**Занятие #22. Использование стандартных классов в проекте «Расписание поездов»**

Соблюдайте [требования к оформлению работы](http://edu.mmcs.sfedu.ru/mod/assign/view.php?id=9215), указанные в предыдущих лабораторных.

**Комментарии к теме**

**Скачайте** [**последнюю версию PascalABC.NET**](http://pascalabc.net/downloads/PascalABCNETMiniSetup.exe)**!**

**Об использовании исключений**

Раньше для проверки некорректных входных данных мы использовали Assert. На самом деле обычно Assert используется только при разработке программного продукта — для тестирования. В так называемых release-версиях продуктов Assert’ы отключаются.

Допустимые значения полей объектов классов являются неотъемлемой *частью абстракции* класса и определяются предметной областью. *Нельзя допускать создание* некорректных объектов. Попытка создать такой объект (например, человека с отрицательным возрастом) считается недопустимой **исключительной** ситуацией. Поэтому в случае некорректности входных данных принято выбрасывать **исключение**.

На этом занятии мы начнём использовать **исключения** для проверки корректности входных данных. В частности, речь идёт о параметрах конструкторов, сеттеров и других методов класса.

**Замечание.** Обратите внимание, что для обработки некорректных входных данных в пространстве имён System уже определены некоторые подходящие типы исключений:

* ArgumentOutOfRangeException: если значение находится вне допустимого диапазона.
* ArgumentNullException: если вместо объекта в качестве аргумента передано nil.

**Классы коллекций и другие стандратные классы**

На этом занятии мы будем использовать следующие *классы коллекций*:

* [List<T>](https://msdn.microsoft.com/ru-ru/library/6sh2ey19%28v=vs.110%29.aspx) — динамический массив объектов (имеется доступ по индексу, вставка за O(n));
* [LinkedList<T>](https://msdn.microsoft.com/ru-ru/library/vstudio/he2s3bh7%28v=vs.100%29.aspx) — двусвязный линейный список (похож на тот, который мы реализовали ранее сами; вместо head/tail используется First/Last);
* [Dictionary<TKey, TValue>](https://msdn.microsoft.com/ru-ru/library/xfhwa508%28v=vs.110%29.aspx) — ассоциативный массив.

Кроме того будут использоваться стандартные классы для работы с *датой и временем* [DateTime](https://msdn.microsoft.com/ru-ru/library/system.datetime%28v=vs.110%29.aspx) и [TimeSpan](https://msdn.microsoft.com/ru-ru/library/system.timespan%28v=vs.110%29.aspx) (примеры работы можно найти [здесь](http://edu.mmcs.sfedu.ru/pluginfile.php/15950/mod_folder/content/0/datetime.pas?forcedownload=1)), а также класс для эффективного формирования строки [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder%28v=vs.110%29.aspx).

При работе с коллекциями пригодятся следующие *методы* (они работают для всех стандартных классов коллекций):

* [Select<TSource, TResult>](https://msdn.microsoft.com/ru-ru/library/bb548891%28v=vs.110%29.aspx) принимает функцию одного аргумента типа TSource, возвращающую значение типа TResult. Select применяет эту функцию ко всем элементам коллеции и возвращает новую коллекцию, содержащую результаты обработки элементов.

Пример.

var reals1 := new List<real>();

reals1.Add(1.5);

reals1.Add(-2.0);

reals1.Add(0.35);

var reals2 := new List<real>(

reals1.Select(

x -> x + 1 // лямбда-функция

)

);

// reals2: 2.5, -1.0, 1.35

Тонкости работы с коллекциями станут ясны чуть позже из лекций, а пока следует обратить внимание на следующее: метод Select<,>, как и многие другие, возвращает некую абстрактную коллекцию. Её можно использовать в качестве параметра конструктора для *любой конкретной коллекции* (в примере выше это такой же тип List<real>). По списку вещественных чисел можно получить и двусвязный список целых чисел — округлённых значений соответствующих элементов:

var ints1 := new LinkedList<integer>(

reals1.Select(round)

);

// ints1: 2, -2, 0

* [Where<TSource>](https://msdn.microsoft.com/ru-ru/library/bb534803%28v=vs.110%29.aspx) принимает функцию-предикат и возвращает новую коллекцию, содержащую только элементы исходной коллекции, удовлетворяющие этому предикату.
* [All<TSource>](https://msdn.microsoft.com/ru-ru/library/bb548541%28v=vs.110%29.aspx)/[Any<TSource>](https://msdn.microsoft.com/ru-ru/library/bb534972%28v=vs.110%29.aspx" \o ") принимает функцию-предикат и возвращает истину, если все (All)/хотя бы один (Any) элементы удовлетворяют предикату, и ложь в противном случае.

**Лабораторная работа**

Оценка: 4 «грязных» балла.

**Требования к выполнению лабораторной**

* Решения задач оформляются в виде подпрограмм.
* Подпрограммы должны сопровождаться документирующими комментариями (с кратким описанием назначения подпрограммы).
* Подпрограммы должны содержать проверку входных данных с помощью *исключений*.
* Подпрограммы и типы данных следует определять в модуле TrainsManagement.pas, демонстрацию проводить в основной программе TrainsDemo.pas.
* Давайте *понятные имена* классам, полям, свойствам и методам. Они должны отражать своё назначение.

**Задание. Проект «Расписание поездов»**

Необходимо разработать систему генерации Расписания поездов. Для разработки этой системы:

* Сначала внимательно прочитайте описание задания, затем — подсказки к выполнению.
* Работу с системой необходимо продемонстрировать в основной программе.

***Терминология.***

* Поезд — train;
* остановка — stop;
* станция — station;
* маршрут — route;
* длительность — duration;
* расстояние — distance;
* скорость — speed;
* место отправления — source;
* место назначения — destination;
* прибытие — arrival;
* отправление — departure;
* расписание (поездов) — timetable.

**Описание**

В распоряжении железнодорожной компании «Гравитация — ж/д с комфортом» есть набор комфортабельных поездов. Каждый поезд характеризуется:

* серийным номером (10 цифр в диапазоне 1..9);
* скоростью движения, на которой поезд идёт максимально тихо.

Ж/д компания обслуживает несколько направлений пассажироперевозок. Например:

* Маршрут «М320-А» обеспечивает движение по маршруту Ростов — Москва со следующими остановками: Ростов, Новочеркасск, Каменская, Миллерово, Россошь, Воронеж, Грязи, Мичуринск, Рязань, Москва.
* Маршрут «М321-Б» обеспечивает движение по маршруту Москва — Ростов со следующими остановками: Москва, Рязань, Воронеж, Россошь, Ростов.

В зависимости от сезона, популярности того или иного напрвления и других факторов, логисты компании составляют расписание маршрутов. Они назначают день и время посадки в пункте отправки; выбирают время стоянки на каждой остановке; среди свободных поездов выбирают, какой поезд пустить по данному маршруту.

**Цель.** По информации о расписании маршрутов необходимо составить расписание поездов для каждой станции, на которой останавливаются поезда данной компании.

На каждой станции должна быть информация о списке останавливающихся поездов. Эту информацию необходимо уметь распечатывать так, чтобы информация о поездах следовала в порядке времени их прибытия на станцию, при этом для каждого поезда должно быть написано:

* номер маршрута;
* пункт назначения;
* время прибытия;
* время отправления.

Кроме того, необходимо обеспечить на станции возможность поиска поезда, который проезжает через заданную станцию и отправляется как можно раньше относительно заданного времени.

**Подсказки к реализации**

[Заготовки](http://edu.mmcs.sfedu.ru/pluginfile.php/15950/mod_folder/content/0/trains-timetable.zip?forcedownload=1) уже содержат часть реализации системы. Основные идеи заключаются в следующем:

1. В предметной области содержится ряд сущностей: *поезд* (железнодорожный состав), *маршрут* (пункты отправления и назначения, список остановок с информацией о расстоянии между станциями); *активный маршрут* (или маршрут на линии — данные о маршруте, прикреплённый состав, выполняющий рейс, время посадки на поезд, время стоянки на остановках).
2. Для поезда определён класс Train. Он содержит информацию о серийном номере и наиболее приятной скорости движения.

**{Оценка: 0.75} Задание.** Просмотрите интерфейс и реализацию класса поезда.

* + Исправьте метод проверки серийного номера CheckSerialNumber так, чтобы он удовлетворял спецификации задания.
  + Обратите внимание на метод ToString  
    function ToString(): string; override;  
    Строковый результат этой функции используется, например, при выводе объекта через Write или WriteFormat. Почему этот метод следует описывать именно так (с ключевым словом override) станет ясно позже из лекций.
  + Запустите основную программу. Убедитесь, что поезда загрузились, и на экран первой выведена строка

[Train(1556529987, speed: 300),Train(1556529456, speed: 312),Train(7556541211, speed: 288),Train(7556541212, speed: 288),Train(7556641237, speed: 295)]

* + Посмотрите, как реализована функция LoadTrainsFromTextFile загрузки списка поездов из файла (загляните в файл trains-dats.txt). Всё ли понятно?

1. Для маршрута определён класс Route с информацией об идентификаторе маршрута, пунктах отправления и назначения. Основная сложность в задании маршрута заключается в описании списка остановок: нам необходимо как-то задать список остановок и расстояние между ними. Если открыть какой-нибудь сайт с информацией о маршруте реального поезда (например, [этот](http://www.tutu.ru/poezda/view_d.php?np=acef5e3c)), можно заметить, что возле каждой остановки указано расстояние от начала пути. Можно выбрать аналогичный подход.

В модуле определён класс Stop — остановка на маршруте. Остановка содержит информацию о названии остановки и расстоянии от начала пути в километрах.  
В классе Route информация об остановках содержится в форме списка List<Stop>. Список должен содержать полный список остановок в порядке их следования, начиная со станции отправления и заканчивая станцией прибытия.

**{Оценка: 1.25} Задание.** Посмотрите на реализацию метода Route.ToString: он демонстрирует использование класса [StringBuilder](https://msdn.microsoft.com/ru-ru/library/system.text.stringbuilder%28v=vs.110%29.aspx). Посмотрите на результат его работы в основной программе (оператор writeln(routes[0].RouteInfo);). Исправьте метод так, чтобы остановки разделялись не запятыми, а тире.

1. Дла активного маршрута (маршрута на линии) определён класс ActiveRoute. Он содержит информацию о дате-времени посадки на поезд, прикреплённом поезде, а также содержит поле типа Dictionary<Stop, TimeSpan>, которое задаёт отображение каждой остановки на время стоянки. Обратите внимание, что в качестве ключа ассоциативного массива используется уже готовый объект Stop, а не название остановки.

**{Оценка: 2.25} Задание.**

* + Посмотрите на конструкторы ActiveRoute: второй принимает полную информацию о маршруте, а первый — всю информацию о маршруте за исключением прикреплённого поезда. Поезд может быть удобно прикрепить в последнюю очередь с помощью сеттера.

Исправьте первый конструктор и сеттер SetTrainInfo таким образом, чтобы выполнялись необходимые проверки и выбрасывались исключения. Ссылка на прикреплённый поезд и информация о длительности стоянок не могут быть nil, время посадки не может быть раньше текущего времени.

* + Исправьте функцию LoadActiveRoutesFromTextFile так, чтобы она загружала в список информацию обо всех маршрутах, находящихся в файле, а не только о первом.  
    ***Указание.*** Используйте [Select](https://msdn.microsoft.com/ru-ru/library/bb548891%28v=vs.110%29.aspx) (посмотрите пример с Select и round в разделе комментариев). Можно пока не анализировать подробно, как выполняется парсинг файла, достаточно знать, что функция ParseActiveRoute по строке возвращает объект соответствующего активного маршрута.
  + Из файла (функцией LoadActiveRoutesFromTextFile) загружается вся информация о маршруте, за исключением прикреплённого поезда.  
    В основной программе, используя список имеющихся поездов trains и список маршрутов routes, прикрепите ко всем маршрутам поезда: к 0-му маршруту 0-й поезд, к 1-му маршруту — 1-й поезд и так далее (поездов больше, чем маршрутов).  
    ***Указание.*** Обратите внимание, что у нас есть приватный сеттер в ActiveRoute. Его можно использовать, чтобы сделать свойство TrainInfo свойством на запись, либо просто сделать сеттер публичным.  
    Размер коллекции можно узнать с помощью свойства Count.

1. Теперь пришло время подумать над генерацией расписания станций. Зная скорость поезда, информацию об остановках и соответствующих им расстояниях, можно полностью рассчитать расписание движение поезда по маршруту:
   * время отправления поезда из пункта назначения равно времени посадки плюс время стоянки;
   * время прибытия поезда на *i*-ую остановку равно чистому времени поезда в пути плюс сумма длительностей остановок на всех предыдущих остановках.

Возможная реализация такова:

* + для каждого поезда сгенерировать список с информацией о расписании для каждой остановки (время прибытия, время отправления и, возможно, чем-то ещё).
  + имея такую информацию для каждого поезда, заполнить расписание для каждой станции.

**Замечание.** Обратите внимание, что в соответствии со спецификацией задания удобно иметь расписание станции отсортированным по времени прибытия поездов на станцию. Это можно реализовать с помощью двусвязного списка, если добавлять запись о прибытии поезда каждый раз в нужное место в зависимости от времени его прибытия.

1. **{Оценка: 3.25}** Определите класс TrainStop с информацией о времени прибытия и времени отправления поезда. Рекомендуется добавить в класс поле типа ActiveRoute. Обо всём остальном подумаете позже.
   * Не забывайте о проверке входных данных, исключениях и инкапсуляции (например, поля должны быть приватными).
   * Реализуйте метод ToString по аналогии с классами, определёнными в модуле. В основной программе создайте объект TrainStop и убедитесь, что ToSring работает.
2. **{Оценка: 4}** Добавьте в модуль функцию, которая по объекту класса ActiveRoute возвращает List<TrainStop> — список с информацией о времени прибытия и отправления для каждой остановки.  
   ***Указание.*** Используйте возможности типов DateTime и TimeSpan (примеры использования см. [здесь](http://edu.mmcs.sfedu.ru/pluginfile.php/15950/mod_folder/content/0/datetime.pas?forcedownload=1)).  
   Самое трудное — подсчитать интервал времени движения поезда. Если известна скорость поезда *S* (км/час) и расстояние *D* (км) до некоторой остановки, то время в пути *T* до этой остановки составляет (*D* / *S*) часов. Среди конструкторов TimeSpan нет такого, который принимает вещественное значение часов. Можно пересчитать время движения в минутах (60 *T*) и округлить до целого (стандартная функция round); тогда можно вспользоваться конструктором TimeSpan с целочисленными параметрами для часов, минут, секунд.
3. В основной программе сформируйте информацию об остановках для каждой станции, распечатайте. Напишите несколько автоматических тестов.

Остальное продолжим на следующей лабораторной.

**Занятие #23. Использование стандартных классов в проекте «Расписание поездов» — продолжение**

Соблюдайте [требования к оформлению работы](http://edu.mmcs.sfedu.ru/mod/assign/view.php?id=9215), указанные в предыдущих лабораторных.

**Комментарии к теме**

Краткую информацию о классах коллекций и стандартных методах можно найти в разделе «Комментарии к теме» [прошлой лабораторной](http://edu.mmcs.sfedu.ru/mod/assign/view.php?id=10267).

**Лабораторная работа**

Оценка: 4 «грязных» балла.

**Требования к выполнению лабораторной**

* Решения задач оформляются в виде подпрограмм.
* Подпрограммы должны сопровождаться документирующими комментариями (с кратким описанием назначения подпрограммы).
* Подпрограммы должны содержать проверку входных данных с помощью *исключений*.
* Подпрограммы и типы данных следует определять в модуле TrainsManagement.pas, демонстрацию проводить в основной программе TrainsDemo.pas.
* Давайте *понятные имена* классам, полям, свойствам и методам. Они должны отражать своё назначение.

**Задание. Проект «Расписание поездов»**

1. **Скачайте** [**последнюю версию PascalABC.NET**](http://pascalabc.net/downloads/PascalABCNETMiniSetup.exe)**!**
2. Если вы не были или не закончили [прошлую лабораторную](http://edu.mmcs.sfedu.ru/mod/assign/view.php?id=10267), начните с неё.

Продолжаем реализацию системы генерации Расписания поездов.

1. **{Оценка: 3}** Разработайте класс StationTimetable для хранения информации о прибывающих поездах некоторой станции (это только расписание, имя станции не нужно). Этот класс нужен для того, чтобы хранить информацию о прибытии поездов в отсортированном виде (по времени прибытия на станцию). Это очень похоже на класс очереди с приоритетом на базе отсортиованного списка, который мы писали раньше.  
   Метод ToString класса StationTimetable должен распечатывать табло расписания в соответствии со спецификацией задания. Продемонстрируйте печать табло для некоторых станций в основной программе.

***Указания.***

* + Используйте внутри StationTimetable двусвязный список LinkedList<TrainStop>. В конструкторе класса StationTimetable (без параметров) следует создать этот список (сначала он пуст).
  + Добавьте в класс StationTimetable метод, который добавляет объект TrainStop в нужное место списка (в порядке времени прибытия на станцию).
  + При печати табло расписания, необходимо указывать номер маршрута и некоторую другую информацию о маршруте. Такая информация есть в TrainStop благодаря полю типа ActiveRoute.

1. **{Оценка: 4}** Напишите функцию, которая по набору активных маршрутов возвращает ассоциативный массив Dictionary<string, StationTimetable> (ключи — названия станций/остановок).  
   **Замечание.** Найдите в коллекции метод, который позволит вам по списку маршрутов List<ActiveRoute> получить список всех остановок на этих маршрутах List<TrainStop>, не используя цикл.

***Указание.*** В TrainStop не хватает информации о том, какой именно остановке маршрута соответствует информация о прибытии поезда. Можно, например, добавить в TrainStop индекс остановки в списке остановок маршрута.

1. Измените скорости поездов. Убедитесь, что расписание меняется.
2. Добавьте в поезд (Train) информацию о количестве пассажиров. Исправьте метод ToString и функцию загрузки информации о поездах из файла: для каждого поезда в файле должна быть указана пассажировместимость.
3. Добавьте в класс StationTimetable метод, который по имени станции возвращает двусвязный список поездов, следующих до этой станции, упорядоченных по времени прибытия на желаемую станцию.
4. Добавьте в файл с информацией о маршрутах ещё пару маршрутов, чтобы они проходили через некоторые уже используемые станции. Добавьте поезда, если нужно. Проверьте работу системы в основной программе.
5. Определите класс для системы расписаний всех станций (значение типа Dictionary<string, StationTimetable>, которое мы получали в задании #2, удобно сделать полем этого класса). Добавьте возможность оперативно менять расписание поезда на линии: например, увеличивать/уменьшать время стоянки на некоторой остановке из-за форс-мажорных обстоятельств. При этом необходимо обновлять расписание на всех станциях, через которые проходит этот поезд.