ГОУ ВПО

Уфимский государственный авиационный технический университет

Кафедра вычислительной математики и кибернетики

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
| **100** |  |  |  |  |  |  |  |  |  |  |  |  |
| **90** |  |  |  |  |  |  |  |  |  |  |  |  |
| **80** |  |  |  |  |  |  |  |  |  |  |  |  |
| **70** |  |  |  |  |  |  |  |  |  |  |  |  |
| **60** |  |  |  |  |  |  |  |  |  |  |  |  |
| **50** |  |  |  |  |  |  |  |  |  |  |  |  |
| **40** |  |  |  |  |  |  |  |  |  |  |  |  |
| **30** |  |  |  |  |  |  |  |  |  |  |  |  |
| **20** |  |  |  |  |  |  |  |  |  |  |  |  |
| **10** |  |  |  |  |  |  |  |  |  |  |  |  |

***Разработка приложения***

**«Моделирование работы остановки городского транспорта»**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе**

**по дисциплине *«Объектно-ориентированное программирование»***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа | Ф.И.О. | Подпись | Дата | Оценка |
| Студент | Доронин С.Г. |  |  |  |
| Консультант |  |  |  |  |
| Принял |  |  |  |  |

Уфа - 2013 г.

Федеральное агентство по образованию

Государственное образовательное учреждение высшего профессионального образования

«УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра вычислительной математики и кибернетики

ЗАДАНИЕ

На курсовую работу по дисциплине «Базы данных»

Студент \_\_Доронин С.Г. Группа \_ПРО-201в Консультант\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ф.И.О номер акад. гр. Ф.И.О. .

1. Тема курсовой работы

Моделирование работы остановки городского транспорта

|  |
| --- |
| наименование темы |
|  |

|  |  |
| --- | --- |
| 1. Основное содержание: | Разработка информационной модели предметной области, |
| разработка программного продукта, создание программной документации | |
|  | |

1. Требования к оформлению
   1. Техническое задание для программного продукта должно быть оформлено согласно ГОСТу 19.201-78 «Техническое задание. Требования к содержанию и оформлению».
   2. Руководство пользователя (оператора) разрабатывается и оформляется согласно ГОСТу19.505-79 «Руководство оператора. Требования к содержанию и оформлению».
   3. В пояснительной записке должны содержаться следующие разделы:

* Введение – в этой части описывается цель курсовой работы, требования к разрабатываемой базе данных и приложению.
* Техническое задание, оформленное согласно ГОСТу.
* Разработка приложения – в этой части должна быть представлена структура модулей программы и их взаимосвязь, описана схема работы с базой данных на уровне компонентов, описаны особенности реализации приложения.
* Руководство пользователя, оформленное согласно ГОСТу.
* Заключение – эта часть содержит описание основных результатов работы, сферу его применения.

Дата выдачи \_\_\_\_\_\_\_\_\_\_\_\_ Дата окончания \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Муслимова Г.Р.

Подпись

# Введение

В ЭГТИ используется большое количество площадок и рабочих для добычи нефти, в связи с этим для эффективной деятельности требуется построение графика работы сотрудников на площадках.

ГОУ ВПО

Уфимский государственный авиационный технический университет

Кафедра вычислительной математики и кибернетики

**Программный продукт**

**«Моделирование работы остановки городского транспорта»**

# Техническое задание

**Листов 6**

2013

***Аннотация***

Программный продукт (ПП) «Моделирование работы остановки городского транспорта» предназначен для эмуляции работы остановки городского транспорта и подсчета статистических данных.

Данный документ описывает:

* наименование, краткую характеристику области применения программного продукта;
* основание для разработки программного продукта;
* назначение разработки, в котором указано функциональное назначение программного продукта;
* требования к программному продукту;
* состав программной документации.

***Введение***

Программа «Моделирование работы остановки городского транспорта» предназначена для эмуляции работы остановки городского транспорта и подсчета статистических данных.

***Основание для разработки***

Вышеназванный программный продукт разрабатывается на основании задания на курсовую работу.

***Назначение разработки***

ПП должен состоять из следующих функциональных модулей:

1. главный модуль, обеспечивающий запуск и остановку эмуляции, а так же вывод статистических данных;
2. Панель вывода анимации движения автобусов и пассажиров.

*Требования к надежности*

«Моделирование работы остановки городского транспорта» должна обеспечивать следующие требования к надежности:

* устойчивая работа системы в целом;
* контроль на ошибочность вводимых данных.

*Требования к составу и параметрам технических средств*

К разрабатываемому программному продукту предъявляются следующие минимальные аппаратные требования:

1. IBM PC-совместимый компьютер с процессором – Intel® Pentium® III или выше;
2. оперативная память – не менее 512Mb;
3. SVGA-монитор с разрешением не ниже 1280x1024;
4. клавиатура;
5. манипулятор типа «мышь»;
6. операционная система Microsoft® Windows® XP (Vista, 7, 8), Linux, MacOS.

*Требования к информационной и программной совместимости*

Программа должна быть реализована на Groovy 2.0.5 в интегрированной среде разработки IntelliJ IDEA 12 с возможностью дальнейшего расширения и переработки с использованием этой среды или ее более поздних версий.

***Требования к программной документации***

Документация на программный продукт должна включать в себя следующие документы: техническое задание, руководство пользователя.

**Исходный код программы**

package com.phpusr.busstop.entity

import com.phpusr.busstop.consts.BusStopConsts

import javax.imageio.ImageIO

import java.awt.image.BufferedImage

/\*\*

\* Created with IntelliJ IDEA.

\* User: phpusr

\* Date: 12.01.13

\* Time: 19:21

\* To change this template use File | Settings | File Templates.

\*/

**/\*\***

**\* Автобус**

**\*/**

**class Bus** {

/\*\* Порядковый № \*/

int number

/\*\* Название \*/

String name

/\*\* Маршрут \*/

String route

/\*\* Кол-во мест \*/

int seatCount

/\*\* Изображение автобуса \*/

BufferedImage image

/\*\* Кол-во Пассажиров (внутри) \*/

int passengerCount

/\*\* Кол-во зашедших и вышедших Пассажиров (для Статистики) \*/

int passengerCountIn, passengerCountOut

Bus(int number, String name, String route, int seatCount, String pathToImage) {

this.number = number

this.name = name

this.route = route

this.seatCount = seatCount

setImage(pathToImage)

}

Bus() {

this(1, 'Test', '777', 10, "$BusStopConsts.BUS\_IMG\_PATH/Pacan.png")

}

void setImage(String path) {

URL url = Bus.class.getResource(path)

image = ImageIO.read(url)

}

/\*\* Возвращает Длину изображения Автобуса \*/

int getWidth() {

image.width

}

/\*\* Возвращает Высоту изображения Автобуса \*/

int getHeight() {

image.height

}

/\*\* Добавить пассажира в автобус \*/

boolean addPassenger() {

if (passengerCount < seatCount) {

passengerCount++

passengerCountIn++

if (BusStopConsts.busLog) println "\t$this"

return true

} else {

if (BusStopConsts.busLog) println ">>В автобусе: $name-$route больше нет места!"

return false

}

}

/\*\* Удалить пассажира из Автобуса \*/

boolean delPassenger() {

if (passengerCount > 0) {

passengerCount--

passengerCountOut++

if (BusStopConsts.busLog) println "\t$this"

return true

} else {

if (BusStopConsts.busLog) println ">>В автобусе: $name-$route больше нет пассажиров!"

return false

}

}

/\*\* Возвращает кол-во Свободных мест в Автобусе \*/

int getFreeSeat() {

return seatCount - passengerCount

}

@Override

String toString() {

return "Автобус: $name-$route, Пассажиров: $passengerCount/$seatCount, Всего зашло: $passengerCountIn, Всего вышло: $passengerCountOut"

}

}

package com.phpusr.busstop.util

import com.phpusr.busstop.consts.BusStopConsts

import com.phpusr.busstop.entity.Bus

import com.phpusr.busstop.frame.BusTableModel

import javax.swing.\*

/\*\*

\* Created with IntelliJ IDEA.

\* User: phpusr

\* Date: 13.01.13

\* Time: 18:01

\* To change this template use File | Settings | File Templates.

\*/

**/\*\***

**\* Утилита для работы с Автобусами**

**\*/**

**class BusParkUtil** {

/\*\* Путь к папке с ихображениями Автобусов \*/

static final String BUS\_IMG\_PATH = BusStopConsts.BUS\_IMG\_PATH

/\*\* Список автобусов \*/

List<Bus> busList = []

BusParkUtil() {

int i = 0

busList << new Bus(i++, 'Pacan', '57', 20, "$BUS\_IMG\_PATH/Pacan.png")

busList << new Bus(i++, 'Boxer', '218', 10, "$BUS\_IMG\_PATH/Boxer.png")

busList << new Bus(i++, 'Dirty', '54', 20, "$BUS\_IMG\_PATH/Dirty.png")

busList << new Bus(i++, 'Feniks', '214', 10, "$BUS\_IMG\_PATH/Feniks.png")

busList << new Bus(i++, 'Tn', '54', 20, "$BUS\_IMG\_PATH/Tn.png")

busList << new Bus(i++, 'Yellow', '39', 15, "$BUS\_IMG\_PATH/Yellow.png")

busList << new Bus(i++, 'White', '101', 15, "$BUS\_IMG\_PATH/White.png")

busList << new Bus(i++, 'Van', '269', 10, "$BUS\_IMG\_PATH/Van.png")

busList << new Bus(i++, 'Merc', '266', 10, "$BUS\_IMG\_PATH/Merc.png")

busList << new Bus(i++, 'TwoFloor', '110', 40, "$BUS\_IMG\_PATH/TwoFloor.png")

busList << new Bus(i++, 'Green', '130', 20, "$BUS\_IMG\_PATH/Green.png")

busList << new Bus(i++, 'GAZ', '243', 15, "$BUS\_IMG\_PATH/GAZ.png")

}

/\*\* Возвращает рандомный автобус из списка \*/

Bus getRandomBus() {

int num = Math.random() \* busList.size()

return busList.get(num)

}

/\*\* Возвращает кол-во всех севших в Автобусы Пассажиров \*/

int getAllPassengerInCount() {

int count = 0

busList.each { bus ->

count += bus.passengerCountIn

}

return count

}

/\*\* Возвращает кол-во всех вышедших из Автобусов Пассажиров \*/

int getAllPassengerOutCount() {

int count = 0

busList.each { bus ->

count += bus.passengerCountOut

}

return count

}

/\*\* Возвращает Автобус, который Больше всех довез Пассажиров \*/

Bus getMaxBus() {

Bus mBus = busList.get(0)

busList.each { bus ->

if (bus.passengerCountOut > mBus.passengerCountOut) {

mBus = bus

}

}

return mBus

}

/\*\* Возвращает Автобус, который Меньше всех довез Пассажиров \*/

Bus getMinBus() {

Bus mBus = busList.get(0)

busList.each { bus ->

if (bus.passengerCountOut < mBus.passengerCountOut) {

mBus = bus

}

}

return mBus

}

/\*\* Вывод статистики по Пассажирам в Автобусах \*/

void printStat() {

println '-------------------------------------------------------------------------------------------------------------------------'

println "Кол-во всех севших в автобусы пассажиров: $allPassengerInCount"

println "Кол-во всех вышедших из автобусов пассажиров: $allPassengerOutCount"

println "Автобус довезший БОЛЬШЕ всех пассажиров: $maxBus"

println "Автобус довезший МЕНЬШЕ всех пассажиров: $minBus"

println '-------------------------------------------------------------------------------------------------------------------------'

}

/\*\* Загрузка начальных значений в таблицу \*/

void initTblStat(BusTableModel model) {

while (model.rowCount > 0) {

model.removeRow(0)

}

busList.each { bus ->

def list = [bus.name, bus.passengerCount, bus.freeSeat, bus.passengerCountOut, bus.passengerCountIn]

model.addRow(list.toArray())

}

}

/\*\* Обновлении информации в Таблице Статистики \*/

void updateTblStat(JTable table, BusTableModel model, Bus bus) {

int i = 0

table.setValueAt(bus.name, bus.number, i++)

table.setValueAt(bus.passengerCount, bus.number, i++)

table.setValueAt(bus.freeSeat, bus.number, i++)

table.setValueAt(bus.passengerCountOut, bus.number, i++)

table.setValueAt(bus.passengerCountIn, bus.number, i++)

model.maxRow = maxBus.number

model.minRow = minBus.number

}

}

package com.phpusr.busstop.frame

import com.phpusr.busstop.consts.BusStopConsts

import com.phpusr.busstop.entity.Bus

import com.phpusr.busstop.util.BusParkUtil

import com.phpusr.busstop.util.DrawUtil

import javax.swing.\*

import java.awt.\*

import java.awt.image.BufferedImage

/\*\*

\* Created with IntelliJ IDEA.

\* User: phpusr

\* Date: 13.01.13

\* Time: 16:21

\* To change this template use File | Settings | File Templates.

\*/

**/\*\***

**\* Панель для отображения симуляции Автобусной остановки**

**\*/**

**class BusStopPanel extends JPanel** {

/\*\* Координаты рисования Пассажиров \*/

int xPosBus, yPosPassenger

/\*\* Кол-во Пассажиров для Выхода и Входа \*/

int passengerCountOut, passengerCountOutConst, passengerCountIn, passengerCountInConst

/\*\* Кол-во остановок \*/

int stopCount

/\*\* Буфер для рисования кадра \*/

Image scrnBuf

/\*\* Холст для рисования от Буфера \*/

Graphics scrnG

/\*\* Утилита для работы с автобусами \*/

BusParkUtil busParkUtil

/\*\* Утилита для рисования объектов \*/

DrawUtil drawUtil

/\*\* Текущий Автобус \*/

Bus bus

/\*\* Форма управления \*/

ControlFrame controlFrame

/\*\* Стоит-ли Автобус на остановке \*/

boolean stop

/\*\* Выходят или Заходят в автобус Пассажиры \*/

boolean out

/\*\* Пауза \*/

boolean pause

BusStopPanel() {

init()

setSize(BusStopConsts.WIDTH, BusStopConsts.HEIGHT)

setVisible(true)

}

/\*\* Инициализация переменных для Рисования \*/

private void init() {

scrnBuf = new BufferedImage(BusStopConsts.WIDTH, BusStopConsts.HEIGHT, BufferedImage.TYPE\_INT\_RGB)

scrnG = scrnBuf.getGraphics()

busParkUtil = new BusParkUtil()

drawUtil = new DrawUtil()

bus = busParkUtil.randomBus

xPosBus = -1 \* bus.width

stop = false

out = true

}

/\*\* Функция Рисования \*/

void paint(Graphics g) {

//Рисование фона

drawUtil.drawBackground(scrnG, BusStopConsts.WIDTH, BusStopConsts.HEIGHT, this)

//Рисование автобуса

drawUtil.drawBus(scrnG, bus, xPosBus, BusStopConsts.Y\_POS, this)

//Если автобус подъехал к остановке

int stopX = BusStopConsts.WIDTH / 2 - bus.width / 2

if (xPosBus >= stopX && xPosBus <= stopX+BusStopConsts.PIXEL\_INC) {

if (!stop) { //Если зашли в это условие первый раз, то генерируем кол-во входящих пассажиров

stopCount++

passengerCountOut = Math.round(Math.random() \* bus.passengerCount)

passengerCountOutConst = passengerCountOut

passengerCountIn = Math.round(Math.random() \* bus.freeSeat)

passengerCountInConst = passengerCountIn

out = true

yPosPassenger = BusStopConsts.Y\_POS + bus.height/2

if (BusStopConsts.busLog) println '-----------------------------------------------------------------------------------'

if (BusStopConsts.paintLog) println "$stopCount\t Кол-во выходящих пассажиров: ${passengerCountOut}/$bus.passengerCount,\t\t Кол-во входящих пассажиров: ${passengerCountIn}/$bus.freeSeat"

if (BusStopConsts.busLog) println '-----------------------------------------------------------------------------------'

if (BusStopConsts.statBusLog && stopCount % 10 == 0) busParkUtil.printStat()

updateBusInfo()

}

if (out) { //Анимация выхода пассажиров

stop = true

if (passengerCountOut > 0) {

drawUtil.drawPassenger(scrnG, (int)BusStopConsts.WIDTH/2, yPosPassenger)

yPosPassenger += BusStopConsts.PIXEL\_INC

if (yPosPassenger > BusStopConsts.HEIGHT) { //Пассажир вышел из Автобуса

passengerCountOut--

bus.delPassenger()

yPosPassenger = BusStopConsts.Y\_POS + bus.height/2

updateBusInfo()

}

} else {

out = false

yPosPassenger = BusStopConsts.HEIGHT

}

} else { //Анимация входа пассажиров

if (passengerCountIn > 0) {

drawUtil.drawPassenger(scrnG, (int)BusStopConsts.WIDTH/2, yPosPassenger)

yPosPassenger -= BusStopConsts.PIXEL\_INC

if (yPosPassenger < BusStopConsts.Y\_POS + bus.height/2) { //Пассажир сел на Автобус

passengerCountIn--

bus.addPassenger()

yPosPassenger = BusStopConsts.HEIGHT

updateBusInfo()

}

} else {

stop = false

}

}

}

//Если автобус не остановлен, то движение автобуса

if (!stop) xPosBus += BusStopConsts.PIXEL\_INC

//Если автобус уехал за пределы

if (xPosBus > BusStopConsts.WIDTH) {

bus = busParkUtil.randomBus

xPosBus = -1 \* bus.width

}

//Рисование на форме изображения из буфера

g.drawImage(scrnBuf, 0, 0, this)

}

/\*\* Срабатывает при Изменении свойств в Автобусе \*/

private updateBusInfo() {

controlFrame.setPassengerCountOut(passengerCountOutConst - passengerCountOut, passengerCountOutConst, bus.passengerCountOut, busParkUtil.allPassengerOutCount)

controlFrame.setPassengerCountIn(passengerCountInConst - passengerCountIn, passengerCountInConst, bus.passengerCountIn, busParkUtil.allPassengerInCount)

busParkUtil.updateTblStat(controlFrame.tblStat, controlFrame.model, bus)

}

void update(Graphics g) { paint(g) }

/\*\* Запуск Рисования в отдельном потоке \*/

void start() {

pause = false

new Thread (new Runnable() {

@Override

public void run() {

if (BusStopConsts.paintLog) println 'start'

while(!pause) {

repaint()

try {

Thread.sleep(BusStopConsts.PAUSE\_MILIS)

} catch (InterruptedException e){}

}

if (BusStopConsts.paintLog) println 'end start'

}

}).start()

}

/\*\* Пауза Рисования \*/

void pause() {

pause = true

}

/\*\* Остановка Рисования \*/

void stop() {

pause()

init()

repaint()

}

}

# Разработка приложения

Программа состоит из следующих функциональных модулей:

1. главный модуль, обеспечивающий запуск и остановку эмуляции, а так же вывод статистических данных;
2. Панель вывода анимации движения автобусов и пассажиров.

Программа написана на языке Groovy 2.0.5, который создан на основе языка Java.

Groovy использует все возможности Java и позволяет писать меньше кода, что увеличивает скорость разработки и сокращает размер исходного кода.

ГОУ ВПО

Уфимский государственный авиационный технический университет

Кафедра вычислительной математики и кибернетики

**Программный продукт**

**«Моделирование работы остановки городского транспорта»**

**Руководство пользователя**

**Листов 12**

2013

***Аннотация***

Программный продукт (ПП) «Моделирование работы остановки городского транспорта» предназначен для эмуляции работы остановки городского транспорта и подсчета статистических данных.

Данный документ описывает:

* наименование, краткую характеристику области применения программного продукта;
* основание для разработки программного продукта;
* назначение разработки, в котором указано функциональное назначение программного продукта;
* требования к программному продукту;
* состав программной документации.

***Назначение программы***

Программный продукт (ПП) «Моделирование работы остановки городского транспорта» предназначен для эмуляции работы остановки городского транспорта и подсчета статистических данных.

***Условия выполнения программы***

Для выполнения программой заданных функций, ей необходимо следующее программно-аппаратной окружение:

1. IBM PC-совместимый компьютер с процессором – Intel® Pentium® III или выше;
2. оперативная память – не менее 512Mb;
3. SVGA-монитор с разрешением не ниже 1280x1024;
4. клавиатура;
5. манипулятор типа «мышь»;
6. веб-браузер Google Chrome, Opera, Firefox, IE (не ниже 10 версии);
7. операционная система Microsoft® Windows® XP (Vista, 7, 8), Linux, MacOS.

***Выполнение программы***

Предполагается, что программа уже находится на жестком диске компьютера.

Работы с программой состоит из следующих основных этапов:

1. запуск программы;
2. запуск или остановка эмуляции.

Для запуска программы с жесткого диска необходимо выполнить следующие действия:

1. установить JRE (Java Runtime Environment) не ниже версии 1.7;
2. Запустить программу файлом «run.bat».

После запуска на экране монитора появится рабочее окно программы (Рисунок 4.1).

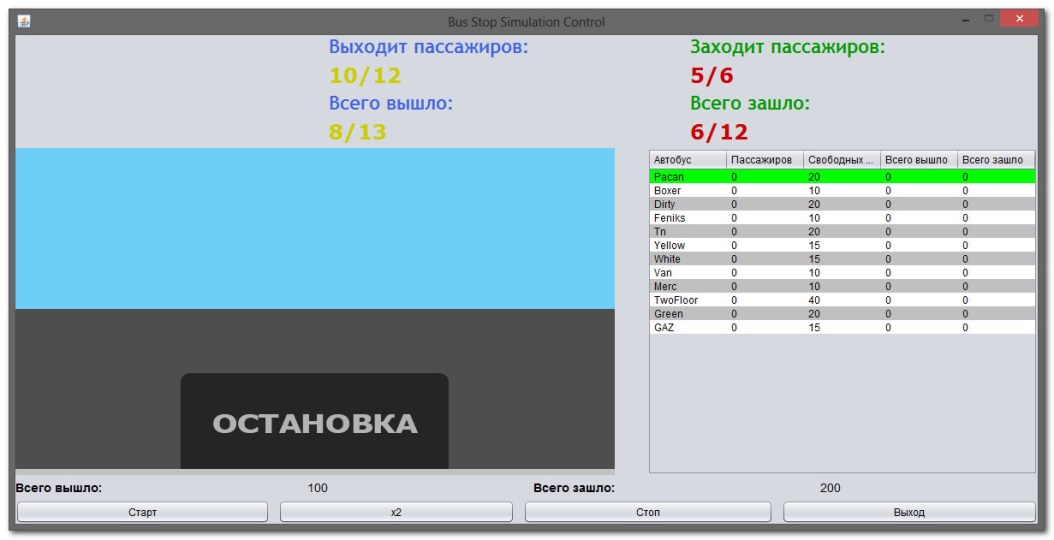


Рисунок 1. Главная форма

# Заключение

В соответствии с поставленной задачей в курсовой работе было выполнено следующее:

1. Изучена работа остановки городского транспорта;
2. разобран алгоритм моделирование работы остановки городского транспорта;
3. разработана программа на Java, реализующая данный алгоритм.