МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**Отчет**

**по лабораторной работе № 2-4**

**«**Объектно-Ориентированное Программирование. Инициализация состояния объекта. Статические компоненты класса. Инкапсуляция. Повторное использование кода. Наследование**»**

по дисциплине: «КПО»

Выполнили: студенты гр. 10701121

Журович К.С.

Гвоздь В.И.

Асанов Н.К.

Принял: преподаватель Станкевич С.Н.

Минск 2023

**Цель работы:**

**1)**Научиться грамотно использовать соответствующие средства, предоставляемые языком Java, для первоначальной инициализации состояния объекта, а также изучить истинное предназначение статических компонентов класса в языке Java.

**2)** Углубить свои фундаментальные знания в использовании методологии ООП, а также научиться практически применять инкапсуляцию с использованием средств, которые предоставляет язык Java.

**3)** Углубить свои фундаментальные знания в использовании методологии ООП, а также научиться практически применять инструменты языка Java для повторного использования кода в виде ассоциации, наследования, агрегации, композиции и делегирования.

**Задание**

**1)** Необходимо в проект, который был спроектирован и разработан в предыдущей лабораторной работе, внести следующие дополнения:

− для грамотной инициализации состояния объектов соответствующей предметной области добавить всевозможные средства инициализации, которые предоставляет язык Java (блоки инициализации, конструктор по умолчанию, конструкторы с параметрами, конструктор-копирования и т.д.);

− проанализировав соответствующую предметную область добавить в проект статические компоненты класса и возможность их первоначальной инициализации с помощью средств, который предоставляет язык Java. Дополнительно необходимо проанализировать стадии и способы инициализации как состояния объектов, так и состояния соответствующих объектов классов (объектов класса Class), а также их очередность вызова JVM. Привести анализ результатов и соответствующие выводы в отчёте.

**2)** Необходимо в проект, который был спроектирован и разработан в предыдущей лабораторной работе, внести следующие изменения и дополнения:

− скрыть реализацию всех компонентов и структур данных проекта, т.е. инкапсулировать все поля классов и методы, которые предназначены для внутреннего использования, с использованием модификаторов доступа языка Java, и предоставить только интерфейсную часть для внешнего взаимодействия;

− ввести, где это необходимо, высокоуровневые объекты-контейнеры, которые инкапсулируют структуру хранения множества объектов предметной области;   
− убрать из класса-контроллера код по инициализации объектов предметной области и ввести соответствующие программные компоненты, которые и будут предназначены для создания и инициализации объектов предметной области, т.е. использовать компоненты в виде фабрик ил строителей («креаторов»).

**3)** Необходимо произвести рефакторинг программной системы, созданной в предыдущей лабораторной работе, следующим образом:

− классы, описывающие объекты соответствующей предметной области (бизнес-объекты), должны быть сведены в иерархическую структуру (произвести, где это необходимо, классификацию типов); к примеру, в предыдущей лабораторной работе была только сущность автомобиль/автотранспорт, а теперь должна быть иерархия автотранспорта с соответствующими характеристиками: легковой автомобиль (седан, универсал, хэтчбэк, …), грузовой автомобиль (фура, самосвал, бетономешалка ,…), пассажирский автомобиль (автобус, микроавтобус, минивэн, …) и т.д., никто никого не ограничивает в фантазиях;

− логика системы должна быть реализована внутри соответствующих функциональных классов;

− логика системы и большинство других компонентов должны зависеть преимущественно только от абстракции, а не от реализаций;

− необходимо дополнительно для безопасности выполнения кода добавить по возможности в методы бизнес логики проверку входящих объектов на соответствие типа, с которым должна взаимодействовать данная логика.

**Результаты выполнения работы**

**1.1. Разработка предметной области**

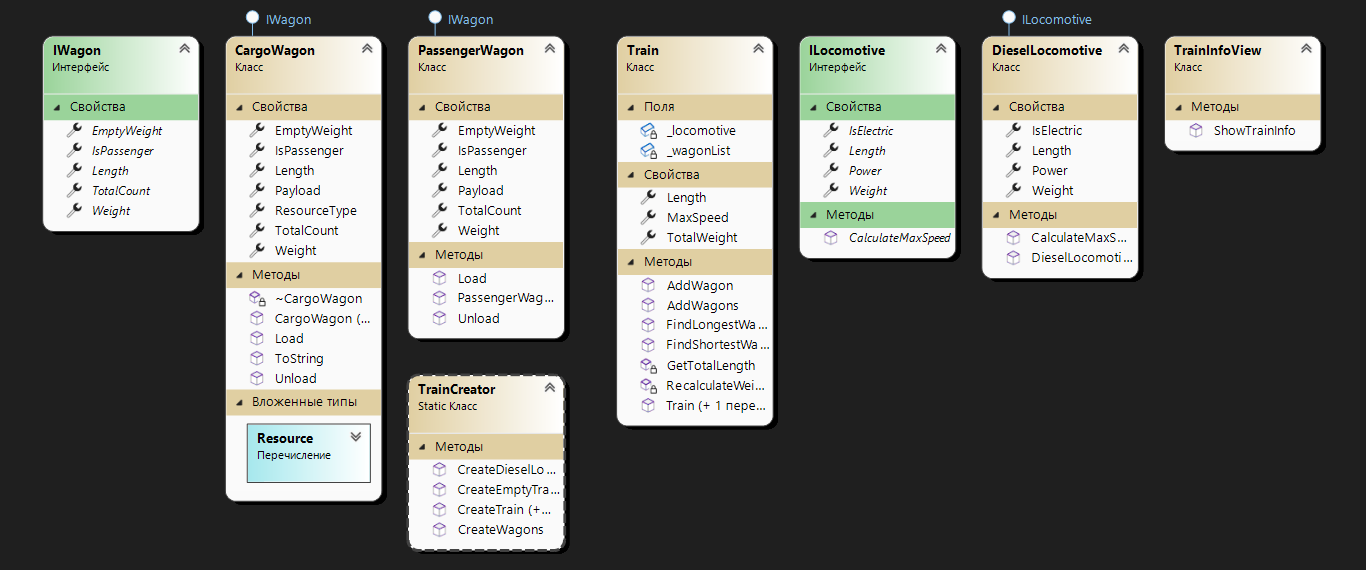
****

Рис. 1 – Диаграмма классов, созданная в среде Visual Studio

**2.2. Листинг программы**

**/Controller**

**//TrainCreator**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RailwayTransport.Controller

{

public static class TrainCreator

{

public static List<CargoWagon> CreateWagons(int count)

{

count = count < 1 ? 1 : count;

List<CargoWagon> cargoWagons = new();

Random random = new Random();

for (int i = 0; i < count; i++)

{

//Генерируем параметры

double length = random.Next(11, 17);

double emptyWeight = random.Next(18, 31);

double payLoad = random.Next(45, 96);

CargoWagon.Resource resource = (CargoWagon.Resource)random.Next(1, 6);

int loadPercentage = random.Next(1, 101);

//Инициализируем вагон

CargoWagon wagon = new CargoWagon(length, emptyWeight, payLoad);

//Загружаем вагон

wagon.Load(resource, loadPercentage);

//Добавляем в список

cargoWagons.Add(wagon);

}

return cargoWagons;

}

public static DieselLocomotive CreateDieselLocomotive()

{

return new DieselLocomotive();

}

public static Train CreateEmptyTrain()

{

return new Train(CreateDieselLocomotive());

}

public static Train CreateTrain(int wagonsCount)

{

wagonsCount = wagonsCount > 100? 100 : wagonsCount;

var locomotive = CreateDieselLocomotive();

var wagons = CreateWagons(wagonsCount);

return new Train(locomotive, wagons);

}

public static Train CreateTrain(DieselLocomotive locomotive, IEnumerable<IWagon> wagons)

{

return CreateTrain(locomotive, wagons);

}

}

}

**/Model**

**//Locomotives**

**///DieselLocomotive**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace RailwayTransport

{

public class DieselLocomotive : ILocomotive

{

public double Length { get; private set; }

public double Power { get; private set; }

public double Weight { get; private set; }

public bool IsElectric { get; private set; }

public DieselLocomotive(double length, double power, double weight)

{

Length = length;

Power = power;

Weight = weight;

IsElectric = false;

}

public DieselLocomotive()

{

Length = 13.5;

Power = 7500;

Weight = 100;

IsElectric = false;

}

public DieselLocomotive(DieselLocomotive locomotive)

{

Length = locomotive.Length;

Power = locomotive.Power;

Weight = locomotive.Weight;

IsElectric = locomotive.IsElectric;

}

public double CalculateMaxSpeed(double totalWeight)

{

return Power / totalWeight \* 24d;

}

public static double CalculateMaxSpeed(double totalWeight, double power)

{

return power / totalWeight \* 24d;

}

}

}

**///ILocomotive**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace RailwayTransport

{

public interface ILocomotive

{

double Length { get; }

double Power { get; }

double Weight { get; }

bool IsElectric { get; }

public double CalculateMaxSpeed(double totalWeight);

}

}

**//Trains**

**///Train**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace RailwayTransport

{

public class Train

{

List<IWagon> \_wagonList;

ILocomotive \_locomotive;

public double TotalWeight { get; private set; }

public double Length { get => GetTotalLength(); }

public double MaxSpeed { get => \_locomotive.CalculateMaxSpeed(TotalWeight); }

public Train(ILocomotive locomotive)

{

\_locomotive = locomotive;

\_wagonList = new List<IWagon>();

RecalculateWeight();

}

public Train(ILocomotive locomotive, IEnumerable<IWagon> wagons) : this(locomotive)

{

AddWagons(wagons);

}

public void AddWagon(IWagon wagon)

{

\_wagonList.Add(wagon);

RecalculateWeight();

}

public void AddWagons(IEnumerable<IWagon> wagons)

{

foreach (var wagon in wagons)

{

\_wagonList.Add(wagon);

}

RecalculateWeight();

}

private void RecalculateWeight()

{

double weight = \_locomotive.Weight;

foreach (var wagon in \_wagonList)

{

weight += wagon.Weight;

}

TotalWeight = weight;

}

public IWagon FindShortestWagon()

{

if (\_wagonList.Count == 0) return null;

IWagon shortestWagon = \_wagonList[0];

foreach (var wagon in \_wagonList)

{

if (wagon.Length < shortestWagon.Length)

{

shortestWagon = wagon;

}

}

return shortestWagon;

}

public IWagon FindLongestWagon()

{

if (\_wagonList.Count == 0) return null;

IWagon longestWagon = \_wagonList[0];

foreach (var wagon in \_wagonList)

{

if (wagon.Length > longestWagon.Length)

{

longestWagon = wagon;

}

}

return longestWagon;

}

private double GetTotalLength()

{

double totalLength = \_locomotive.Length;

foreach (var wagon in \_wagonList)

{

totalLength += wagon.Length;

}

return totalLength;

}

}

}

**//Wagons**

**///CargoWagon**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace RailwayTransport

{

public class CargoWagon : IWagon

{

public enum Resource

{

Nothing, Coal, Wood, AgroCombine, Fuel, Animals

}

public double Length { get; private set; }

public bool IsPassenger { get; private set; }

public double EmptyWeight { get; private set; }

public double Weight { get; private set; }

public double Payload { get; private set; }

public Resource ResourceType { get; private set; }

public static int TotalCount { get; private set; }

public CargoWagon()

{

Length = 15d;

EmptyWeight = 25d;

Weight = EmptyWeight;

Payload = 75d;

ResourceType = Resource.Nothing;

IsPassenger = false;

TotalCount++;

}

public CargoWagon(double length, double emptyWeight, double payload)

{

Length = length;

EmptyWeight = emptyWeight;

Weight = emptyWeight;

Payload = payload;

ResourceType = Resource.Nothing;

IsPassenger = false;

TotalCount++;

}

public CargoWagon(CargoWagon wagon)

{

Length = wagon.Length;

EmptyWeight = wagon.EmptyWeight;

Weight = wagon.Weight;

Payload = wagon.Payload;

ResourceType = wagon.ResourceType;

IsPassenger = false;

TotalCount++;

}

static CargoWagon()

{

TotalCount = 0;

}

public void Unload()

{

ResourceType = Resource.Nothing;

Weight = EmptyWeight;

}

public void Load(Resource resource, int loadPercentage)

{

ResourceType = resource;

loadPercentage = loadPercentage > 100? 100 : loadPercentage < 0 ? 0 : loadPercentage;

Weight += Payload \* loadPercentage / 100d;

}

public override string ToString()

{

return $"Грузовой вагон, везущий {ResourceType.ToString()}" +

$"\nДлина: {Length} метров" +

$"\nВес: {Weight.ToString("f2")} тонн";

}

~CargoWagon()

{

TotalCount--;

}

}

}

**///PassengerWagon**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace RailwayTransport

{

public class PassengerWagon : IWagon

{

public double Length { get; private set; }

public bool IsPassenger { get; private set; }

public double EmptyWeight { get; private set; }

public double Weight { get; private set; }

public int Payload { get; private set; }

public static int TotalCount { get; private set; }

public PassengerWagon(double length, double emptyWeight, int maxPassengers)

{

Length = length;

EmptyWeight = emptyWeight;

IsPassenger = true;

Weight = emptyWeight;

Payload = maxPassengers;

TotalCount++;

}

public PassengerWagon()

{

Length = 25.0;

EmptyWeight = 56.0;

IsPassenger = true;

Weight = EmptyWeight;

Payload = 64;

TotalCount++;

}

public PassengerWagon(PassengerWagon wagon)

{

Length = wagon.Length ;

EmptyWeight = wagon.EmptyWeight;

IsPassenger = wagon.IsPassenger;

Weight = wagon.Weight;

Payload = wagon.Payload;

TotalCount++;

}

static PassengerWagon()

{

TotalCount = 0;

}

public void Unload()

{

Weight = EmptyWeight;

}

public void Load(int passengersCount)

{

passengersCount = passengersCount > Payload ? Payload : passengersCount < 0 ? 0 : passengersCount;

Weight += passengersCount \* 0.07d;

}

~PassengerWagon()

{

TotalCount--;

}

}

}

**///IWagon**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace RailwayTransport

{

public interface IWagon

{

public double Length { get; }

public bool IsPassenger { get; }

public double EmptyWeight { get; }

public double Weight { get; }

public static int TotalCount { get; }

}

}

**/View**

**//TrainInfoView**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RailwayTransport.View

{

internal class TrainInfoView

{

public static void ShowTrainInfo(Train train)

{

Console.WriteLine("Длина поезда: " + train.Length + "м");

Console.WriteLine("\nСамый длинный вагон:\n" + train.FindLongestWagon().ToString());

Console.WriteLine("\nСамый короткий вагон:\n" + train.FindShortestWagon().ToString());

Console.WriteLine("\nВес поезда: " + train.TotalWeight.ToString("f2") + " т");

Console.WriteLine("\nМаксимальная скорость: " + train.MaxSpeed.ToString("f1") + " км/ч");

}

}

}

**//Program**

using RailwayTransport.Controller;

using RailwayTransport.View;

namespace RailwayTransport

{

internal class Program

{

static void Main(string[] args)

{

var train = TrainCreator.CreateTrain(wagonsCount: 30);

TrainInfoView.ShowTrainInfo(train);

}

}

}

**3. Выполнение программы**

- Была реализована архитектура MVC. Добавлен класс «TrainCreator.cs» в модуле Controller для инициализации компонентов поезда, а также класс «TrainInfoView.cs» в модуле View для вывода информации о поезде в консоль.

- Были добавлены конструкторы по умолчанию, копирования деструкторы для классов «CargoWagon», «PassengerWagon» и «DieselLocomotive».

- Были добавлены статические переменные и методы.

- Также была введена проверка некоторых входных параметров на корректность.

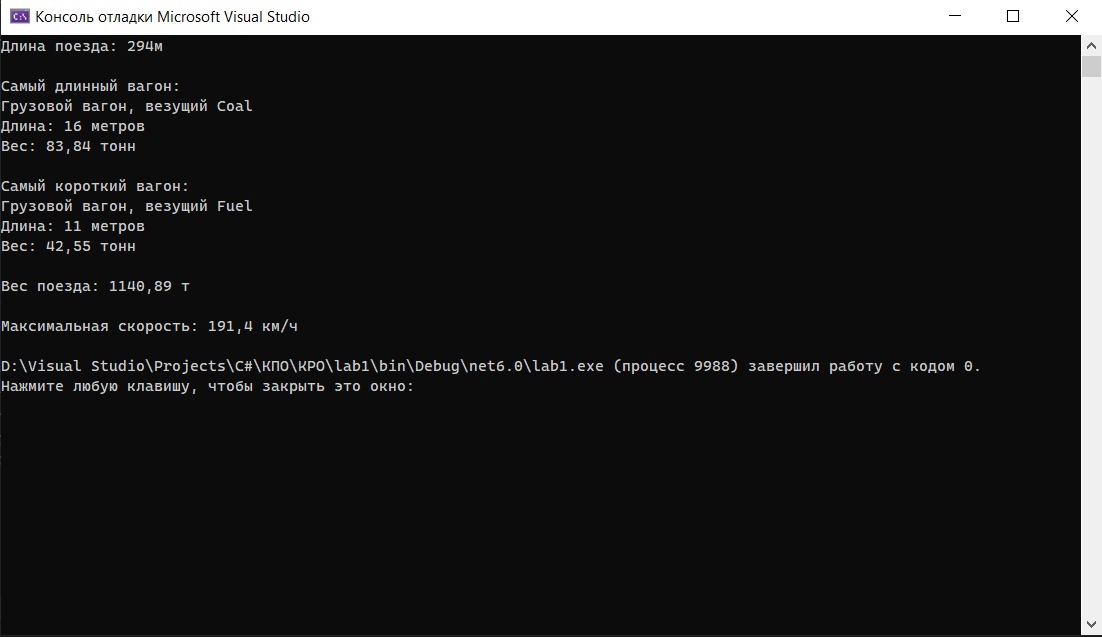


Рис. 4 – Результат выполнения метода Main

**4. Выводы**

Научились грамотно использовать соответствующие средства, предоставляемые языком C#, для первоначальной инициализации состояния объекта, а также изучили истинное предназначение статических компонентов класса в языке C#. Углубили свои фундаментальные знания в использовании методологии ООП, а также научились практически применять инкапсуляцию с использованием средств, которые предоставляет язык C#. Углубили свои фундаментальные знания в использовании методологии ООП, а также научились практически применять инструменты языка C# для повторного использования кода в виде ассоциации, наследования, агрегации, композиции и делегирования.