Московский государственный университет имени М. В. Ломоносова

факультет Вычислительной математики и кибернетики

**ОТЧЕТ ПО ПРАКТИКУМУ**

**Тема задания: «Табло на станции метро»**

*Выполнил:* студент 427 группы

Пантелеев Иван Михайлович

*Принял:* доц. кафедры СП, канд. физ.-мат. наук

[Малышко Виктор Васильевич](http://sp.cs.msu.ru/staff/mvv.html)

Москва 2010 г.

Оглавление

[Глава 1. Постановка задачи 3](#_Toc280709553)

[Глава 2. Анализ требований 4](#_Toc280709554)

[Глоссарий 4](#_Toc280709555)

[Описание действующих лиц 4](#_Toc280709556)

[Диаграмма вариантов использования 4](#_Toc280709557)

[Описание вариантов использования 5](#_Toc280709558)

[Вариант использования «Обработать прибытие/отправление поезда» (Process train) 5](#_Toc280709559)

[Вариант использования «Получить данные о нарушениях расписания» (Get schedule exceptions) 6](#_Toc280709560)

[Вариант использования «Продвинуть бегущую строку» (Creeping Line) 6](#_Toc280709561)

[Вариант использования «Загрузить рекламные сообщения» (Load advertising messages) 7](#_Toc280709562)

[Вариант использования «Получить импульс» (Recieve impulse) 7](#_Toc280709563)

[Вариант использования «Установка текущего времени» (Set current time) 8](#_Toc280709564)

[Вариант использования «Обновить текущее время» (Update current time) 8](#_Toc280709565)

[Вариант использования «Обновить расписание» (Update schedule) 9](#_Toc280709566)

[Вариант использования «Обновить таймер» (Update timer) 9](#_Toc280709567)

[Глава 3. Анализ системы 10](#_Toc280709568)

[Диаграмма ключевых абстракций 10](#_Toc280709569)

[Диаграммы последовательности для вариантов использования 11](#_Toc280709570)

[Вариант использования «Обработать прибытие/отправление поезда» (Process train) 11](#_Toc280709571)

[Вариант использования «Установка текущего времени» (Set current time) 13](#_Toc280709572)

[Вариант использования «Получить данные о нарушениях расписания» (Get schedule exceptions) 14](#_Toc280709573)

[Вариант использования «Обновить расписание» (Update schedule) 15](#_Toc280709574)

[Вариант использования «Загрузить рекламные сообщения» (Load advertising messages) 16](#_Toc280709575)

[Вариант использования «Получить импульс» (Recieve impulse) 17](#_Toc280709576)

[Вариант использования «Продвинуть бегущую строку» (Creeping Line) 18](#_Toc280709577)

[Вариант использования «Обновить текущее время» (Update current time) 19](#_Toc280709578)

[Вариант использования «Обновить таймер» (Update timer) 20](#_Toc280709579)

[Глава 4. Проектирование 21](#_Toc280709580)

[Иерархия проектных классов системы 21](#_Toc280709581)

[Диаграмма состояний, описывающая поведение класса TimeController 27](#_Toc280709582)

[Диаграмма классов подсистемы (Gen\_subsystem) 28](#_Toc280709583)

[Вид на подсистему изнутри 28](#_Toc280709584)

[Вид на подсистему снаружи 28](#_Toc280709585)

[Диаграммы последовательности, описывающие реализацию операций интерфейса ITimeController 29](#_Toc280709586)

[Диаграмма размещения 34](#_Toc280709587)

[Схема базы данных 34](#_Toc280709588)

[Заключение 35](#_Toc280709589)

# Глава 1. Постановка задачи

Табло на станции метро

Табло расположены на каждой станции метро. Они работают под управлением единого пункта управления (ПУ) информационной службы метро. Табло отображает текущее время (часы, минуты, секунды) и время, прошедшее с момента отправления последнего поезда (минуты, секунды). Момент прибытия и отправления поезда определяется при помощи датчиков, устанавливаемых на путях. Все табло метро синхронизованы, текущее время отсчитывается и устанавливается из центральной службы времени, находящейся на ПУ.

На табло высвечивается конечная станция назначения прибывающего поезда. Эти данные содержатся в расписании движения поездов, которое хранится в памяти табло и периодически обновляется с ПУ.

В "бегущей строке" табло отображается рекламная информация. Память табло хранит до 10 рекламных сообщений. Сообщения отображаются друг за другом с небольшими паузами, циклически. Содержание рекламных сообщений поступает с ПУ.

Дополнительная функция табло – по запросу с ПУ оно пересылает данные о нарушениях расписания (преждевременных отправлениях поездов или опозданиях).

В ходе выполнения задания должна быть создана схема базы данных для хранения рекламных сообщений, расписания и сведений о нарушении расписаний.

*Пояснение: в задании требуется разработать модель ПО только для табло, но не для пункта управления информационной службы.*

# Глава 2. Анализ требований

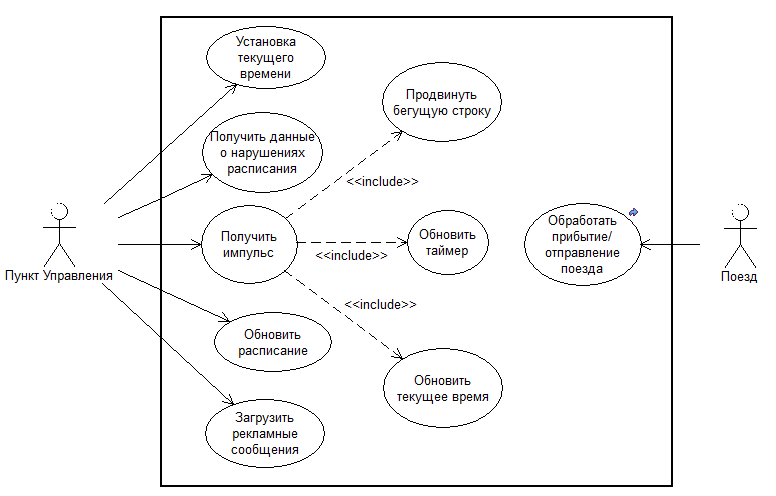
## Глоссарий

|  |  |
| --- | --- |
| Табло  (Indicator board) | Устройство, отображающее различную информацию |
| Поезд  (Train) | Сформированный состав, состоящий из нескольких сцепленных между собой вагонов. |
| Пульт Управления  (Сontrol panel) | Цент управления информационной службы метро |
| Бегущая строка  (Scrolling text) | Информация, выводящаяся на табло в виде прокручиваемого текста |
| Датчик  (Sensor) | Прибор, устанавливаемый на путях для определения времени прошедшего со времени отправления последнего поезда |
| Рекламное сообщение  (Advertising message) | Информация, направленная на привлечение внимания и отображаемая на бегущей строке табло. |

## Описание действующих лиц

|  |  |
| --- | --- |
| Пункт Управления | Специализированное средство, с помощью которого корректируется действие системы. |
| Поезд | Транспортное средство, которое приезжать и отправляться, подавая при этом сигнал на датчики, и тем самым влияя на поведение нашей системы. |

## Диаграмма вариантов использования



## **Описание вариантов использования**

### Вариант использования «Обработать прибытие/отправление поезда» (Process train)

Краткое описание

Данный вариант использования описывает обработку прибытия и отправления поезда, а так же отображение на табло конечной станции прибывшего поезда.

Основной поток событий

Данный вариант использования начинает выполняться, когда с датчиков, установленных на путях, поступает сигнал о прибытии поезда.

1. Система устанавливает в расписании поездов время прибытия поезда.
2. Система запрашивает у расписания конечную станцию для данного поезда.
3. Система отображает на табло конечную станцию прибывшего поезда.

Альтернативный поток событий

Данный вариант использования начинает выполняться, когда с датчиков, установленных на путях, поступает сигнал об отправлении поезда.

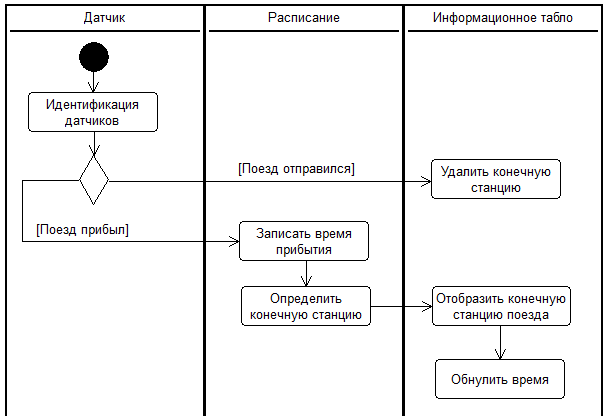
1. Система удаляет конечную станцию для отправившегося поезда с табло.
2. Обнуляется время, прошедшее с момента отправления поезда.
3. Начинается отсчёт времени прошедшего с момента отправления поезда.

Предусловия

Перед началом выполнения данного варианта использования система получает данные от датчиков, установленных на путях.

Постусловия

Если вариант использования выполнен успешно, на табло отразится конечная станция прибывшего поезда. В противном случае состояние системы не изменяется.

Диаграмма деятельности для варианта использования «Обработать прибытие/отправление поезда» (Process Train) 

### Вариант использования «Получить данные о нарушениях расписания» (Get schedule exceptions)

Краткое описание

Данный вариант использования описывает получение данных о нарушениях расписания (преждевременные отправления поездов или опоздания).

Основной поток событий

Данный вариант использования начинает выполняться, когда с Пункта Управления поступит запрос на получение данных о нарушениях расписания.

1. Пункт Управления запрашивает у системы, данные о нарушениях расписания.

2. Система передаёт данные о нарушениях расписания на Пункт Управления.

Предусловия

Перед началом выполнения данного варианта использования Пункт Управления запрашивает данные о нарушениях расписания.

Постусловия

Если вариант использования выполнен успешно, Пункт Управления получит необходимые данные о нарушениях расписания. В противном случае состояние системы не изменяется.

### Вариант использования «Продвинуть бегущую строку» (Creeping Line)

Краткое описание

Данный вариант использования описывает посимвольную прокрутку текущего рекламного сообщения и при обнаружении конца текущего рекламного сообщения, запрашивает следующее.

Основной поток событий

Данный вариант использования начинает выполняться, когда с Пункта Управления поступает очередной такт времени.

1. Система сдвигает текущее сообщение на один символ вправо.

2. При обнаружении конца текущего рекламного сообщения, система отображает следующий символ следующего рекламного сообщения.

3. При показе последнего рекламного сообщения, система заново начинает отображать первое рекламное сообщение.

Альтернативные потоки

Отсутствуют.

Предусловия

Перед началом выполнения данного варианта использования Пункт Управления посылает очередной такт времени.

Постусловия

Если вариант использования выполнен успешно, бегущая строка получит следующий символ рекламного сообщения, который необходимо отобразить и сдвинется на один символ вправо.

### Вариант использования «Загрузить рекламные сообщения» (Load advertising messages)

Краткое описание

Данный вариант использования описывает загрузку рекламных сообщений, которые впоследствии будут отображаться в виде бегущей строки.

Основной поток событий

Данный вариант использования начинает выполняться, когда с Пункта Управления поступит запрос на модификацию рекламных сообщений. Пункт Управления отправляет набор рекламных сообщений на бегущую строку.

Предусловия

Перед началом выполнения данного варианта использования система получает данные от Пункта Управления.

Постусловия

Если вариант использования выполнен успешно, данные будут сохранены или удалены из памяти электронного табло. В противном случае состояние системы не изменяется.

### Вариант использования «Получить импульс» (Recieve impulse)

Краткое описание

Данный вариант использования описывает получение импульса через равные промежутки времени от Пункта Управления.

Основной поток событий

Данный вариант использования начинает выполняться, когда с Пункта Управления поступает импульс.

1. Пункт Управления инициирует импульс.
2. Система получает импульс.

Предусловия

Перед началом выполнения данного варианта использования Пульт Управления отсылает импульс.

### Вариант использования «Установка текущего времени» (Set current time)

Краткое описание

Данный вариант использования описывает установку времени на основном информационном табло.

Основной поток событий

Данный вариант использования начинает выполняться, когда служба управления времени инициирует установку нового времени на основном информационном табло.

1. Пункт Управления передаёт системе новое время
2. Система сохраняет время
3. Система передаёт время на табло

Альтернативные потоки

Отсутствуют.

Предусловия

Перед началом выполнения данного варианта использования должны быть получены данные от центральной службы управления временем, расположенной на Пункте Управления

Постусловия

Если вариант использования выполнен успешно, новое время будет отражено на информационном табло. В противном случае состояние системы не изменяется.

### Вариант использования «Обновить текущее время» (Update current time)

Краткое описание

Данный вариант использования описывает обновление часов на информационном табло.

Основной поток событий

Данный вариант использования начинает выполняться, когда с Пункта Управления поступают новые данные о текущем времени.

1. Система сохраняет обновлённое время.
2. Система отображает новое время на информационном табло.

Предусловия

Перед началом выполнения данного варианта использования Пульт Управления генерирует новый импульс и предоставляет данные об изменившемся текущем времени.

Постусловия

Если вариант использования выполнен успешно, то на информационном табло произойдёт обновление текущего времени. В противном случае состояние системы не изменяется.

### Вариант использования «Обновить расписание» (Update schedule)

Краткое описание

Данный вариант использования описывает получение данных от Пункта Управления о новом расписании движения поездов.

Основной поток событий

Данный вариант использования начинает выполняться, когда с Пункта Управления поступает запрос на обновления текущего расписания движения поездов.

1. Пункт Управления инициирует запрос на обновление текущего расписания движения поездов.
2. Система удаляет старое расписание движения поездов.
3. Система загружает новое расписание движения поездов.

Предусловия

Перед началом выполнения данного варианта использования Пульт Управления запрашивает данные о расписании движения поездов.

Постусловия

Если вариант использования выполнен успешно, память электронного табло будет содержать обновлённое расписание. В противном случае состояние системы не изменяется.

### Вариант использования «Обновить таймер» (Update timer)

Краткое описание

Данный вариант использования описывает обновление таймера с момента отправления последнего поезда на информационном табло.

Основной поток событий

Данный вариант использования начинает выполняться, когда с Пункта Управления поступает новый импульс.

1. Система увеличивает на единицу время на таймере.
2. Система отображает новое время на информационном табло.

Предусловия

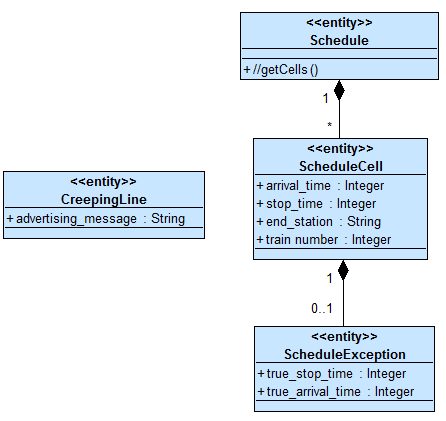
Перед началом выполнения данного варианта использования Пульт Управления генерирует новый импульс.

Постусловия

Если вариант использования выполнен успешно, то на информационном табло произойдёт обновление времени с момента отправления последнего поезда. В противном случае состояние системы не изменяется.

# Глава 3. Анализ системы

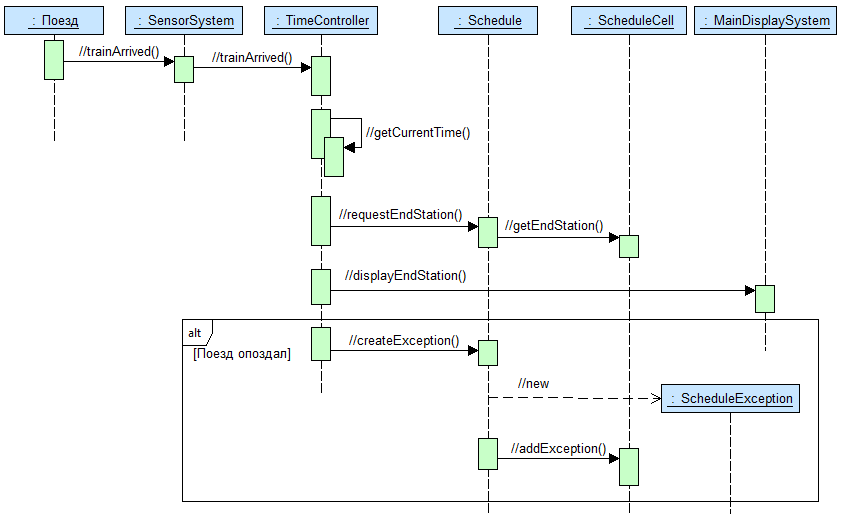
## Диаграмма ключевых абстракций



## Диаграммы последовательности для вариантов использования

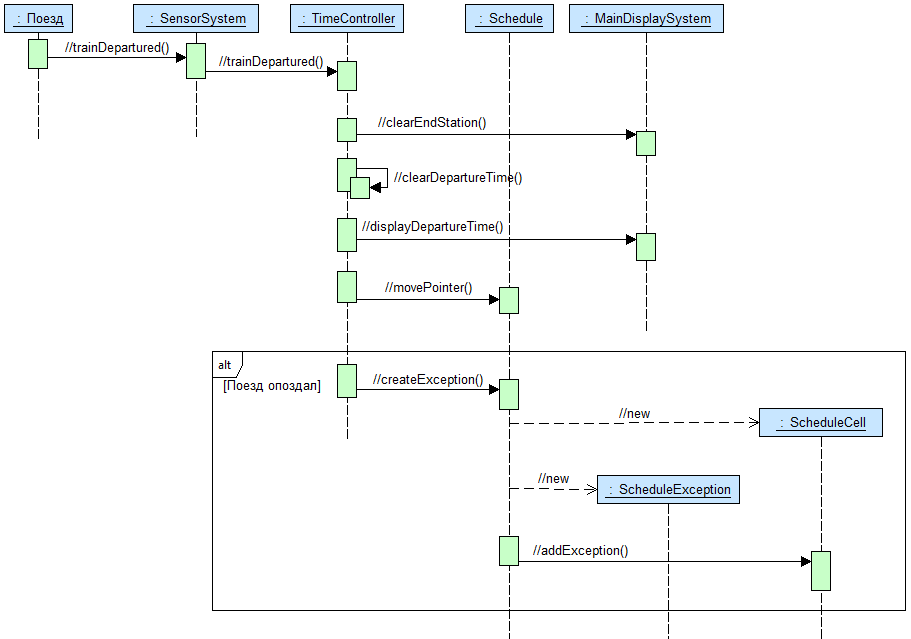
### Вариант использования «Обработать прибытие/отправление поезда» (Process train)

**Диаграмма последовательности (основной поток событий):**

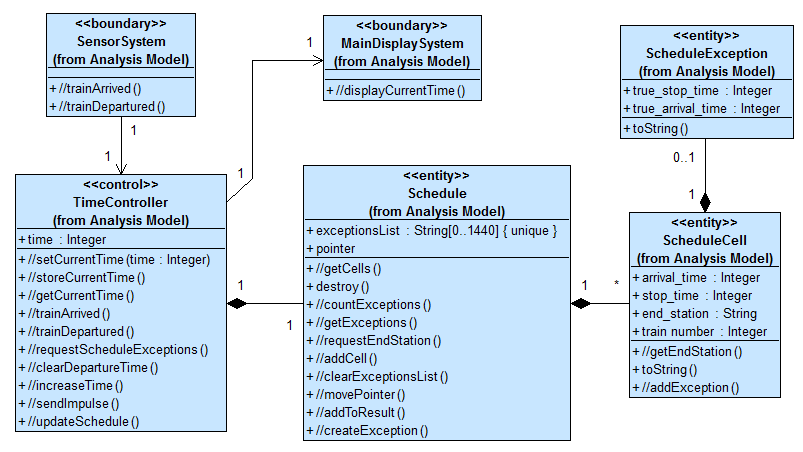


Поезд взаимодействует с экземпляром класса SensorSystem - сенсорами устанавливаемыми на путях, для детектирования факта и времени пребывания поезда. Если поезд опоздал, выполняется альтернативный блок и создаётся экземпляр класса ScheduleException.

**Диаграмма последовательности (альтернативный поток событий):**

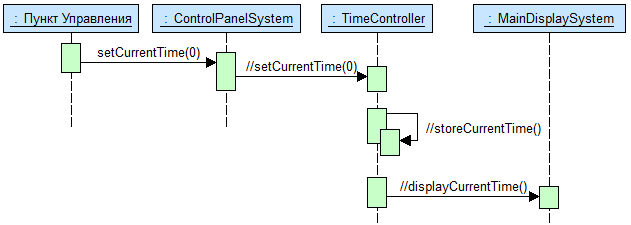


**Диаграмма взаимодействующих классов:**



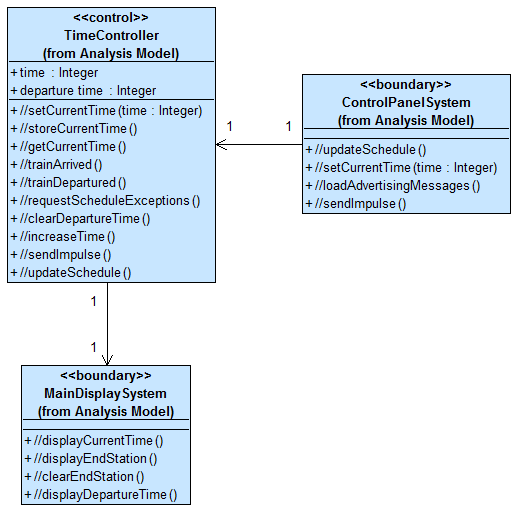
### Вариант использования «Установка текущего времени» (Set current time)

**Диаграмма последовательности:**



Действующее лицо «Пункт Управления» работает с экземпляром класса ControlPanelSystem для отправки нового текущего времени на дисплей.

**Диаграмма взаимодействующих классов:**



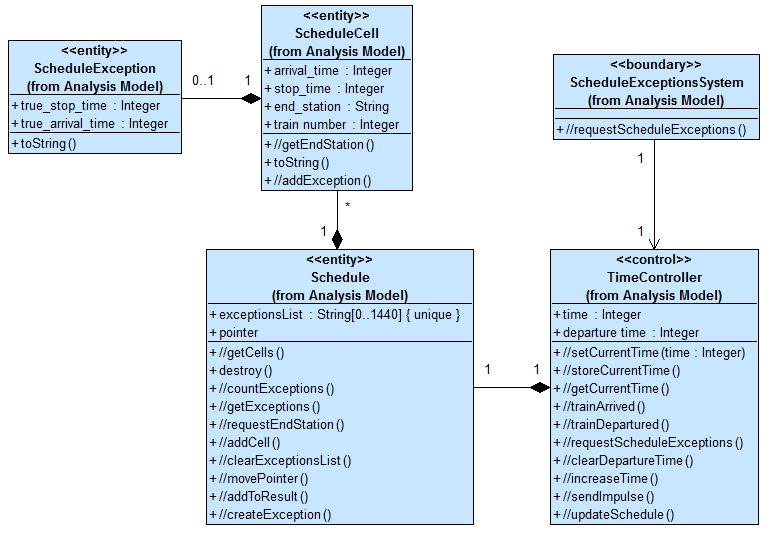
### Вариант использования «Получить данные о нарушениях расписания» (Get schedule exceptions)

**Диаграмма последовательности:**

### 

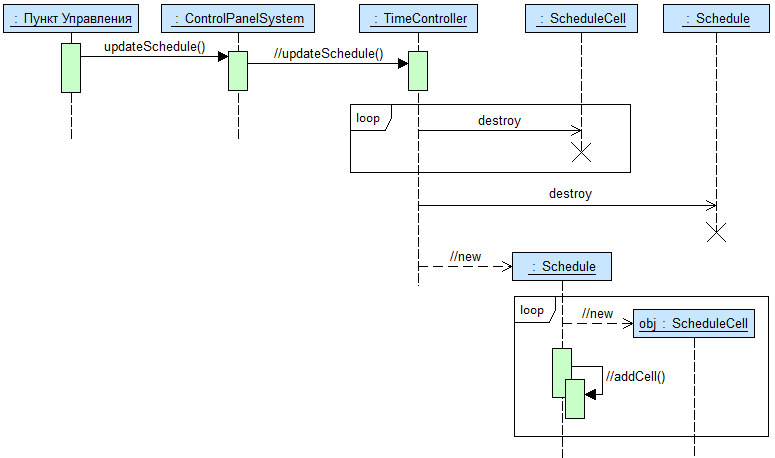
Действующее лицо «Пункт Управления» запрашивает набор всех ошибок в расписании движения поездов. В цикле в экземпляр класса Schedule добавляются все ошибки, предварительно удалив старые.

**Диаграмма взаимодействующих классов:**



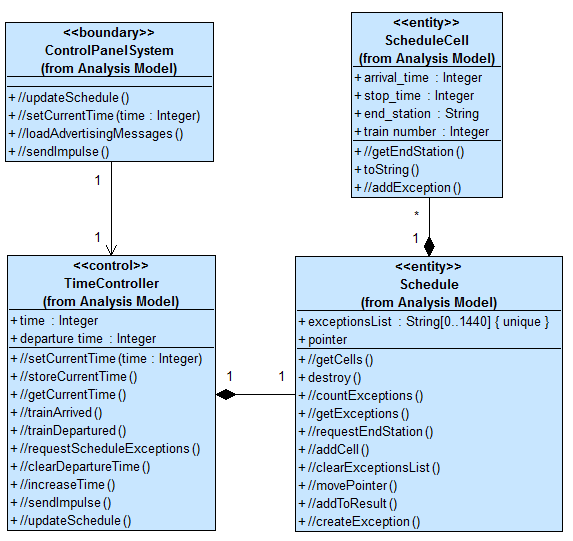
### Вариант использования «Обновить расписание» (Update schedule)

**Диаграмма последовательности:**



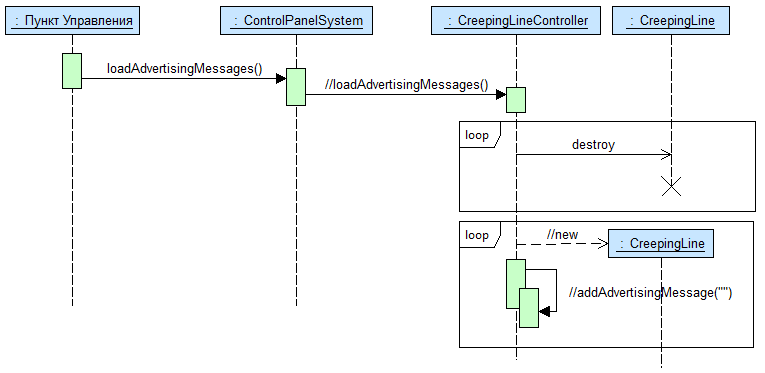
Для обновления расписания, действующее лицо «Пункт Управления» инициирует запрос updateSchedule. Удаляются все экземпляры классов ScheduleCell и Schedule. Затем создаются новые.

**Диаграмма взаимодействующих классов:**



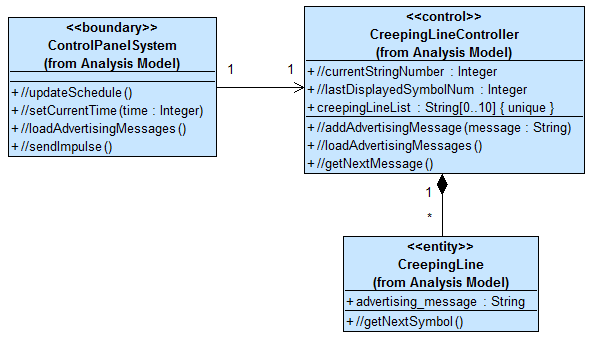
### Вариант использования «Загрузить рекламные сообщения» (Load advertising messages)

**Диаграмма последовательности:**



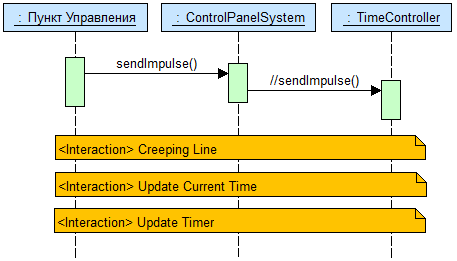
Действующее лицо «Пункт Управления» инициирует запрос на обновление всех рекламных сообщений. Перед обновлением удаляются все старые рекламные сообщения, после чего загружаются новые.

**Диаграмма взаимодействующих классов:**



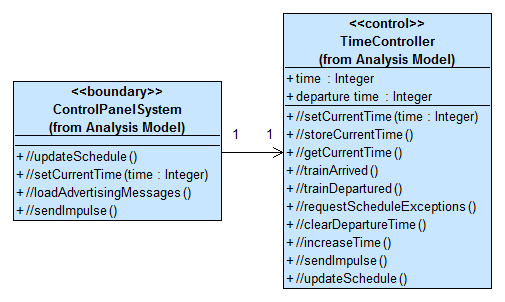
### Вариант использования «Получить импульс» (Recieve impulse)

**Диаграмма последовательности:**



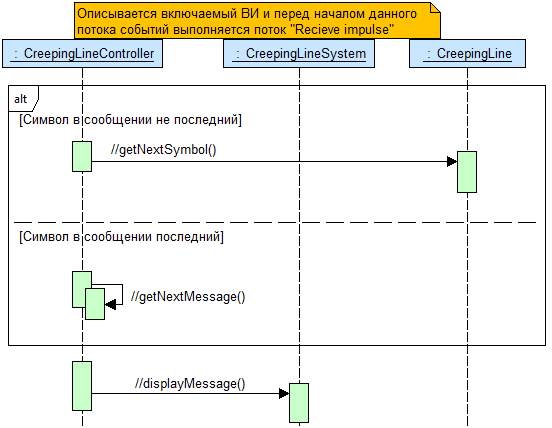
Действующее лицо «Пункт управления» инициирует отправку импульса экземпляру класса ControlPanelSystem. Далее выполняются 3 подчинённых потока.

**Диаграмма взаимодействующих классов:**



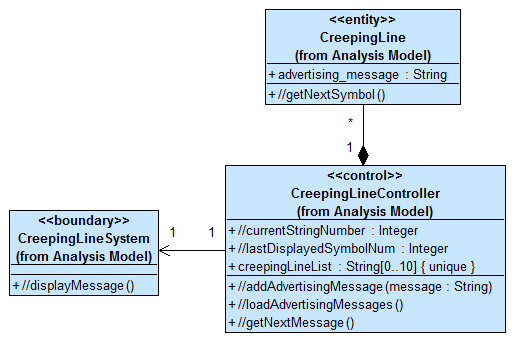
### Вариант использования «Продвинуть бегущую строку» (Creeping Line)

**Диаграмма последовательности:**



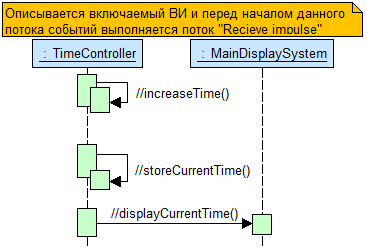
Экземпляр класса CreepingLineController запрашивает у экземпляра класса CreepingLine, где хранится текст рекламного сообщения, следующий символ для показа затем отображает его.

**Диаграмма взаимодействующих классов:**



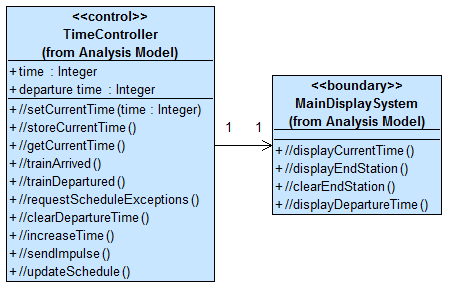
### Вариант использования «Обновить текущее время» (Update current time)

**Диаграмма последовательности:**



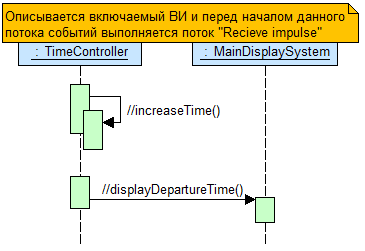
При получении импульса увеличивается текущее время и отображается на табло.

**Диаграмма взаимодействующих классов:**



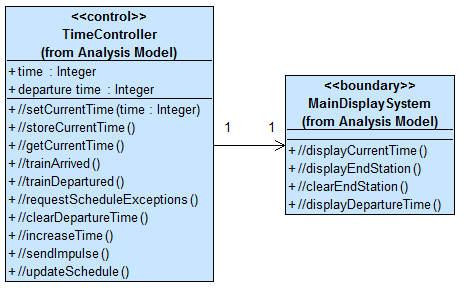
### Вариант использования «Обновить таймер» (Update timer)

**Диаграмма последовательности:**



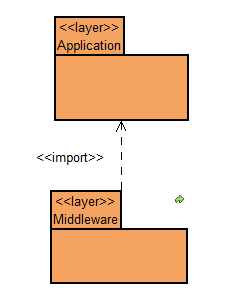
При получении импульса увеличивается время таймера, который отображает время, прошедшее с момента отправления последнего поезда.

**Диаграмма взаимодействующих классов:**



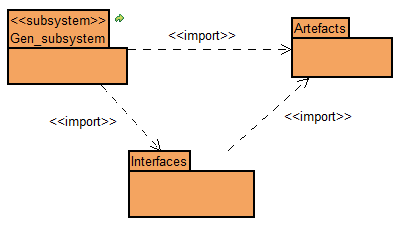
# Глава 4. Проектирование

## Иерархия проектных классов системы



Пакет **Middleware** содержит элементы, обеспечивающие сервисы, независящие от платформы.

Пакет **Application** содержит классы – элементы, относящиеся к предметной области, подсистему и интерфейс к ней.



Подсистема **Gen\_subsystem** отвечает за работу с бегущей строкой, расписанием, ошибками в расписании и временем.

Пакет **Artefacts** содержит единственный класс-сущность ScheduleExceptionsList содержащий в себе все ошибки, произошедшие в расписании движения поездов.

Пакет **Interfaces** содержит единственный интерфейс ITimeController

**Class SensorSystem**

*Экземпляр этого класса – система отвечающая за взаимодействие с датчиками, устанавливаемыми на путях.*

|  |  |  |
| --- | --- | --- |
| Имя фукнции | Возвращаемый тип | Описание |
| trainArrived () | void | Функция вызывается в момент срабатывания датчиков, установленных на путях, сигнализирующих о том, что поезд прибыл на станцию. |
| trainDepartured () | void | Функция вызывается в момент срабатывания датчиков, установленных на путях, сигнализирующих о том, что поезд отправился со станции. |

**Class ControlPanelSystem**

|  |  |  |
| --- | --- | --- |
| Имя фукнции | Возвращаемый тип | Описание |
| updateSchedule () | void | Функция вызывается в момент, когда с Пункта Управления приходит новое расписание движения поездов. |
| setCurrentTime (time:Integer) | void | Установка нового текущего времени, которое приходит с Пункта Управления |
| loadAdvertisingMessages (msg:CreepingLine) | void | Загрузка новых рекламных сообщений, которые приходят с Пункта Управления. |
| sendImpulse () | void | Функция вызывается в момент, когда с Пункта Управления приходит новый синхроимпульс. |

**Class ScheduleExceptionsSystem**

|  |  |  |
| --- | --- | --- |
| Имя фукнции | Возвращаемый тип | Описание |
| requestScheduleExceptions () | String | Запрос на получение всех нарушений в расписании движения поездов, инициируемым Пунктом Управления. |

**Class ScheduleExceptionsList**

*Экземпляр этого класса – список всех нарушений в расписании движения поездов*

|  |  |  |
| --- | --- | --- |
| Имя фукнции | Возвращаемый тип | Описание |
| add (obj:ScheduleException) | void | Добавить в список нарушений расписания очередное нарушение. |

**Class Schedule**

*Экземпляр этого класса – расписание движения поездов*

|  |  |  |
| --- | --- | --- |
| Имя Атрибута | Тип | Описание |
| exceptionsList | String | Список ошибок в расписании |
| pointer |  | Указатель на следующую строку в расписании |

|  |  |  |
| --- | --- | --- |
| Имя фукнции | Возвращаемый тип | Описание |
| getCells (cell:ScheduleCell) | void | Запрос на получение строк из расписания |
| countExceptions () | Integer | Подсчёт количества нарушений в расписании |
| getExceptions (exlst:ScheduleExceptionsList) | void | Запрос нарушений в расписании движения поездов. |
| requestEndStation () | String | Запрос конечной станции для данного поезда |
| addCell (cell:ScheduleCell) | void | Добавление новой строки в расписание |
| clearExceptionsList () | void | Очистка списка нарушений, произошедших в расписании движения поездов |
| movePointer () | void | Сдвиг указателя |
| addToResult () | void | Добавить нарушение |
| createException (ex:ScheduleException) | void | Создать новое нарушение в расписании |

**Class CreepingLine**

*Экземпляр этого класса – рекламное сообщение, предназначенное для отображения на бегущей строке*

|  |  |  |
| --- | --- | --- |
| Имя Атрибута | Тип | Описание |
| advertising\_message | String | Рекламное сообщение |

|  |  |  |
| --- | --- | --- |
| Имя фукнции | Возвращаемый тип | Описание |
| getNextSymbol () | char | Получить следующий символ текущего рекламного сообщения бегущей строки. |

**Class CreepingLineSystem**

*Экземпляр этого класса – дисплей, отображающий бегущую строку*

|  |  |  |
| --- | --- | --- |
| Имя фукнции | Возвращаемый тип | Описание |
| displayMessage (msg:CreepingLine) | void | Отобразить на дисплее бегущую строку, передаваемую в параметре msg. |

**Class ScheduleException**

*Экземпляр этого класса – ошибка в расписании движения поездов*

|  |  |  |
| --- | --- | --- |
| Имя Атрибута | Тип | Описание |
| true\_departure\_time | Integer | фактическое время отправления |
| true\_arrival\_time | Integer | фактическое время прибытия |

|  |  |  |
| --- | --- | --- |
| Имя фукнции | Возвращаемый тип | Описание |
| toString () | void | Преобразование в строку |

**Class ScheduleCell**

*Экземпляр этого класса – строка в расписании движения поездов*

|  |  |  |
| --- | --- | --- |
| Имя Атрибута | Тип | Описание |
| arrival\_time | Integer | Время прибытия поезда |
| stop\_time | Integer | Время стоянки поезда на станции |
| end\_station | String | Конечная станция поезда |
| train number | Integer | Номер поезда |

|  |  |  |
| --- | --- | --- |
| Имя фукнции | Возвращаемый тип | Описание |
| getEndStation () | String | Запрос конечной станции для данного поезда |
| toString () | void | Преобразование в строку |
| addException (ex:ScheduleException) | void | Создать новое нарушение в расписании |

**Class MainDisplaySystem**

*Экземпляр этого класса – дисплей, отображающий всю информацию, кроме бегущей строки*

|  |  |  |
| --- | --- | --- |
| Имя фукнции | Возвращаемый тип | Описание |
| displayCurrentTime (time:Integer) | void | Отображение на дисплее текущего времени |
| displayDepartureTime (time:Integer) | void | Отображение времени прошедшего, с момента отправления поезда |
| displayEndStation (station:String) | void | Отображение конечной станции поезда |
| clearEndStation () | void | Удаление конечной станции поезда |

**Class TimeController**

|  |  |  |
| --- | --- | --- |
| Имя Атрибута | Тип | Описание |
| currentStringNumber | Integer | Номер текущего рекламного сообщения |
| lastDisplayedSymbolNum | Integer | Номер последнего отображённого символа в текущем рекламном сообщении |
| creepingLineList | String | Список рекламных сообщений |
| time | Integer | Текущее время |
| departure time | Integer | Время, прошедшее с момента отправления последнего поезда |

**Class ITimeController**

*Данный интерфейс отвечает за контакт с подсистемой Gen\_subsystem.*

|  |  |  |
| --- | --- | --- |
| Имя фукнции | Возвращаемый тип | Описание |
| addAdvertisingMessage (message:String) | void | Добавление нового рекламного сообщения |
| loadAdvertisingMessages (msg:CreepingLine) | void | Загрузка нового рекламного сообщения |
| getNextMessage () | String | Получение следующего рекламного сообщения |
| setCurrentTime (time:Integer) | void | Установка текущего времени |
| storeCurrentTime () | void | Сохранение текущего времени |
| getCurrentTime () | Integer | Получение текущего времени |
| trainArrived () | Boolean | Возвращает true, если поезд прибыл, в противном случае false |
| trainDepartured () | Boolean | Возвращает true, если поезд отправился, в противном случае false |
| requestScheduleExceptions () | ScheduleExceptionsList | Запрос нарушений в расписании |
| clearDepartureTime () | void | Очистка времени, прошедшего с момента отправления последнего поезда |
| increaseTime () | void | Увеличение времени |
| sendImpulse () | void | Функция, сигнализирующая о том, что с Пульта Управления был послан импульс |
| updateSchedule (schedule:Schedule) | void | Обновление текущего расписания движения поездов |

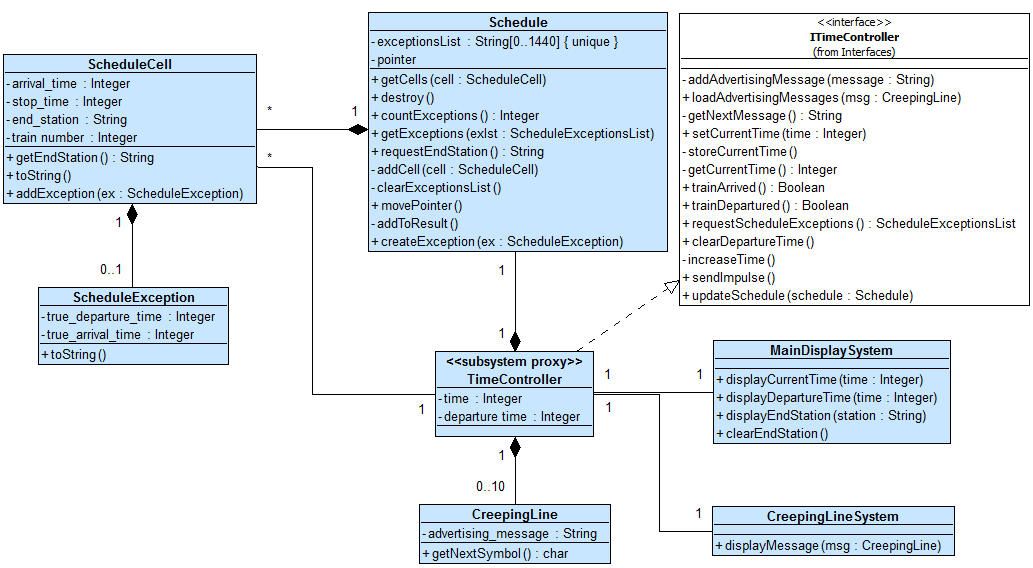
## 

## Диаграмма состояний, описывающая поведение класса TimeController

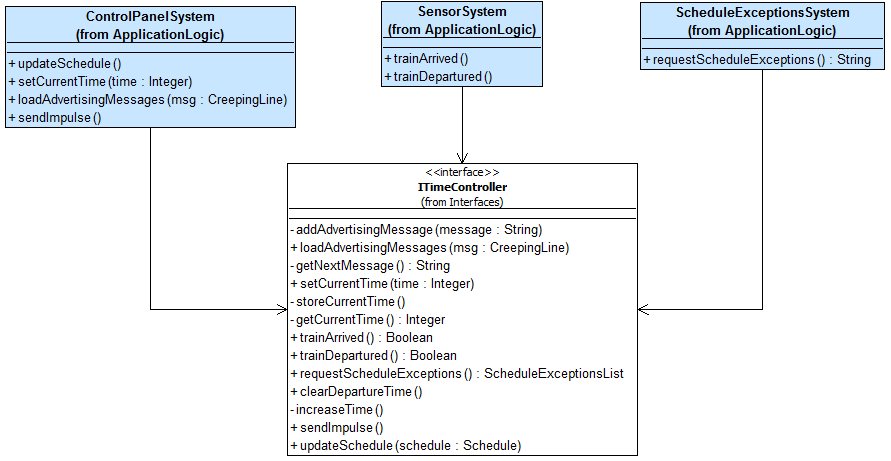
## 

## Диаграмма классов подсистемы (Gen\_subsystem)

### Вид на подсистему изнутри

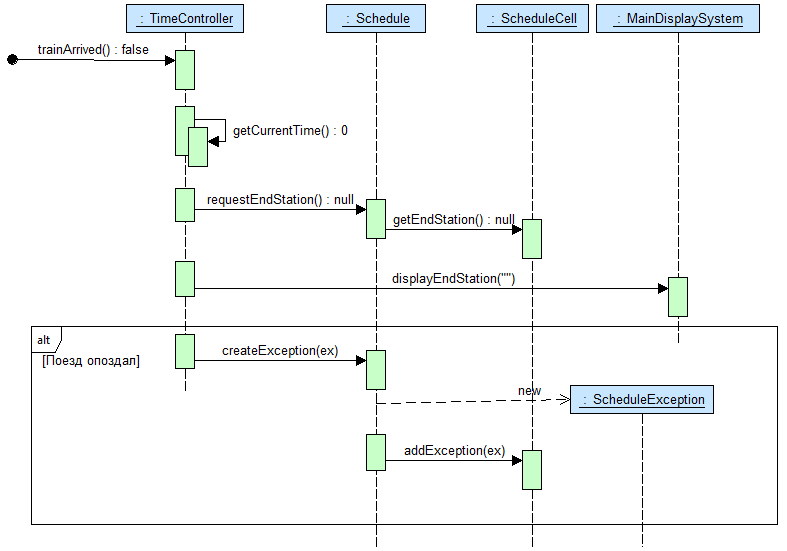


### Вид на подсистему снаружи

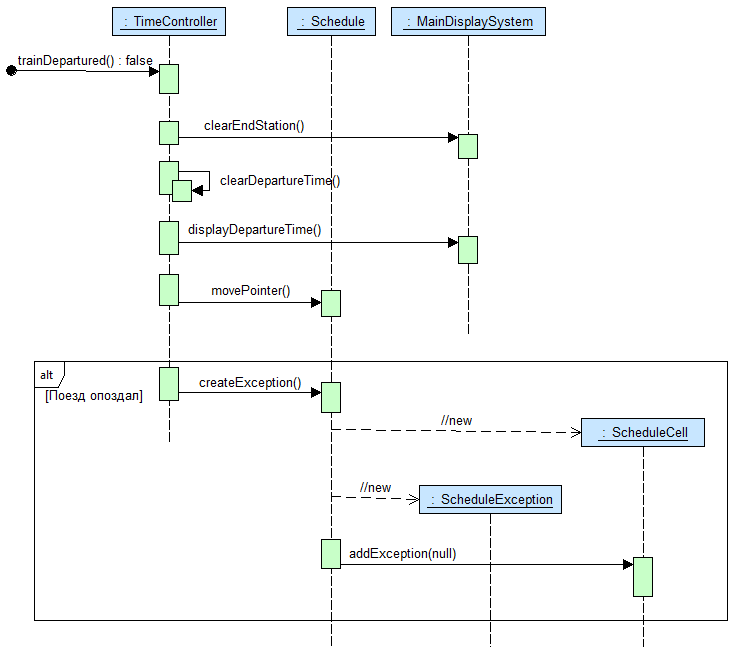


### Диаграммы последовательности, описывающие реализацию операций интерфейса ITimeController

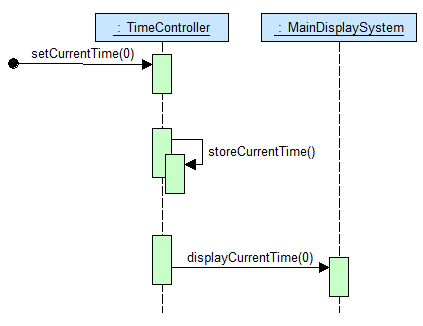
#### Диаграмма последовательности для операции trainArrived



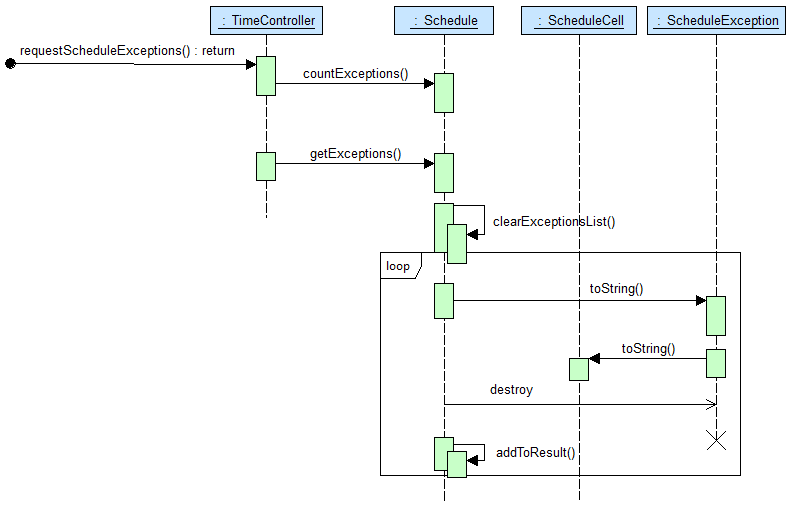
#### Диаграмма последовательности для операции trainDepartured



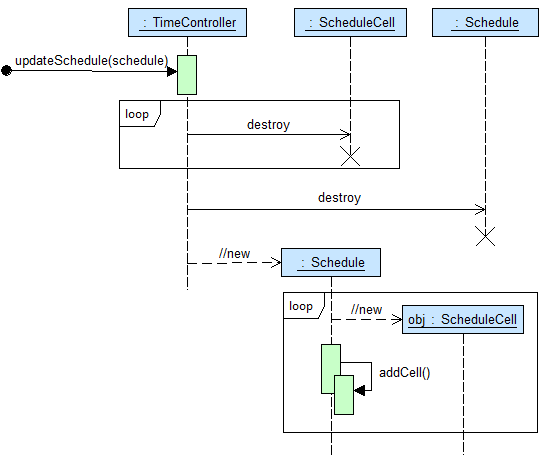
#### Диаграмма последовательности для операции setCurrentTime



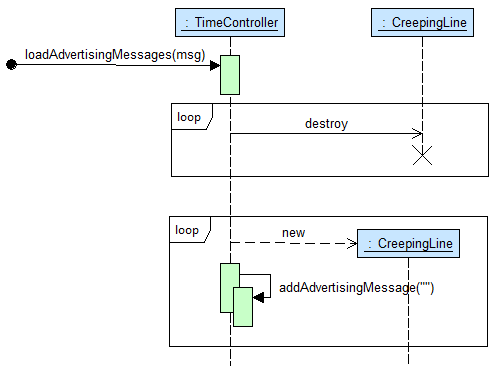
#### Диаграмма последовательности для операции requestScheduleExceptions



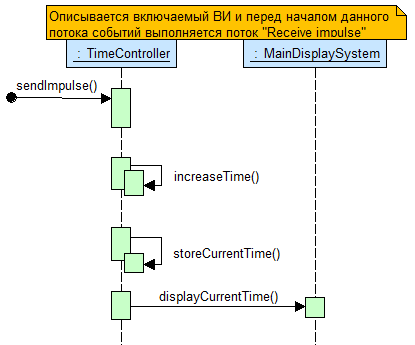
#### Диаграмма последовательности для операции updateSchedule



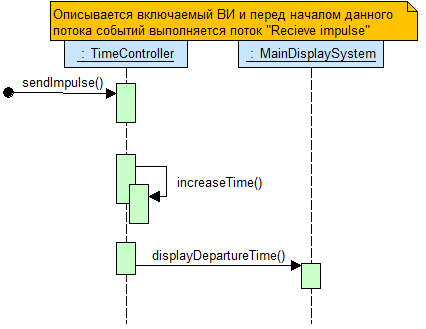
#### Диаграмма последовательности для операции loadAdvertisingMessages



#### Диаграмма последовательности для операции sendImpulse



#### Диаграмма последовательности для операции sendImpulse



#### Диаграмма последовательности для операции sendImpulse

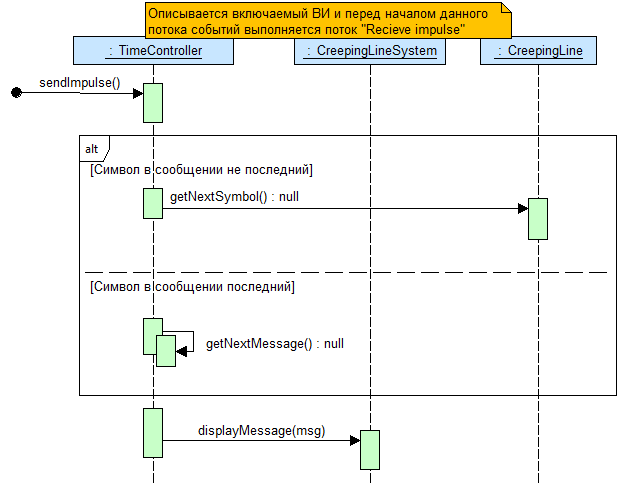
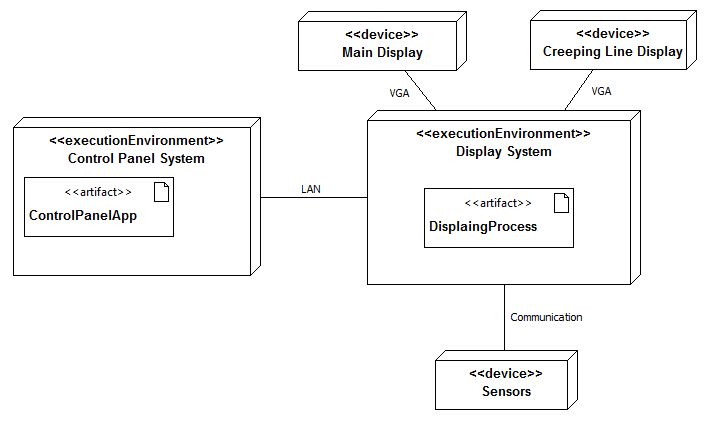
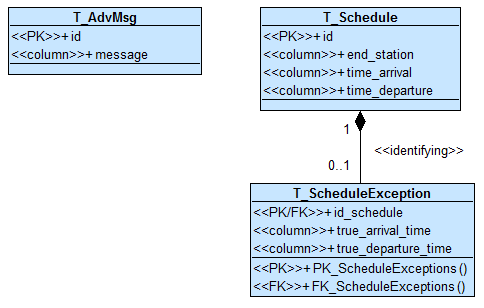


Диаграмма размещения



## 

## Схема базы данных



# Заключение

При проектировании системы «Табло в метро» было выделено:

* 9 вариантов использования
* 11 классов (1 основных классов, 3 граничных классов, 7 классов подсистемы)

Было построено:

* 1 диаграмма вариантов использования
* 1 диаграмма состояний
* 1 диаграмма деятельности
* 1 диаграмма размещения
* 14 диаграмм последовательности (6 диаграмм для различных вариантов использования, 8 диаграммы для реализаций операций интерфейса подсистемы)
* 13 диаграмм классов (2 диаграммы связей пакетов системы, 1 диаграмма ключевых абстракций, 6 диаграмм взаимодействующих классов - по одной для каждого варианта использования, 2 диаграммы для описания взаимодействий классов на разных уровнях системы, 2 диаграмма классов подсистемы)

Итого было построено **31** различные диаграммы.

Система «Табло в метро» полностью спроектирована, включая подсистему.