Белгородский государственный технологический университет им. В. Г. Шухова

Кафедра ПОВТиАС

Расчетно-графическое задание на тему   
«Моделирование информационного процесса средствами объектно-ориентированного программирования»

Выполнил:  
студент группы ПВ-21  
Адаменко Игорь

Проверил:  
старший преподаватель  
Гарибов А. И.

Белгород  
2014

# Задание к работе

1. Для заданного в соответствии с вариантом описания информационного процесса выполнить объектно-ориентировочный анализ и выделить основные типы объектов предметной области.
2. Реализовать выделенные типы объектов в виде классов, выбрав необходимые свойства, методы и события.
3. Разработать структуру приложения Windows Forms для моделирования предметной области. Для каждой формы приложения определить выполняемые функции и разработать алгоритм их реализации, а также принципы и стратегии взаимодействия объектов.
4. Выполнить реализацию разработанных функций и алгоритмов в виде приложения Windows Forms. При спроектировании интерфейса использовать элементы управления DevExpress или ComponentOne.
5. Сравнить функциональность выбранных элементов, сделать выводы о возможности замены выбранных компонентов DevExpress или ComponentOne стандартными (выходящими в состав .NET Framework).

## Задание для первого варианта

На каждой станции метро расположены информационные табло, которые работают под управлением единой информационной службы метро. На табло отображается текущее время (часы, минуты, секунды) и время отправления последнего поезда. Момент прибытия и отправления поезда определяется при помощи датчиков, устанавливаемых на путях. Все табло в метро синхронизированы, текущее время отсчитывается и устанавливается из единой службы времени, являющейся частью единой информационной службы метро.

На табло высвечивается конечная станция назначения прибывающего поезда. Эти данные содержатся в расписании движения поездов, которое хранится в памяти табло и периодически обновляется единой информационной службой метро.

В «бегущей строке» табло отображается рекламная информация. Память табло хранит до 10 рекламных сообщений. Сообщения отображаются друг за другом с небольшими паузами. Содержание рекламных сообщений поступает из единой информационной службы метро.

Дополнительная функция табло — по запросу единой информационной службы оно пересылает данные о нарушениях расписания (преждевременных отправлениях поездов или опозданиях).

# Хранение данных

Вся информация, касающаяся поездов, хранится в файле ресурсов *DataTrains.resx*, в котором имеет вид <Ключ, Значение>, где <Ключ> — это конечный пункт в маршруте поезда, а <Значение> — это строка вида <A#B#C>, где A — время прибытия поезда, B — время его остановки, C — время отправления.

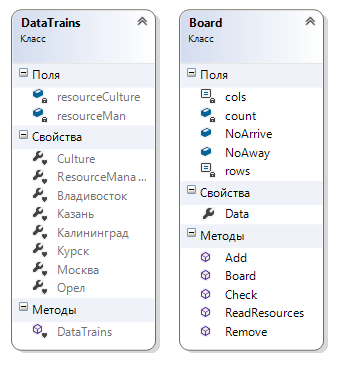
Информация о рекламе хранится в файле ресурсов *Ads.resx,* в котором имеет вид <Ключ, Значение>, где <Ключ> — любая уникальная последовательность символов, а <Значение> — рекламное сообщение.

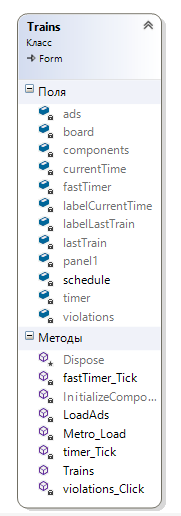
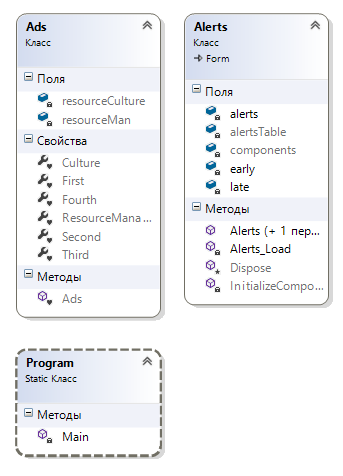
В ожидании обновления расписания и рекламной информации файлы перезагружаются каждую минуту.

# Используемые компоненты

1. Для реализации отображения времени используются компоненты Label в совокупности с Timer (*Tick* которого установлен в 1000).
2. Для отображения информационного табло с расписанием поездов и информацией об их задержке используется компонент FlexGrid из набора ComponentOne, у которого скрыт первый столбец и неактивны *ScrollBar*, а так же отключены все возможности его редактирования (фактически получился статический, но обновляющийся элемент).   
     
   Фактически можно было бы реализовать все это и без использования FlexGrid, поскольку никакого редактирования информации не требуется, а значит, можно было бы использовать для реализации интерфейса элементы Label, в совокупности с Panel.
3. Для реализации кнопки отображения информации о задержке поездов используется элемент Button.
4. Для реализации «бегущей строки» используется элемент Label, который заполняется длинной строкой, которая впоследствии (благодаря еще одному элементу Timer с *Tick* равным 100) укорачивается на один символ слева из-за переноса его в правую часть данной строки.

# Схемы классов





# Классы, методы и свойства

## Ads, DataTrains

Классы ресурса со строгой типизацией для поиска локализованных строк и т. п. Созданы автоматически стандартным классом StronglyTypedResourceBuilder.

## Board

Основной класс, используемый в программе. В нем реализованы основные методы и свойства для реализации табло метро.

### Поля

* NoArrive — публичное поле, представленное словарем *<string, int>;* используется для хранения информации о поездах, которые не прибыли вовремя.
* NoAway — публичное поле, представленное словарем *<string, int>;* используется для хранения информации о поездах, которые не убыли вовремя.
* Count — приватное поле типа *int*, которое содержит число пунктов расписания.
* Rows — приватное константное поле типа *int*, содержащее количество строк табло.
* Cols — приватное константное поле типа *int*, содержащее количество столбцов табло.

### Свойства

* Data — публичное свойство, реализующее доступ к матрице типа *string*, использующейся для хранения информации о поездах.

### Методы

* Add — публичный метод, используемый для добавления новой строки Data; принимает параметр row типа *string[]* — добавляемая строка.
* Remove — публичный метод, удаляющий необходимую строку в Data; принимает параметр number типа *int* — номер удаляемой строки.
* ReadResources — публичный метод, считывающий данные о расписании из файла ресурсов DataTrains.resx.
* Check — публичный метод, сравнивающий время отправления/прибытия с заявленным и регистрирующий задержки/опоздания; принимает параметр time типа *string* — время.

## Trains

Класс, реализующий визуальную составляющую программы. Регистрирует события и вызывает необходимые функции при их возникновении.

### Поля

* ads — поле типа *Label*, содержащее рекламную информацию;
* board — поле типа *C1FlexGrid* (элемент ComponentOne), содержащее основную информацию о расписании;
* currentTime — поле типа *Label*, содержащее информацию о текущем времени;
* fastTimer — поле типа *Timer*, содержащее информацию о скорости движения рекламной строки;
* labelCurrentTime — поле типа *Label*, содержащее пояснительную информацию для currentTime;
* lastTrain — поле типа *Label*, содержащее информацию о времени отправления последнего поезда;
* labelLastTrain — поле типа *Label*, содержащее пояснительную информацию для lastTrain;
* timer — поле типа *Timer*, содержащее информацию о скорости изменения времени (currentTime);
* violations — поле типа *Button*, содержащее информацию о кнопке на табло, отвечающей за появление информации о нарушении расписания.

### Методы

* fastTimer\_Tick — метод, изменяющий положение рекламной строки при возникновении события *Tick* объекта fastTimer;
* LoadAds — метод, загружающий информацию об объявлениях для рекламной строки из файла ресурсов Ads.resx;
* Metro\_Load — метод, загружающий информацию о поездах при возникновении события *Load* главного окна программы;
* timer\_Tick — метод, изменяющий отображаемое текущее время при возникновении события *Tick* объекта timer;
* violations\_Click — метод, открывающий окно с информацией о нарушении расписания, при возникновении события *Click* объекта violations.

## Alerts

Класс, отвечающий за хранение и отображение информации о нарушениях расписания.

### Поля

* alerts — приватное поле-матрица типа *string*, используемое для хранения информации о нарушениях расписания;
* late — приватное поле-словарь *<string, int>,* хранящее время опоздания поездов;
* early — приватное поле-словарь *<string, int>,* хранящее разницу во времени между ожидаемым и фактическим прибытием поездов.

### Методы

* Alerts — публичный метод-перегрузка стандартного конструктора, принимающий параметры-словари first и second вида *<string, int>* и сохраняющий их в late и early соответственно;
* Alerts\_Load — приватный метод, вызываемый при возникновении события *Load* окна Alerts; добавляет информацию о нарушениях расписания в окно Alerts.