еще одна случайная величина — прибыль, получаемая банком от обслуживания клиента, она варьируется в пределах от 3 тыс. до 50 тыс. рублей.

Поступившие заявки (клиенты) образуют общую очередь, максимальная длина которой —  $K$  человек ( $10 \leq K \leq 25$ ). Если очередь достигла такой длины, то вновь прибывающие клиенты уходят — тем самым банк теряет своих потенциальных клиентов.

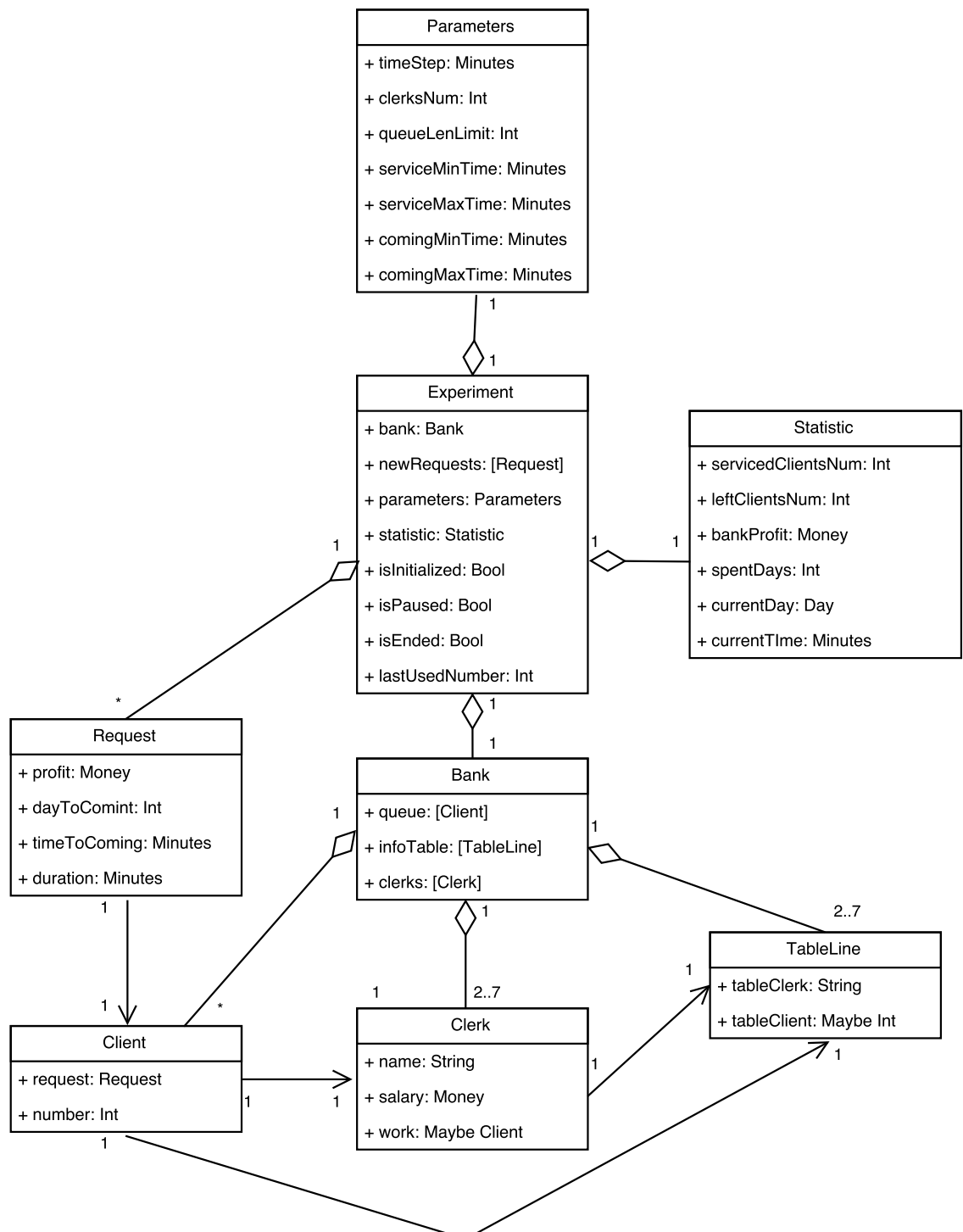
Клиенты банка ожидают своей очереди на обслуживание в общем зале с информационным табло, на котором высвечиваются номер клиента, взятого только что на обслуживание, и номер места клерка, обслуживающего этого клиента. В каждый день работы филиала заявки на обслуживание нумеруются последовательно, начиная с 1, по мере их прихода в банк.

Цель моделирования работы банка — определение прибыли банка и ее зависимости от числа работающих клерков; выявление «узких» мест в работе банка: нехватки клерков (возможное следствие этого — потеря клиентов), простой клерков (следствие — лишние траты на их зарплату). Прибыль высчитывается с учетом дневной зарплаты каждого клерка (2 тыс. руб.).

Период моделирования — месяц, шаг моделирования интервал времени от 10 минут до 1 часа. Следует включить в параметры моделирования: числа  $N$  и  $K$ , шаг моделирования, диапазоны разброса случайных величин промежутка между приходом клиентов и время их обслуживания.

Визуализация моделируемого процесса должна предусматривать показ текущей ситуации в банке, в том числе скопившуюся очередь, занятость клерков, появление новых и уход обслуженных клиентов, информационное табло. Следует предусмотреть вывод в ходе моделирования и по его окончании подсчитанной статистики: количества обслуженных и потерянных клиентов, а также полученную банком прибыль.

## 2 Диаграмма основных классов



### 3 Спецификации интерфейса

```
-- * Эксперимент
data Experiment = Experiment
{ bank          :: Bank          -- ^ банк
, newRequests   :: [Request]     -- ^ поток поступающих заявок
, parameters    :: Parameters    -- ^ параметры
, statistic     :: Statistic     -- ^ статистика
, isInitialized :: Bool          -- ^ инициализирован ли
, isPaused      :: Bool          -- ^ поставлен ли на паузу
, isEnded       :: Bool          -- ^ закончен ли
, lastUsedNumber :: Int          -- ^ последний использованный номер клиента
} deriving (Eq)

class ExperimentClass experiment where
-- | Инициализировать эксперимент
initExperiment      :: experiment
-- | Изменить параметр
changeParameter     :: ParametersField -> ChangeAction -> experiment -> experiment
-- | Начать эксперимент
startExperiment     :: StdGen -> experiment -> experiment
-- | Продолжить/приостановить эксперимент
playPauseExperiment :: experiment -> experiment
-- | Сбросить эксперимент
resetExperiment     :: experiment
-- | Перейти к концу эксперимента
finishExperiment    :: experiment -> experiment
-- | Добавить заявки в очередь в виде клиентов и обновить статистику.
-- Учитываются ушедшие клиенты.
addToQueue          :: [Request] -> experiment -> experiment
-- | Уход клиентов из очереди
leftFromQueue       :: experiment -> experiment
```

```

-- | Занять при возможности свободных клерков
setWorkToClerks      :: experiment -> experiment
-- | Обновить очередь клиентов
updateClients        :: experiment -> experiment
-- | Обновить состояние клерков
updateClerks         :: experiment -> experiment
-- | Добавить клиентов в очередь.
addClients           :: [Request] -> experiment -> experiment
-- | Моделирование заданного промежутка времени.
addTime              :: Minutes -> experiment -> experiment

```

```

-- * Банк

```

```

data Bank = Bank
{ queue      :: [Client]      -- ^ очередь клиентов
, infoTable  :: [TableLine]   -- ^ информационное табло
, clerks     :: [Clerk]       -- ^ клерки
} deriving (Eq)

```

```

class BankClass bank_ where
-- | Инициализировать банк
initBank :: bank_
-- | Добавить клерка
addClerk :: bank_ -> bank_
-- | Удалить клерка
delClerk :: bank_ -> bank_

```

```

-- * Клиент
data Client = Client
{ request      :: Request -- ^ заявка
, number       :: Int      -- ^ номер клиента
} deriving (Eq)

class ClientClass client where
-- | Вычесть 1 мин. из длительности обработки заявки
subtractDuration :: client -> client
-- | Завершение обслуживания при исчерпании длительности обработки заявки
completeService  :: Maybe client -> Maybe client

-- * Заявка
data Request = Request
{ profit       :: Money    -- ^ прибыль
, dayToComing  :: Int      -- ^ дней до прихода
, timeToComing :: Minutes  -- ^ времени до прихода (в день прихода)
, duration     :: Minutes  -- ^ длительность
} deriving (Eq)

class RequestClass request_ where
-- | Инициализировать поток заявок
initNewRequests :: [request_]
-- | Сгенерировать поток заявок
genNewRequests  :: StdGen -> Parameters -> [request_]
-- | Создать заявку
mkRequest       :: Money -> Int -> Minutes -> Minutes -> request_

```



```

-- * Строка информационного табла
data TableLine = TableLine
{ tableClerk  :: String      -- ^ имя клерка
, tableClient :: Maybe Int   -- ^ номер клиента (при наличии)
} deriving (Eq)

```

```

class TableLineClass tableLine where
-- | Инициализировать информационное табло
initInfoTable  :: [tableLine]
-- | Удалить строку информационного табла
delTableLine   :: Clerk -> [tableLine] -> [tableLine]
-- | Добавить строку информационного табла
addTableLine   :: Clerk -> [tableLine] -> [tableLine]
-- | Инициализировать строку информационного табла
initTableLine  :: Clerk -> tableLine
-- | Обновить информационное табло
updateInfoTable :: [Clerk] -> [tableLine] -> [tableLine]

```

```

-- * Клерк
data Clerk = Clerk
{ name    :: String      -- ^ имя
, salary  :: Money       -- ^ зарплата
, work    :: Maybe Client -- ^ обслуживаемый клиент
} deriving (Eq)

```

```

class ClerkClass clerk where
-- | Инициализировать клерков
initClerks      :: [clerk]

```

```

-- | Начать обслуживание клиента клерком
takeClientToClerk :: Client -> clerk -> clerk

-- | Обновить время обслуживание
serviceTime      :: clerk -> clerk

-- | Завершение обслуживания при исчерпании длительности обработки заявки
serviceComplete  :: clerk -> clerk

-- | Не завершилось ли обслуживание клиента
isServiced       :: clerk -> Bool

-- | Полученная прибыль от обслуживания заявки
serviceProfit     :: clerk -> Money

-- | Свободен ли клерк
withoutWork       :: clerk -> Bool

-- | Список доступных банку клерков (по умолчанию)
defaultClerks     :: [clerk]

-- | Создать клерка
mkClerk           :: String -> clerk


-- * Параметры
data Parameters = Parameters
{ timeStep        :: Minutes -- ^ шаг моделирования (мин.)
, simulationPeriod :: Int     -- ^ период моделирования (дней)
, clerksNum       :: Int     -- ^ число клерков
, queueLenLimit   :: Int     -- ^ максимальная длина очереди
, serviceMinTime  :: Minutes -- ^ минимальное время обслуживания (мин.)
, serviceMaxTime  :: Minutes -- ^ максимальное время обслуживания (мин.)
, comingMinTime   :: Minutes -- ^ минимальное время между приходом клиентов
, comingMaxTime   :: Minutes -- ^ максимальное время между приходом клиентов
} deriving (Eq)

```

```

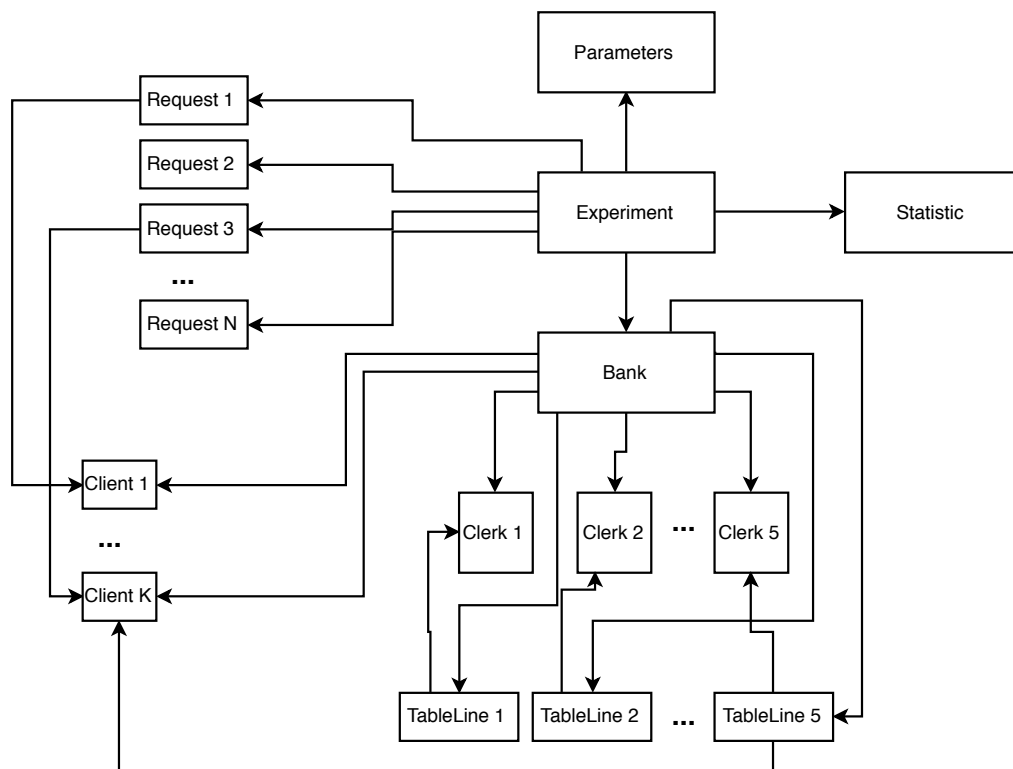
class ParametersClass parameters_ where
-- | Инициализация параметров
initParameters :: parameters_

-- * Статистика
data Statistic = Statistic
{ servicedClientsNum    :: Int      -- ^ число обслужанных клиентов
, leftClientsNum        :: Int      -- ^ число потерянных клиентов
, bankProfit             :: Money    -- ^ прибыль банка
, spentDays              :: Int      -- ^ число пройденных дней
, currentDay             :: Day      -- ^ текущий день
, currentTime           :: Minutes  -- ^ текущее время
} deriving (Eq)

class StatisticClass statistic_ where
-- | Инициализация статистики
initStatistic :: statistic_

```

## 4 Диаграмма объектов



## 5 Инструментальные средства

Язык разработки: *Haskell*

Используемые библиотеки: *Miso*

## 6 Описание файловой структуры системы

**Main.hs** — точка входа для запуска проекта;

**Experiment.hs** — обработка событий, отрисовка интерфейса;

**Model.hs** — описание классов и реализация их методов;

**Constants.hs** — используемые константы.

## 7 Пользовательский интерфейс

Пользовательский интерфейс представляет собой единое окно для настройки, визуализации банка и отображения статистики эксперимента.

На экране отображены основные параметры, кнопки для их изменения и кнопки для управления экспериментом:

Число клерков (2 - 7):  
- 2 +

Шаг моделирования (10 - 60):  
- 10 мин. +

Период моделирования (7 - 28):  
- 7 дней +

Максимальный размер очереди (10 - 25):  
- 10 +

Время обслуживания (2 - 30):  
От: - 2 мин. +  
До: - 30 мин. +

Время между приходом клиентов (0 - 10):  
От: - 0 мин. +  
До: - 10 мин. +

Инициализировать   Продолжить   Сброс  
Сделать шаг   Завершить эксперимент

Текущее время: 10 : 00  
Дней прошло: 0  
Текущий день недели: Пн.  
Число обслуженных клиентов: 0  
Число потерянных клиентов: 0  
Прибыль банка: 0 тыс. руб.

Clerk 1

Clerk 2

Пн.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Вт.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Ср.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Чт.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Пт.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Сб.: 10 : 00 - 17 : 00 (обед: 13 : 00 - 14 : 00)

Clerk 1: свободен  
Clerk 2: свободен

Стоит отметить, что после инициализации эксперимента изменять параметры становится невозможным, за исключением параметра «Шаг моделирования».

Для начала эксперимента пользователю нужно нажать кнопку «Инициализировать», а затем кнопку «Продолжить», и эксперимент начнет моделироваться в реальном времени (1 мин. ~ 1 сек.):

Число клерков (2 - 7):

Шаг моделирования (10 - 60):

Период моделирования (7 - 28):

Максимальный размер очереди (10 - 25):

Время обслуживания (2 - 30):  
От:  До:

Время между приходом клиентов (0 - 10):  
От:  До:

Инициализировать

Приостановить

Сброс

Сделать шаг

Завершить эксперимент

Текущее время: 11 : 07  
Дней прошло: 8  
Текущий день недели: Вт.  
Число обслужанных клиентов: 634  
Число потерянных клиентов: 5  
Прибыль банка: 16743 тыс. руб.

Clerk 1  
  
010  


Clerk 2  
  
011  


Clerk 3  
  
—  


Clerk 4  
  
009  


Clerk 5  
  
012  




Пн.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Вт.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Ср.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Чт.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Пт.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Сб.: 10 : 00 - 17 : 00 (обед: 13 : 00 - 14 : 00)

Clerk 3: свободен  
Clerk 5: 012  
Clerk 2: 011  
Clerk 1: 010  
Clerk 4: 009

Эксперимент можно приостановить, нажав кнопку «Приостановить»:

Число клерков (2 - 7):  

- 5 +

Шаг моделирования (10 - 60):  

- 60 мин. +

Период моделирования (7 - 28):  

- 28 дней +

Максимальный размер очереди (10 - 25):  

- 15 +

Время обслуживания (2 - 30):  
От: 

- 10 мин. +

  
До: 

- 22 мин. +

Время между приходом клиентов (0 - 10):  
От: 

- 2 мин. +

  
До: 

- 6 мин. +

Инициализировать

Продолжить

Сброс

Сделать шаг

Завершить эксперимент

Текущее время: 16 : 00  
Дней прошло: 0  
Текущий день недели: Пн.  
Число обслужанных клиентов: 73  
Число потерянных клиентов: 1  
Прибыль банка: 1932 тыс. руб.

Clerk 1

072

Clerk 2

070

Clerk 3

071

Clerk 4

073

Clerk 5

074

075

076

Пн.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Вт.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Ср.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Чт.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Пт.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Сб.: 10 : 00 - 17 : 00 (обед: 13 : 00 - 14 : 00)

Clerk 5: 074  
Clerk 4: 073  
Clerk 1: 072  
Clerk 3: 071  
Clerk 2: 070

В данном состоянии можно смоделировать заданный в параметре «Шаг моделирования» промежуток времени, нажав кнопку «Сделать шаг», смоделировать весь эксперимент до конца, нажав кнопку «Завершить эксперимент», или же продолжить моделирование в реальном времени, нажав кнопку «Продолжить»



В конце эксперимента будет выдан окончательный результат в таблице со статистикой:

Число клерков (2 - 7):  
- 5 +

Шаг моделирования (10 - 60):  
- 60 мин. +

Период моделирования (7 - 28):  
- 28 дней +

Максимальный размер очереди (10 - 25):  
- 15 +


Время обслуживания (2 - 30):  
От: - 10 мин. +  
До: - 22 мин. +


Время между приходом клиентов (0 - 10):  
От: - 2 мин. +  
До: - 6 мин. +


Инициализировать  
Сделать шаг


Продолжить  
Завершить эксперимент


Сброс

Clerk 1  


Clerk 2  


Clerk 3  


Clerk 4  


Clerk 5  


Пн.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Вт.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Ср.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Чт.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Пт.: 10 : 00 - 19 : 00 (обед: 14 : 00 - 15 : 00)  
Сб.: 10 : 00 - 17 : 00 (обед: 13 : 00 - 14 : 00)

Clerk 1: свободен  
Clerk 3: свободен  
Clerk 4: свободен  
Clerk 2: свободен  
Clerk 5: свободен

Текущее время: 00 : 00  
Дней прошло: 28  
Текущий день недели: Пн.  
Число обслужанных клиентов: 2598  
Число потерянных клиентов: 5  
Прибыль банка: 68352 тыс. руб.

В любой момент эксперимента его можно сбросить к первоначальной конфигурации, нажав кнопку «Сброс».

16