Московский государственный технический университет им. Н.Э. Баумана

Факультет “Радиотехнический”
Кафедра “Системы обработки информации и управления”

Курс «Парадигмы и конструкции языков программирования»

Отчет по рубежному контролю №2
Вариант №18

Выполнил:	Проверил:
студент группы РТ5-31Б:	преподаватель каф. ИУ5
Филатов И. В.	Гапанюк Ю. Е.

Москва, 2025 г.

Текст программы

Program.cs

```
namespace rk1
{
   internal class Program
   {
      static void Main(string[] args)
      {
         var dataService = DataInitializer.CreateSampleDataService();

         var reportGenerator = new ReportGenerator(dataService);
         reportGenerator.GenerateAllReports();
      }
   }
}
```

Detail.cs

```
namespace rk1
{
   public class Detail
   {
      public int Id { get; set; }
      public string Name { get; set; }
      public decimal Price { get; set; }
      public int SupplierId { get; set; }

      public Detail(int id, string name, decimal price, int supplierId)
      {
         Id = id;
         Name = name;
         Price = price;
         SupplierId = supplierId;
      }
   }
}
```

Supplier.cs

```
namespace rk1
{
   public class Supplier
   {
      public int Id { get; set; }
      public string Name { get; set; }

      public Supplier(int id, string name)
      {
         Id = id;
         Name = name;
      }
   }
}
```

DetailSupplier.cs
```
namespace rk1
{
    public class DetailSupplier
    {
        public int DetailId { get; set; }
        public int SupplierId { get; set; }

        public DetailSupplier(int detailId, int supplierId)
        {
            DetailId = detailId;
            SupplierId = supplierId;
        }
    }
}
```

DataInitializer.cs
```
namespace rk1
{
    public static class DataInitializer
    {
        public static DataService CreateSampleDataService()
        {
            var suppliers = new List<Supplier>
            {
                new(1, "Основной отдел поставщиков"),
                new(2, "Вспомогательный отдел комплектующих"),
                new(3, "Склад запчастей болты и гайки"),
                new(4, "Другой отдел поставщиков"),
                new(5, "отдел перекупа деталей"),
                new(6, "Поставщик про100деталь")
            };

            var details = new List<Detail>
            {
                new(1, "Болт M8", 15.50m, 1),
                new(2, "Гайка M10", 8.75m, 2),
                new(3, "Шайба A4", 3.20m, 3),
                new(4, "Винт M6", 12.80m, 3),
                new(5, "Подшипник 6205", 45.90m, 3)
            };

            var detailSuppliers = new List<DetailSupplier>
            {
                new(1, 1),
                new(2, 2),
                new(3, 3),
                new(4, 3),
                new(5, 3),
                new(1, 4),
                new(2, 5),
                new(3, 6),
                new(4, 6),
                new(5, 6)
            };

            return new DataService(suppliers, details, detailSuppliers);
        }
    }
}
```

DataService.cs

```
namespace rk1
{
    public class DataService
    {
        public List<Supplier> Suppliers { get; }
        public List<Detail> Details { get; }
        public List<DetailSupplier> DetailSuppliers { get; }

        public DataService(
            List<Supplier> suppliers,
            List<Detail> details,
            List<DetailSupplier> detailSuppliers)
        {
            Suppliers = suppliers ?? new List<Supplier>();
            Details = details ?? new List<Detail>();
            DetailSuppliers = detailSuppliers ?? new List<DetailSupplier>();
        }

        // Задание Е1: Поставщики с "отдел" и их детали
        public IEnumerable<SupplierDetailsResult> GetDepartmentSuppliersWithDetails()
        {
            var manyToManyTemp = from supplier in Suppliers
                        join ds in DetailSuppliers on supplier.Id equals ds.SupplierId
                        select new { SupplierName = supplier.Name, ds.SupplierId, ds.DetailId };

            var result = from supplier in Suppliers
                    where supplier.Name.Contains("отдел", StringComparison.OrdinalIgnoreCase)
                    join ds in DetailSuppliers on supplier.Id equals ds.SupplierId into supplierDetails
                    from sd in supplierDetails
                    join detail in Details on sd.DetailId equals detail.Id
                    group detail by supplier into supplierGroup
                    select new SupplierDetailsResult
                    {
                        SupplierName = supplierGroup.Key.Name,
                        Details = supplierGroup.Select(d => new DetailInfo
                        {
                            Name = d.Name,
                            Price = d.Price
                        }).ToList()
                    };

            return result;
        }

        // Задание Е2: Средняя цена деталей по поставщикам
        public IEnumerable<SupplierAveragePriceResult> GetSuppliersWithAveragePrice()
        {
            var result = from supplier in Suppliers
                         join ds in DetailSuppliers on supplier.Id equals ds.SupplierId
                         join detail in Details on ds.DetailId equals detail.Id
                         group detail by supplier into supplierGroup
                         let avgPrice = Math.Round(supplierGroup.Average(d => d.Price), 2)
                         select new SupplierAveragePriceResult
                         {
                             SupplierName = supplierGroup.Key.Name,
                             AveragePrice = avgPrice,
                             DetailsCount = supplierGroup.Count()
                         };

            return result.OrderBy(x => x.AveragePrice);
        }

        // Задание Е3: Детали на букву "Б" и их поставщики
        public IEnumerable<DetailWithSupplierResult> GetDetailsStartingWithB()
        {
            var manyToManyTemp = from supplier in Suppliers
                                 join ds in DetailSuppliers on supplier.Id equals ds.SupplierId
                                 select new { SupplierName = supplier.Name, ds.SupplierId, ds.DetailId };

            var result = from detail in Details
                         where detail.Name.StartsWith("Б", StringComparison.OrdinalIgnoreCase)
                         join temp in manyToManyTemp on detail.Id equals temp.DetailId into
detailSuppliersTemp
                         from dst in detailSuppliersTemp
                         join supplier in Suppliers on dst.SupplierId equals supplier.Id
                         select new DetailWithSupplierResult
                         {
                             DetailName = detail.Name,
                             DetailPrice = detail.Price,
                             SupplierName = supplier.Name
                         };

            return result;
        }
    }

    public class SupplierDetailsResult
    {
        public string SupplierName { get; set; } = string.Empty;
        public List<DetailInfo> Details { get; set; } = new();
    }

    public class DetailInfo
    {
        public string Name { get; set; } = string.Empty;
        public decimal Price { get; set; }
    }

    public class SupplierAveragePriceResult
    {
        public string SupplierName { get; set; } = string.Empty;
        public decimal AveragePrice { get; set; }
        public int DetailsCount { get; set; }
    }

    public class DetailWithSupplierResult
    {
        public string DetailName { get; set; } = string.Empty;
        public decimal DetailPrice { get; set; }
        public string SupplierName { get; set; } = string.Empty;
    }
}
```

ReportGenerator.cs

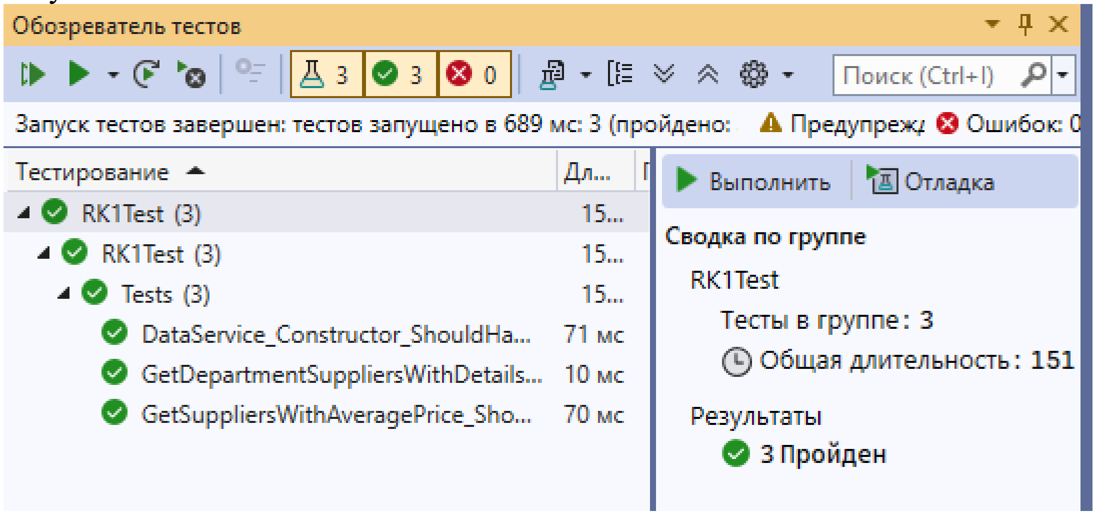
```
namespace rk1
{
    public class ReportGenerator
    {
        private readonly DataService _dataService;

        public ReportGenerator(DataService dataService)
        {
            _dataService = dataService;
        }

        public void GenerateAllReports()
        {
            GenerateDepartmentSuppliersReport();
            GenerateAveragePriceReport();
            GenerateDetailsStartingWithBReport();
        }

        public void GenerateDepartmentSuppliersReport()
        {
            Console.WriteLine("Задание Е1");
            Console.WriteLine("Список всех поставщиков, у которых в названии присутствует
слово \"отдел\", и список их деталей:");

            var results = _dataService.GetDepartmentSuppliersWithDetails();

            foreach (var item in results)
            {
                Console.WriteLine($"\nПоставщик: {item.SupplierName}");
                foreach (var detail in item.Details)
                {
                    Console.WriteLine($"  Деталь: {detail.Name}, Цена: {detail.Price} руб.");
                }
            }
        }

        public void GenerateAveragePriceReport()
        {
            Console.WriteLine("\nЗадание Е2");
            Console.WriteLine("Список поставщиков со средней ценой деталей в каждом
отделе, отсортированный по средней цене:");

            var results = _dataService.GetSuppliersWithAveragePrice();

            foreach (var item in results)
            {
                Console.WriteLine($"Поставщик: {item.SupplierName}, Средняя цена:
{item.AveragePrice} руб. (детали: {item.DetailsCount})");
            }
        }

        public void GenerateDetailsStartingWithBReport()
        {
            Console.WriteLine("\nЗадание Е3");
            Console.WriteLine("Список всех деталей, у которых название начинается с буквы
\"Б\", и названия их поставщиков:");

            var results = _dataService.GetDetailsStartingWithB();

            foreach (var item in results)
            {
                Console.WriteLine($"Деталь: {item.DetailName}, Цена: {item.DetailPrice} руб. ->
Поставщик: {item.SupplierName}");
            }
        }
    }
}
```

UnitTest1.cs

```
namespace RK1Test
{
  public class Tests
  {
    [Test]
    public void GetDepartmentSuppliersWithDetails_ShouldHandleSupplierWithoutDetails()
    {
        var suppliers = new List<Supplier>
        {
            new(1, "Отдел без деталей")
        };

        var details = new List<Detail>();
```

```
            var detailSuppliers = new List<DetailSupplier>();

            var dataService = new DataService(suppliers, details, detailSuppliers);

            var result = dataService.GetDepartmentSuppliersWithDetails().ToList();

            Assert.That(result, Is.Empty, "Поставщик без деталей не должен попасть в
результат");
        }
        [Test]
        public void GetSuppliersWithAveragePrice_ShouldCalculateCorrectlyWithSingleDetail()
        {
            var suppliers = new List<Supplier>
            {
                new(1, "Поставщик с одной деталью")
            };

            var details = new List<Detail>
            {
                new(1, "Деталь", 7.89m, 1)
            };

            var detailSuppliers = new List<DetailSupplier>
            {
                new(1, 1)
            };

            var dataService = new DataService(suppliers, details, detailSuppliers);

            var result = dataService.GetSuppliersWithAveragePrice().ToList();

            Assert.That(result, Has.Count.EqualTo(1));
            Assert.That(result[0].AveragePrice, Is.EqualTo(7.89m));
            Assert.That(result[0].DetailsCount, Is.EqualTo(1));
        }

        [Test]
        public void DataService_Constructor_ShouldHandleNullParameters()
        {
            var dataService = new DataService(null, null, null);

            Assert.That(dataService.Suppliers, Is.Not.Null);
            Assert.That(dataService.Details, Is.Not.Null);
            Assert.That(dataService.DetailSuppliers, Is.Not.Null);
            Assert.That(dataService.Suppliers, Is.Empty);
            Assert.That(dataService.Details, Is.Empty);
            Assert.That(dataService.DetailSuppliers, Is.Empty);

            Assert.DoesNotThrow(() =>
dataService.GetDepartmentSuppliersWithDetails().ToList());
            Assert.DoesNotThrow(() => dataService.GetSuppliersWithAveragePrice().ToList());
            Assert.DoesNotThrow(() => dataService.GetDetailsStartingWithB().ToList());
        }
    }
}
```

Результаты



Обозреватель тестов

Запуск тестов завершен: тестов запущено в 689 мс: 3 (пройдено: ⚠ Предупреж ✖ Ошибок: 0

```
Тестирование                                  Дл...
◢ ✓ RK1Test (3)                               15...
  ◢ ✓ RK1Test (3)                             15...
    ◢ ✓ Tests (3)                             15...
        ✓ DataService_Constructor_ShouldHa... 71 мс
        ✓ GetDepartmentSuppliersWithDetails... 10 мс
        ✓ GetSuppliersWithAveragePrice_Sho... 70 мс
```

Сводка по группе

RK1Test

Тесты в группе: 3

Общая длительность: 151

Результаты

✓ 3 Пройден