Министерство образования и науки РФ

Институт информационных технологий и управления в технических системах

Севастопольский государственный университет

Лабораторная работа №2

по дисциплине: «Платформа .Net»

Выполнил:

ст.гр. ИСб/22о

Воронин И.Ю.

Проверил:

Шишкевич В.Е.

г. Севастополь

2016 г.

1. Цель работы

Познакомиться с пользовательскими типами данных в языке С#: структура и перечисление;

Ознакомиться со структурой класса, его созданием и использованием, описанием членов класса: полей, свойств, инициализации объектов класса с помощью конструкторов;

Изучить механизм создания иерархий классов в С# и применение интерфейсов при наследовании;

Изучить механизм генерации и обработки исключений.

1. Вариант задания

1. Для заданной структуры данных разработать абстрактный класс и класс-наследник. В классе реализовать несколько конструкторов. Создать методы, работающие с полями класса. Часть из них должны быть виртуальными. Добавить методы-свойства и индексаторы.

2. Разработать интерфейсные классы, добавляющие некоторые методы в класс-потомок. Изучить причины возникновения коллизий имен при наследовании и способы их устранения.

3. Разработать классы исключительных ситуаций и применить их для обработки, возникающих исключений.

4. Написать демонстрационную программу.

Вариант 2.

ИЗДЕЛИЕ: название, шифр, количество, комплектация.

1. Код программы

Интерфейс IDeliver:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace sharp\_lab2

{

public interface IDeliver

{

//доставка на дом

void DeliverToHome(Product product);

//доставка в точку самовывоза

void DeliverToOurOffice(Product product);

//Отменить заказ

void DeliverCancel();

//состояние доставки

void DeliverState();

}

}

Класс IDeliverImpl:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace sharp\_lab2

{

public class IDeliverImpl:IDeliver

{

String date;

public IDeliverImpl()

{

date = "Days: ";

}

//доставка на дом

public void DeliverToHome(Product product)

{

if (product.getOrder() == Order.avaible)

{

if (product.getCode() % 3 == 1)

{

date += product.getCode() % 6;

}

else

{

date += product.getCode() % 3;

}

}

}

//доставка в точку самовывоза

public void DeliverToOurOffice(Product product)

{

if (product.getOrder() == Order.avaible)

{

if (product.getCode() % 4 == 2)

{

date += product.getCode() % 2 + 1;

}

else

{

date += product.getCode() % 4 +2;

}

}

}

//Отменить заказ

public void DeliverCancel(){

date = "Deliver was canceled.";

Console.WriteLine(date);

}

//состояние доставки

public void DeliverState()

{

Console.WriteLine(date);

}

}

}

Класс Product:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace sharp\_lab2

{

//состояния заказа

public enum Order

{

avaible,

waiting,

not\_avaible

}

public abstract class Product

{

//наименование

protected String name;

//шифр продукта

protected int code;

//количество

protected int amount;

//комплектация

protected String package;

//заказан ли товари или нет

private Order order;

public int getCode()

{

return code;

}

public virtual void getInfo(){

Console.WriteLine("\nName: " + name);

Console.WriteLine("Code: " + code);

Console.WriteLine("Amount: " + amount);

Console.WriteLine("Package: " + package);

}

public virtual void MakeOrder(){

switch (order)

{

case Order.avaible: {

if (amount > 0)

{

Console.WriteLine("You can buy it and bought already.");

amount--;

}

if (amount == 0)

{

order = Order.not\_avaible;

amount = 0;

}

break;

}

case Order.waiting: Console.WriteLine("You should wait."); break;

case Order.not\_avaible: Console.WriteLine("You cant buy."); break;

}

}

public Order getOrder()

{

switch (order)

{

case Order.avaible: Console.WriteLine("Avaible"); return Order.avaible; break;

case Order.waiting: Console.WriteLine("Waiting"); return Order.waiting; break;

case Order.not\_avaible: Console.WriteLine("No avaible"); return Order.not\_avaible; break;

}

return Order.not\_avaible;

}

public void setOrder(Order order)

{

this.order = order;

}

public virtual void setAll()

{

Console.Write("Name: ");

name = Console.ReadLine();

Console.Write("Code: ");

code = Convert.ToInt32(Console.ReadLine());

Console.Write("Amount: ");

amount = Convert.ToInt32(Console.ReadLine());

Console.Write("Package: ");

package = Console.ReadLine();

}

}

}

Класс Smartphone:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace sharp\_lab2

{

class Smartphone: Product

{

public Smartphone(String name, int code , int amount, String package):base()

{

this.name = name;

this.code = code;

this.amount = amount;

this.package = package;

if (amount > 0) this.setOrder(Order.avaible);

if (amount == 0) this.setOrder(Order.not\_avaible);

if (amount < 0) this.setOrder(Order.waiting);

}

public Smartphone(String name):base()

{

this.name = name;

code = amount = 0;

this.package = "";

this.setOrder(Order.not\_avaible);

}

public Smartphone():base() { }

public override void setAll()

{

base.setAll();

if (amount > 0) this.setOrder(Order.avaible);

if (amount == 0) this.setOrder(Order.not\_avaible);

if (amount < 0) this.setOrder(Order.waiting);

}

}

}

Класс Program:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace sharp\_lab2

{

//состояния заказа

public enum Order

{

avaible,

waiting,

not\_avaible

}

public abstract class Product

{

//наименование

protected String name;

//шифр продукта

protected int code;

//количество

protected int amount;

//комплектация

protected String package;

//заказан ли товари или нет

private Order order;

public int getCode()

{

return code;

}

public virtual void getInfo(){

Console.WriteLine("\nName: " + name);

Console.WriteLine("Code: " + code);

Console.WriteLine("Amount: " + amount);

Console.WriteLine("Package: " + package);

}

public virtual void MakeOrder(){

switch (order)

{

case Order.avaible: {

if (amount > 0)

{

Console.WriteLine("You can buy it and bought already.");

amount--;

}

if (amount == 0)

{

order = Order.not\_avaible;

amount = 0;

}

break;

}

case Order.waiting: Console.WriteLine("You should wait."); break;

case Order.not\_avaible: Console.WriteLine("You cant buy."); break;

}

}

public Order getOrder()

{

switch (order)

{

case Order.avaible: Console.WriteLine("Avaible"); return Order.avaible; break;

case Order.waiting: Console.WriteLine("Waiting"); return Order.waiting; break;

case Order.not\_avaible: Console.WriteLine("No avaible"); return Order.not\_avaible; break;

}

return Order.not\_avaible;

}

public void setOrder(Order order)

{

this.order = order;

}

public virtual void setAll()

{

try

{

Console.Write("Name: ");

name = Console.ReadLine();

Console.Write("Code: ");

code = Convert.ToInt32(Console.ReadLine());

Console.Write("Amount: ");

amount = Convert.ToInt32(Console.ReadLine());

Console.Write("Package: ");

package = Console.ReadLine();

}

catch(Exception){

Console.WriteLine("\nExeption. Retry enter data again\n");

setAll();

}

}

}

}

1. Выполнение программы

Работа программы описана в классе Programm. В данном случае мы создаём объект смартфон с нужными нам атрибутами, после нужное число раз запрашиваем данные о телефоне, делаем заказ и проверяем, возможно ли заказать телефон. Далее при помощи интерфейса IDeliver мы оформляем доставку на дом, проверяем её состояние, отменяем и снова проверяем состояние. При запуске данной программы на экран выводится следующий диалог. (рис 4.1)

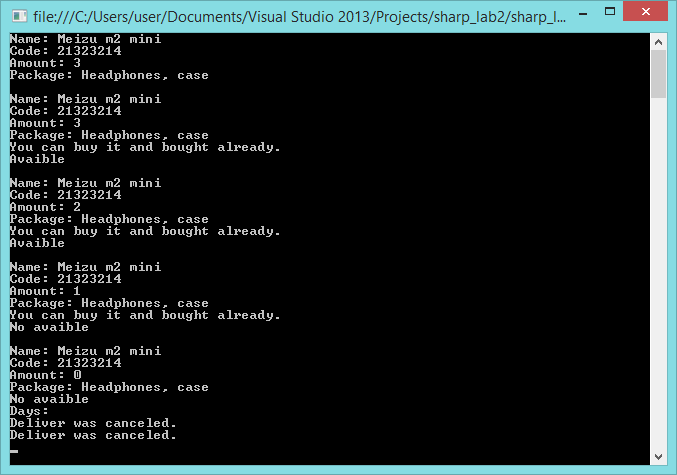


Рисунок 4.1 – Тест программы.

ВЫВОДЫ

Была написана программа на языке С# в среде программирования Microsoft Visual Studio 2013. Были рассмотрены возможности языка как объектно-ориентированного. Были использованы интерфейсы, наследование классов, описание исключительных ситуаций.