|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет социально-экономических и компьютерных наук* |
|  |
| Гуцол Степан Дмитриевич  **ЛАБОРАТОРНАЯ РАБОТА №10**  по направлению подготовки *38.03.05 Бизнес-информатика*  образовательная программа «Разработка информационных систем для бизнеса»   |  |  | | --- | --- | |  | Руководитель  Преподаватель кафедры ИТБ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Найданов И.В. |   Пермь, 2023 |

**Оглавление**

[**Часть №1** 3](#_Toc129019412)

[**1 Постановка задачи** 3](#_Toc129019413)

[**2 Диаграмма классов** 4](#_Toc129019414)

[**3 Программа (листинг)** 5](#_Toc129019415)

[**4 Код Unit-тестов** 6](#_Toc129019416)

[**5 Анализ покрытия кода тестами** 8](#_Toc129019417)

[**Часть №2** 9](#_Toc129019418)

[**1 Постановка задачи** 9](#_Toc129019419)

[**2 Диаграмма классов** 10](#_Toc129019420)

[**3 Программа (листинг)** 11](#_Toc129019421)

[**4 Код Unit-тестов** 12](#_Toc129019422)

[**5 Анализ покрытия кода тестами** 14](#_Toc129019423)

[**Часть №3 Абстрактные классы и интерфейсы** 15](#_Toc129019424)

[**1 Постановка задачи** 15](#_Toc129019425)

[**2 Диаграмма классов** 16](#_Toc129019426)

[**3 Программа (листинг)** 17](#_Toc129019427)

[**4 Код Unit-тестов** 18](#_Toc129019428)

[**5 Анализ покрытия кода тестами** 20](#_Toc129019429)

[ПРИЛОЖЕНИЕ А **Программа Program.cs** 21](#_Toc129019430)

[ПРИЛОЖЕНИЕ Б **Программа Goods.cs** 27](#_Toc129019431)

[ПРИЛОЖЕНИЕ В **Программа Product.cs** 30](#_Toc129019432)

[ПРИЛОЖЕНИЕ Г **Программа MilkProduct.cs** 32](#_Toc129019433)

[ПРИЛОЖЕНИЕ Д **Программа Toy.cs** 35](#_Toc129019434)

[ПРИЛОЖЕНИЕ Е **Программа Animal.cs** 37](#_Toc129019435)

[ПРИЛОЖЕНИЕ Ё **Программа SortByPrice.cs** 39](#_Toc129019436)

[ПРИЛОЖЕНИЕ Ж **Программа IInit.cs** 40](#_Toc129019437)

**Часть №1**

**1 Постановка задачи**

Написать программу, в которой создается иерархия классов. Записать объекты классов в массив, выполнить просмотр элементов массива. Показать использование виртуальных функций.

7 вариант - игрушка, продукция, молочная продукция, **товар**.

Необходимо использовать виртуальные функции, так как массив класса Goods будет пытаться присвоить элементам класс Goods, а с виртуальными функциями он будет у каждого элемента брать начальный класс, а также методы именно этого класса.

**2 Диаграмма классов**



**Рисунок 1 – Диаграмма классов**

**3 Программа (листинг)**

Код программы представлен в приложении. См. Приложение А

**4 Код Unit-тестов**

using lab;

namespace lab.Tests

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestGoods()

{

Goods expected = new Goods(0, 21);

Goods actual = new Goods(0, 21);

Assert.AreEqual(expected,actual);

}

[TestMethod]

public void TestGoods2()

{

Goods expected = new Goods();

Goods actual = new Goods(0, 0);

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMilkProduct()

{

MilkProduct expected = new MilkProduct(20, 30, "Test");

MilkProduct actual = new MilkProduct(20, 30, "Test");

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMilkProduct2()

{

MilkProduct expected = new MilkProduct();

MilkProduct actual = new MilkProduct(0, 0, "NoName");

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMilkProductTotalCost()

{

MilkProduct expected = new MilkProduct(25,30 ,"name");

Assert.AreEqual(expected.TotalCost(), expected.Amount\*expected.Price) ;

}

[TestMethod]

public void TestProduct()

{

Product expected = new Product(20, 30, "Test");

Product actual = new Product(20, 30, "Test");

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestProduct2()

{

Product expected = new Product();

Product actual = new Product(0, 0, "NoName");

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestProductTotalCost()

{

Product expected = new Product(25, 30, "name");

Assert.AreEqual(expected.TotalCost(), expected.Amount\*expected.Price);

}

[TestMethod]

public void TestToy()

{

Toy expected = new Toy(20, 30, "Test");

Toy actual = new Toy(20, 30, "Test");

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestToy2()

{

Toy expected = new Toy();

Toy actual = new Toy(0, 0, "NoName");

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestToyTotalCost()

{

Toy expected = new Toy(25, 30, "name");

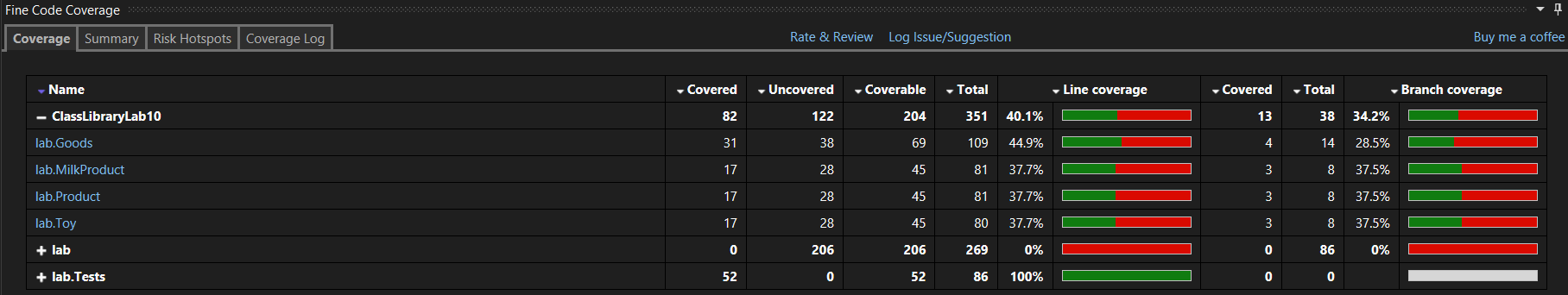
Assert.AreEqual(expected.TotalCost(), expected.Amount\*expected.Price);

}

}

}

**5 Анализ покрытия кода тестами**

****

**Рисунок 2 – покрытие кода тестами**

**Часть №2**

**1 Постановка задачи**

Реализовать метод для выполнения заданных запросов. При необходимости (для выполнения запроса) в класс могут быть добавлены новые поля (по сравнению с частью 1). В программе должно быть минимум ввода с клавиатуры. Объекты формируются с помощью ДСЧ. С клавиатуры вводятся только параметры запроса.

Реализовать не менее трех запросов, соответствующих иерархии классов (можно реализовать свои запросы).

**2 Диаграмма классов**



**Рисунок 3 – диаграмма классов**

**3 Программа (листинг)**

Код программы представлен в приложении. См. Приложение А

**4 Код Unit-тестов**

using lab;

namespace lab.Tests

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestGoods()

{

Goods expected = new Goods(0, 21);

Goods actual = new Goods(0, 21);

Assert.AreEqual(expected,actual);

}

[TestMethod]

public void TestGoods2()

{

Goods expected = new Goods();

Goods actual = new Goods(0, 0);

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMilkProduct()

{

MilkProduct expected = new MilkProduct(20, 30, "Test");

MilkProduct actual = new MilkProduct(20, 30, "Test");

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMilkProduct2()

{

MilkProduct expected = new MilkProduct();

MilkProduct actual = new MilkProduct(0, 0, "NoName");

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMilkProductTotalCost()

{

MilkProduct expected = new MilkProduct(25,30 ,"name");

Assert.AreEqual(expected.TotalCost(), expected.Amount\*expected.Price) ;

}

[TestMethod]

public void TestProduct()

{

Product expected = new Product(20, 30, "Test");

Product actual = new Product(20, 30, "Test");

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestProduct2()

{

Product expected = new Product();

Product actual = new Product(0, 0, "NoName");

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestProductTotalCost()

{

Product expected = new Product(25, 30, "name");

Assert.AreEqual(expected.TotalCost(), expected.Amount\*expected.Price);

}

[TestMethod]

public void TestToy()

{

Toy expected = new Toy(20, 30, "Test");

Toy actual = new Toy(20, 30, "Test");

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestToy2()

{

Toy expected = new Toy();

Toy actual = new Toy(0, 0, "NoName");

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestToyTotalCost()

{

Toy expected = new Toy(25, 30, "name");

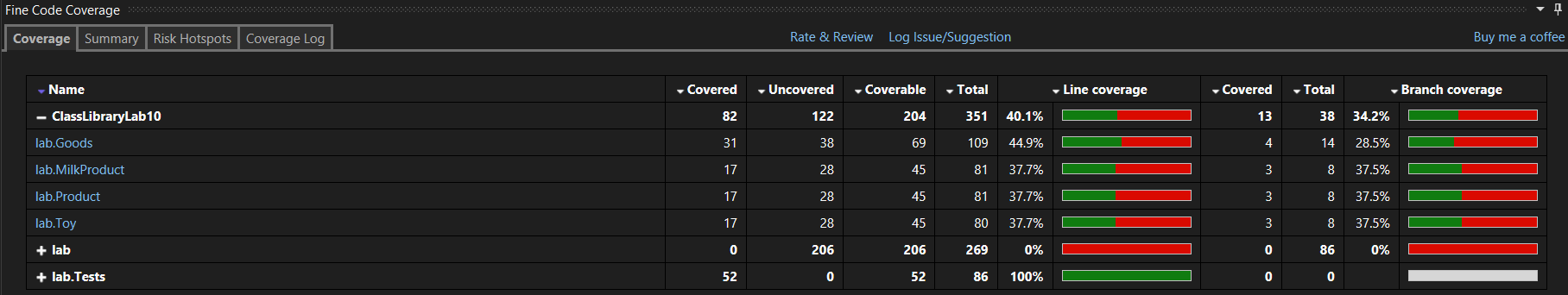
Assert.AreEqual(expected.TotalCost(), expected.Amount\*expected.Price);

}

}

}

**5 Анализ покрытия кода тестами**

****

**Рисунок 4 - покрытие кода тестами**

**Часть №3 Абстрактные классы и интерфейсы**

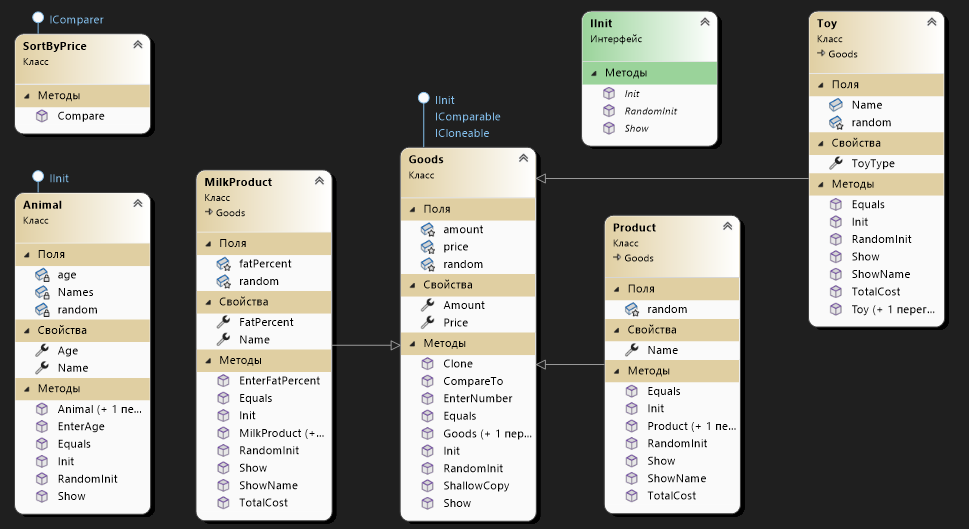
**1 Постановка задачи**

Выполнить сортировку по разным критериям (полям) с использованием стандартных интерфейсов IComparable и ICompare. Выполнить поиск элементов в упорядоченном массиве (бинарный поиск).

Выполнить создание и печать коллекции (массива), хранящего ссылки на интерфейс, содержащий описание методов для заполнения массива случайными элементами (RandomInit()), и элементами который пользователь ввел с клавиатуры (Init()).

Выполнить глубокое клонирование элементов массива с использованием стандартного интерфейса IClonable() и поверхностное копирование с помощью метода MemberwiseClone(). Показать различие между глубоким и поверхностным копированием.

**2 Диаграмма классов**



**Рисунок 5 – диаграмма классов**

**3 Программа (листинг)**

Код программы представлен в приложении. См. Приложение А

**4 Код Unit-тестов**

using ClassLibraryLab10;

using lab;

using System.Diagnostics.CodeAnalysis;

namespace lab.Tests

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestGoods()

{

Goods expected = new Goods(0, 21);

Goods actual = new Goods(0, 21);

Assert.AreEqual(expected,actual);

}

[TestMethod]

public void TestGoods2()

{

Goods expected = new Goods();

Goods actual = new Goods(0, 0);

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMilkProduct()

{

MilkProduct expected = new MilkProduct(20, 30, "Test",20);

MilkProduct actual = new MilkProduct(20, 30, "Test", 20);

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMilkProduct2()

{

MilkProduct expected = new MilkProduct();

MilkProduct actual = new MilkProduct(0, 0, "NoName",0);

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestMilkProductTotalCost()

{

MilkProduct expected = new MilkProduct(25,30 ,"name",20);

Assert.AreEqual(expected.TotalCost(), expected.Amount\*expected.Price) ;

}

[TestMethod]

public void TestProduct()

{

Product expected = new Product(20, 30, "Test");

Product actual = new Product(20, 30, "Test");

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestProduct2()

{

Product expected = new Product();

Product actual = new Product(0, 0, "NoName");

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestProductTotalCost()

{

Product expected = new Product(25, 30, "name");

Assert.AreEqual(expected.TotalCost(), expected.Amount\*expected.Price);

}

[TestMethod]

public void TestToy()

{

Toy expected = new Toy(20, 30, "Test", "TypeTest");

Toy actual = new Toy(20, 30, "Test", "TypeTest");

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestToy2()

{

Toy expected = new Toy();

Toy actual = new Toy(0, 0, "NoName","NoType");

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestToyTotalCost()

{

Toy expected = new Toy(25, 30, "name","type");

Assert.AreEqual(expected.TotalCost(), expected.Amount\*expected.Price);

}

[TestMethod]

public void TestGoodsClone()

{

Goods expected = new Goods(25, 30);

Goods actual = new Goods(123, 456);

actual= (Goods)expected.Clone();

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestAnimal()

{

Animal expected = new Animal("Test", 32);

Animal actual = new Animal("Test", 32);

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void TestAnimal2()

{

Animal expected = new Animal();

Animal actual = new Animal("NoName", 1);

Assert.AreEqual(expected, actual);

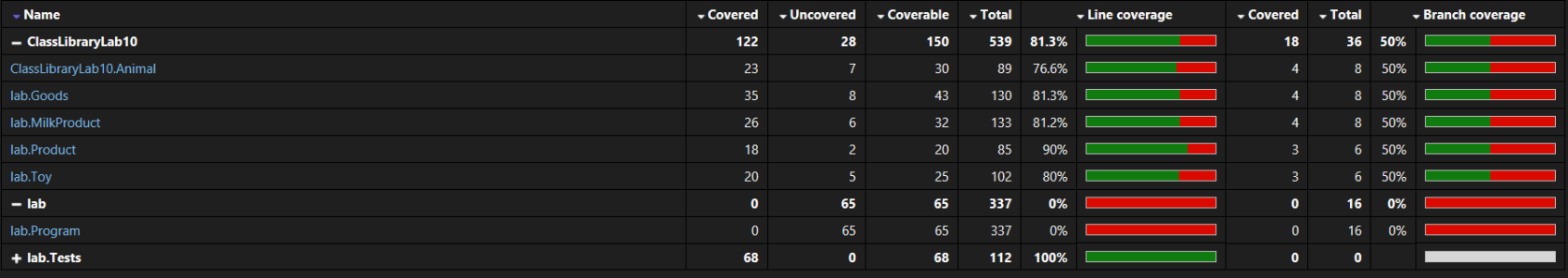
}

}

}

**5 Анализ покрытия кода тестами**

С помощью Unit-тестов удалось достичь покрытия кода в 80%, были протестированы методы инициализации, подсчёта общей стоимости, а также полного клонирования.

****

**Рисунок 6 - покрытие кода тестами**

# ПРИЛОЖЕНИЕ А **Программа Program.cs**

using System.Transactions;

namespace lab

{

internal class Program

{

static void Main(string[] args)

{

// Часть 1

Console.WriteLine("Ввод для игрушки:");

Toy toy1 = new Toy();

toy1.Init();

Toy toy2 = new Toy();

toy2.RandomInit();

Toy toy3 = new Toy();

toy3.RandomInit();

Toy toy4 = new Toy();

toy4.RandomInit();

Console.Clear();

Console.WriteLine("Ввод для продукта:");

Product product1 = new Product();

product1.Init();

Product product2 = new Product();

product2.RandomInit();

Product product3 = new Product();

product3.RandomInit();

Product product4 = new Product();

product4.RandomInit();

Console.Clear();

Console.WriteLine("Ввод для молочного продукта:");

MilkProduct milkProduct1 = new MilkProduct();

milkProduct1.Init();

MilkProduct milkProduct2 = new MilkProduct();

milkProduct2.RandomInit();

MilkProduct milkProduct3 = new MilkProduct();

milkProduct3.RandomInit();

MilkProduct milkProduct4 = new MilkProduct();

milkProduct4.RandomInit();

Console.Clear();

Goods[] arr = { toy1, toy2,toy3,toy4, product1, product2, product3, product4, milkProduct1, milkProduct2, milkProduct3, milkProduct4 };

foreach (Goods goods in arr)

{

goods.Show();

}

// Часть 2

//Первый запрос - узнать наименования товаров по отделам

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine(@"

Узнать наименования товаров в отделах:

[1] - Отдел с игрушками

[2] - Отдел с продуктами

[3] - Отдел с молочными продуктами");

int enter = 0;

do

{

enter = EnterNumber();

} while (!((enter==1)||(enter==2)||(enter==3)));

switch (enter)

{

case 1:

Console.Clear();

foreach (Goods goods in arr)

{

if (goods is Toy toy)

{

toy.ShowName();

}

}

break;

case 2:

Console.Clear();

foreach (Goods goods in arr)

{

if (goods is Product product)

{

product.ShowName();

}

}

break;

case 3:

Console.Clear();

foreach (Goods goods in arr)

{

if (goods is MilkProduct milkProduct)

{

milkProduct.ShowName();

}

}

break;

}

// Второй запрос - Узнать общую стоимость товаров по отделам

Console.ForegroundColor=ConsoleColor.Green;

Console.WriteLine(@"

Узнать общую стоимость товаров в определённом отделе:

[1] - Отдел с игрушками

[2] - Отдел с продуктами

[3] - Отдел с молочной продукцией");

int summary = 0;

do

{

enter = EnterNumber();

} while (!((enter==1)||(enter==2)||(enter==3)));

switch (enter)

{

case 1:

Console.Clear();

foreach (Goods goods in arr)

{

if (goods is Toy toy)

{

summary += toy.TotalCost();

}

}

break;

case 2:

Console.Clear();

foreach (Goods goods in arr)

{

if (goods is Product product)

{

summary += product.TotalCost();

}

}

break;

case 3:

Console.Clear();

foreach (Goods goods in arr)

{

if (goods is MilkProduct milkProduct)

{

summary += milkProduct.TotalCost();

}

}

break;

}

Console.WriteLine(@$"Сумма всех товаров в отделе: {summary} руб.");

// Третий запрос - Узнать информацию про самый дорогой и самый дешёвый товар из отдела

Console.ForegroundColor=ConsoleColor.Blue;

Console.WriteLine(@"

Узнать информацию про самый дорогой и самый дешёвый товар из отдела:

[1] - Отдел с игрушками

[2] - Отдел с продуктами

[3] - Отдел с молочной продукцией");

int max = int.MinValue;

int maxPosition = 0;

int min = int.MaxValue;

int minPosition = 0;

int count = 0;

do

{

enter = EnterNumber();

} while (!((enter==1)||(enter==2)||(enter==3)));

switch (enter)

{

case 1:

Console.Clear();

foreach (Goods goods in arr)

{

if (goods is Toy toy)

{

if (toy.Price>max)

{

max = toy.Price;

maxPosition=count;

}

if (toy.Price <min)

{

min = toy.Price;

minPosition = count;

}

}

count++;

}

break;

case 2:

Console.Clear();

foreach (Goods goods in arr)

{

if (goods is Product product)

{

if (product.Price>max)

{

max = product.Price;

maxPosition=count;

}

if (product.Price <min)

{

min = product.Price;

minPosition = count;

}

}

count++;

}

break;

case 3:

Console.Clear();

foreach (Goods goods in arr)

{

if (goods is MilkProduct milkProduct)

{

if (milkProduct.Price>max)

{

max = milkProduct.Price;

maxPosition=count;

}

if (milkProduct.Price <min)

{

min = milkProduct.Price;

minPosition = count;

}

}

count++;

}

break;

}

Console.WriteLine("Самый дешёвый товар:");

arr[minPosition].Show();

Console.WriteLine("Самый дорогой товар:");

arr[maxPosition].Show();

//Часть 3

//Интерфейсы необходимы для того, чтобы определять поведение, которое впоследствии будет реализовано в каком-то

//конкретном классе. Это необходимо при множественном наследовании, чтобы не было конфликтов среди объектов разных классов

Goods[] goodsArray = new Goods[10];

for (int i = 0; i<5; i++)

{

goodsArray[i] = new Toy();

goodsArray[i].RandomInit();

}

for (int i = 5; i<10; i++)

{

goodsArray[i] = new Goods();

goodsArray[i].RandomInit();

}

foreach (var item in goodsArray)

{

item.Show();

}

Array.Sort(goodsArray);

Console.WriteLine("Сортированный массив по количеству продукции:");

foreach (var item in goodsArray)

{

item.Show();

}

Console.WriteLine();

Console.ForegroundColor= ConsoleColor.Red;

Array.Sort(goodsArray, new SortByPrice());

Console.WriteLine("Сортированный массив по стоимости продукции:");

foreach (var item in goodsArray)

{

item.Show();

}

Console.WriteLine();

Console.ForegroundColor = ConsoleColor.Blue;

IInit[] initArray = new IInit[10];

initArray[0] = new Toy();

initArray[0].Init();

for (int i = 1; i<5; i++)

{

initArray[i] = new Toy();

initArray[i].RandomInit();

}

initArray[5] = new Animal();

initArray[5].Init();

for (int i = 6; i<10; i++)

{

initArray[i] = new Animal();

initArray[i].RandomInit();

}

foreach (var item in initArray)

{

item.Show();

}

Console.WriteLine();

Goods goods1 = new Goods(12, 54);

Goods goods2 = new Goods(23, 120);

goods1 = (Goods)goods2.ShallowCopy();

goods2.Amount = 123;

goods1.Show();

goods2.Show();

goods1 = new Goods(12, 54);

goods2 = new Goods(23, 120);

goods1 = (Goods)goods2.Clone();

goods2.Amount = 123;

goods1.Show();

goods2.Show();

// В этом случае не играет роль будет ли поверхностное копирование или глубокое клонирование,так как в этом случае поле

// объекта не является ссылкой на объект другого класса, и происходит простое копирование значений, в ином бы случае

//Пришлось бы пользоваться глубоким клонированием , чтобы скопировать не только ссылку, а весь объект целиком внутри поля

}

static int EnterNumber() // Ввод целого числа

{

bool ok;

int Num;

do

{

ok = int.TryParse(Console.ReadLine(), out Num);

if (!ok)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Вы ввели неверное значение!");

Console.ForegroundColor = ConsoleColor.White;

}

} while (!ok);

if (Num <= 0)

{

return 0;

}

else

{

return Num;

}

}

}

}

# ПРИЛОЖЕНИЕ Б **Программа Goods.cs**

using ClassLibraryLab10;

using System;

using System.Collections.Generic;

using System.Diagnostics.CodeAnalysis;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace lab

{

public class Goods:IInit, IComparable, ICloneable

{

protected int amount = 0;

protected Random random = new Random();

public int Amount

{

get => amount;

set

{

if (amount>=0)

{

amount = value;

}

else

{

amount=0;

}

}

}

protected int price = 0;

public int Price

{

get => price;

set

{

if (price>=0)

{

price = value;

}

else

{

price=0;

}

}

}

[ExcludeFromCodeCoverage]

public virtual void Init() // Инициализация цены и количества

{

Console.WriteLine("Введите количество товара:");

Amount = EnterNumber();

Console.WriteLine("Введите цену товара:");

Price = EnterNumber();

}

[ExcludeFromCodeCoverage]

public virtual void RandomInit() // Инициализация цены и количества с помощью ДСЧ

{

Amount = random.Next(1, 101);

Price = random.Next(1, 101);

}

public Goods()

{

Amount = 0;

Price = 0;

}

public Goods (int amount,int price)

{

Amount = amount;

Price = price;

}

[ExcludeFromCodeCoverage]

public virtual void Show() //Вывод цены и количества

{

Console.WriteLine($"Количество товара: {Amount}, Стоимость товара: {Price} руб.");

}

public override bool Equals(object obj) //Перегрузка метода сравнения для тестов

{

if (obj is Goods goods)

{

return ((goods.Amount == this.Amount)&&(goods.Price == this.Price));

}

else

{

return false;

}

}

[ExcludeFromCodeCoverage]

public static int EnterNumber() // Ввод целого числа

{

bool ok;

int Num;

do

{

ok = int.TryParse(Console.ReadLine(), out Num);

if (!ok)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Вы ввели неверное значение!");

Console.ForegroundColor = ConsoleColor.White;

}

} while (!ok);

if (Num <= 0)

{

return 0;

}

else

{

return Num;

}

}

[ExcludeFromCodeCoverage]

public int CompareTo(object? obj)

{

if (!(obj is Goods)) return -1;

Goods goods = (Goods)obj;

return Amount.CompareTo(goods.Amount);

}

[ExcludeFromCodeCoverage]

public object ShallowCopy()

{

return this.MemberwiseClone();

}

public object Clone()

{

return new Goods(Amount, Price);

}

}

}

# ПРИЛОЖЕНИЕ В **Программа Product.cs**

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Diagnostics.CodeAnalysis;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab

{

public class Product: Goods

{

protected Random random = new Random();

[ExcludeFromCodeCoverage]

public override void Init()

{

base.Init();

Console.WriteLine("Введите название товара:");

Name = Console.ReadLine();

if (Name =="")

{

Name ="NoName";

}

else

{

Name = Name;

}

}

[ExcludeFromCodeCoverage]

public override void RandomInit()

{

base.RandomInit();

string[] nameArray = new string[] { "Лук Репчатый", "Jack Daniels", "Томатная паста", "Рожки", "Покровский хлеб", "Горошек зелёный", "Питахайя", "Мороженое Baskin Robbins", "Яблоки гольден","Кола" };

Name = nameArray[random.Next(nameArray.Length)];

}

public string Name { get; set; }

public Product()

: base()

{

Name ="NoName";

}

public Product(int amount, int price, string name)

: base(amount, price)

{

Name = name;

}

[ExcludeFromCodeCoverage]

public override void Show()

{

base.Show();

Console.WriteLine($"Наименование - {Name}");

}

[ExcludeFromCodeCoverage]

public new void ShowName()

{

Console.WriteLine($"Наименование - {Name}");

}

//Метод переопределения функции(overload) работает некорректно, так как в массиве будет вызываться элемент класса Goods, а не каждый отдельно

//public new void Show()

//{

// base.Show();

// Console.WriteLine($"Наименование - {Name}");

//}

public int TotalCost()

{

return Amount\*Price;

}

public override bool Equals(object obj)

{

if (obj is Product product)

{

return ((this.Price == product.Price)&&(this.Amount == product.Amount)&&(string.Compare(this.Name, product.Name)==0));

}

else

{

return false;

}

}

}

}

# ПРИЛОЖЕНИЕ Г **Программа MilkProduct.cs**

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Diagnostics.CodeAnalysis;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab

{

public class MilkProduct: Goods

{

protected Random random = new Random();

protected int fatPercent = 0;

public int FatPercent

{

get => fatPercent;

set

{

if (fatPercent>=0)

{

fatPercent=value;

}

else

{

fatPercent=0;

}

}

}

[ExcludeFromCodeCoverage]

public override void Init()

{

base.Init();

Console.WriteLine("Введите название товара:");

Name = Console.ReadLine();

if (Name =="")

{

Name ="NoName";

}

else

{

Name = Name;

}

Console.WriteLine("Введите жирность продукта:");

FatPercent = EnterFatPercent();

}

[ExcludeFromCodeCoverage]

public override void RandomInit()

{

base.RandomInit();

string[] nameArray = new string[] { "Сырок Б. Ю. Александров","Молоко из Нытвы","Талицкий йогурт","Простокваша","Сметана","Молочный коктейль Чудо","Кумыс","Айран","Творог","Биолайф"};

Name = nameArray[random.Next(nameArray.Length)];

FatPercent = random.Next(0,26);

}

public string Name { get; set; }

public MilkProduct()

: base()

{

Name = "NoName";

}

public MilkProduct(int amount, int price, string name, int fatPercent)

: base(amount, price)

{

Name = name;

FatPercent = fatPercent;

}

[ExcludeFromCodeCoverage]

public override void Show()// Вывод количество товара, его цены и имя

{

base.Show();

Console.WriteLine($"Наименование - {Name}");

Console.WriteLine($"Жирность продукта - {FatPercent}%");

}

[ExcludeFromCodeCoverage]

public new void ShowName() //Вывод наименования товара

{

Console.WriteLine($"Наименование - {Name}");

}

//Метод переопределения функции(overload) работает некорректно, так как в массиве будет вызываться элемент класса Goods, а не каждый отдельно

//public new void Show()

//{

// base.Show();

// Console.WriteLine($"Наименование - {Name}");

//}

public int TotalCost() // Нахождение общей стоимости

{

return Amount\*Price;

}

public override bool Equals(object obj) //Перегрузка метода сравнения

{

if (obj is MilkProduct milkProduct)

{

return ((this.Price == milkProduct.Price)&&(this.Amount == milkProduct.Amount)&&(string.Compare(this.Name, milkProduct.Name)==0));

}

else

{

return false;

}

}

[ExcludeFromCodeCoverage]

public static int EnterFatPercent() // Ввод целого числа

{

bool ok;

int Num;

do

{

ok = int.TryParse(Console.ReadLine(), out Num);

if ((!ok)||(Num>100))

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Вы ввели неверное значение!");

Console.ForegroundColor = ConsoleColor.White;

}

} while (!(ok&& (Num<=100)));

if (Num <= 0)

{

return 0;

}

else

{

return Num;

}

}

}

}

# ПРИЛОЖЕНИЕ Д **Программа Toy.cs**

using System;

using System.Collections.Generic;

using System.Diagnostics.CodeAnalysis;

using System.Linq;

using System.Security.Cryptography.X509Certificates;

using System.Text;

using System.Threading.Tasks;

namespace lab

{

public class Toy:Goods

{

public string Name;

protected Random random = new Random();

[ExcludeFromCodeCoverage]

public override void Init()

{

base.Init();

Console.WriteLine("Введите название товара:");

Name = Console.ReadLine();

if (Name =="")

{

Name ="NoName";

}

else

{

Name = Name;

}

Console.WriteLine("Введите тип игрушки:");

ToyType = Console.ReadLine();

if ( ToyType== "")

{

ToyType = "NoType";

}

else

{

ToyType = ToyType;

}

}

[ExcludeFromCodeCoverage]

public override void RandomInit()

{

base.RandomInit();

string[] nameArray = new string[] { "Хагги Вагги", "Амонг Ус", "Бравл Старс","Акула из икеи","Капибара","Денис Малинин(кукла вуду)","Утка-обнимашка","Базз Лайтер","Спиннер","Поп-Ит" };

Name = nameArray[random.Next(nameArray.Length)];

base.RandomInit();

string[] typeArray = new string[] { "Мягкая игрушка", "Конструктор", "Головоломка", "Пазлы", "Модель" };

ToyType = typeArray[random.Next(typeArray.Length)];

}

public string ToyType { get;set; }

public Toy()

: base()

{

Name = "NoName";

ToyType = "NoType";

}

public Toy(int amount, int price, string name, string toyType)

:base (amount, price)

{

Name = name;

ToyType = toyType;

}

[ExcludeFromCodeCoverage]

public override void Show()

{

base.Show();

Console.WriteLine($"Наименование - {Name}");

Console.WriteLine($"Тип игрушки - {ToyType}");

}

public new void ShowName()

{

Console.WriteLine($"Наименование - {Name}");

}

//Метод переопределения функции(overload) работает некорректно, так как в массиве будет вызываться элемент класса Goods, а не каждый отдельно

//public new void Show()

//{

// base.Show();

// Console.WriteLine($"Наименование - {Name}");

//}

public int TotalCost()

{

return Amount\*Price;

}

public override bool Equals(object obj)

{

if (obj is Toy toy)

{

return ((this.Price == toy.Price)&&(this.Amount == toy.Amount)&&(string.Compare(this.Name, toy.Name)==0));

}

else

{

return false;

}

}

}

}

# ПРИЛОЖЕНИЕ Е **Программа Animal.cs**

using lab;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Diagnostics.CodeAnalysis;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibraryLab10

{

public class Animal:IInit

{

static Random random = new Random();

static string[] Names = { "Лиса", "Слон", "Лев", "Капибара", "Аксолотль", "Мурена", "Крокодил" };

public string Name { get; set; }

int age;

public int Age

{

get => age;

set

{

if (age>=0 && age<=50)

{

age=value;

}

else

{

age=0;

}

}

}

public Animal()

{

Age = 1;

Name = "NoName";

}

public Animal(string name, int age)

{

Name=name;

Age=age;

}

[ExcludeFromCodeCoverage]

public void Show()

{

Console.WriteLine($"Название животного: {Name}, Возраст: {Age}");

}

[ExcludeFromCodeCoverage]

public virtual void Init()

{

Console.WriteLine("Введите название животного:");

Name = Console.ReadLine();

Console.WriteLine("Введите возраст:");

Age = EnterAge();

}

[ExcludeFromCodeCoverage]

public virtual void RandomInit()

{

Name = Names[random.Next(Names.Length)];

Age = random.Next(1, 50);

}

[ExcludeFromCodeCoverage]

public static int EnterAge()

{

int Number;

bool ok;

do

{

ok = ((int.TryParse(Console.ReadLine(), out Number))&&(Number >0)&&(Number <51));

if (!ok)

{

Console.WriteLine("Вы ввели неверное значение!");

}

} while (!ok);

return Number;

}

public override bool Equals(object obj)

{

if (obj is Animal animal)

{

return ((this.Age == animal.Age)&&(string.Compare(this.Name, animal.Name)==0));

}

else

{

return false;

}

}

}

}

# ПРИЛОЖЕНИЕ Ё **Программа SortByPrice.cs**

using lab;

using System;

using System.Collections;

using System.Diagnostics.CodeAnalysis;

using System.Linq;

using System.Runtime.InteropServices.ObjectiveC;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibraryLab10

{

[ExcludeFromCodeCoverage]

public class SortByPrice:IComparer

{

public int Compare(object x, Object y)

{

Goods goods1 = x as Goods;

Goods goods2 = y as Goods;

if (goods1.Price<goods2.Price)

{

return -1;

}

else

{

if (goods2.Price==goods1.Price)

{

return 0;

}

else

{

return 1;

}

}

}

}

}

# ПРИЛОЖЕНИЕ Ж **Программа IInit.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography.X509Certificates;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibraryLab10

{

public interface IInit

{

void RandomInit();

void Init();

void Show();

}

}