**Лабораторная работа № 3**

**Широков Ф. А. з22928/2**

**Тема:** Создание проектов с визуальными элементами пользовательского   
интерфейса: списками, редакторами.

**Вариант 10**: разработать программу ввода названий товаров, количества товаров на складе, их закупочной стоимости и цены реализации, добавления этих сведений в поле многострочного редактора и расчёта прибыли от продажи перечисленных товаров.

**Цель работы:** получить практические навыки создания элементов пользовательского интерфейса со списками, редакторами, и разработки методов, реализующих работу с такими элементами интерфейса.

**Ход работы:**

1. Прочитайте свой вариант задания (см. ниже). Запишите в отчёт, что дано и что надо найти, с указанием типов исходных данных и результатов.
2. Разработайте описание класса в соответствие с условием задачи вашего варианта. Включите в описание класса следующие методы: конструктор, перегрузку метода ToString() для формирования строки для печати данных экземпляра класса, свойство для чтения значения скрытого поля, необходимого для расчёта.
3. Продумайте, какие элементы интерфейса необходимо разместить на форме, чтобы обеспечить удобную и наглядную работу пользователя. Интерфейс должен включать список listBox или comboBox и многострочный редактор richTextBox.
4. Настройте элементы интерфейса. Скопируйте в отчёт скриншот формы на этапе конструирования и надпишите компоненты с указанием их типов.

**Ход выполнения:**

1. Так как задание предполагает хранение товара и проведение над ним неких операций, создадим два класса: Storage – класс, описывающий виртуальный склад и класс Product – описывающий товар, который хранится на складе и над которым будут производить какие-то действия.

internal class Storage

{

private List<Product> \_products;

public Storage()

{

\_products = new List<Product>();

}

public int GetCapacity()

{

return \_products.Count;

}

public void AddProduct(Product product)

{

\_products.Add(product);

}

public Product GetProduct(int index)

{

if (index > 0 & index < \_products.Count())

{

return \_products[index];

}

return null;

}

}

internal class Product

{

public enum ProductType

{

Food,

Hardware,

Medical\_s,

Tool,

Unknown

}

private String \_productName;

private ProductType \_productType;

private int \_amount;

private double \_buyingPrice;

private double \_sellinglPrice;

public Product(String Name, String Type,int Amount, double BuyingPrice, double SellingPrice)

{

this.\_productName = Name;

this.\_productType = (ProductType)Enum.Parse(typeof(ProductType), Type);

this.\_buyingPrice = BuyingPrice;

this.\_sellinglPrice = SellingPrice;

\_amount += Amount;

}

public void SetProductType(String Type)

{

this.\_productType = (ProductType)Enum.Parse(typeof(ProductType), Type);

}

public void SetName(String Name)

{

this.\_productName = Name;

}

public void SetBuyingPrice(double BuyingPrice)

{

this.\_buyingPrice = BuyingPrice;

}

public void SetSellingPrice(double SellingPrice)

{

this.\_sellinglPrice = SellingPrice;

}

public String GetProductName()

{

return this.\_productName;

}

public ProductType GetProductType()

{

return this.\_productType;

}

public double GetBuyingPrice()

{

return this.\_buyingPrice;

}

public double GetSellingPrice()

{

return this.\_sellinglPrice;

}

public int GetAmount()

{

return \_amount;

}

public double GetProfit()

{

return (this.\_sellinglPrice - this.\_buyingPrice) \* this.\_amount;

}

public override string? ToString()

{

return this.\_productName +"\t" + " | "+ "\t" + this.\_productType + "\t" + " | " + "\t" + this.GetBuyingPrice() +

" | " + "\t" + this.GetSellingPrice() + " | " + "\t" + this.GetProfit();

}

}

2. Добавим форму, которая будет интерфейсом для работы с продуктами. С помощью нее мы будем добавлять продукты на склад.

public partial class Form1 : Form

{

Storage storage;

Product product;

public Form1()

{

InitializeComponent();

storage = new Storage();

}

private void button1\_Click\_1(object sender, EventArgs e)

{

String name = NameProduxtTextBox.Text;

String productType = ProductTypecomboBox.Text;

int count = Int32.Parse(AmountProducttextBox.Text);

double BuyingPrice = Double.Parse(BuyingPriceProducttextBox.Text);

double SellingPrice = Double.Parse(SellingPriceProducttextBox.Text);

product = new Product(name, productType, count, BuyingPrice, SellingPrice);

storage.AddProduct(product);

richTextBox1.Text += product.ToString() + "\n";

NameProduxtTextBox.Clear();

AmountProducttextBox.Clear();

BuyingPriceProducttextBox.Clear();

SellingPriceProducttextBox.Clear();

}

private void AmountProducttextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))

{

e.Handled = true;

}

}

private void BuyingPriceProducttextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) &&

(e.KeyChar != '.'))

{

e.Handled = true;

}

// only allow one decimal point

if ((e.KeyChar == '.') && ((sender as TextBox).Text.IndexOf('.') > -1))

{

e.Handled = true;

}

}

private void SellingPriceProducttextBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) &&

(e.KeyChar != '.'))

{

e.Handled = true;

}

// only allow one decimal point

if ((e.KeyChar == '.') && ((sender as TextBox).Text.IndexOf('.') > -1))

{

e.Handled = true;

}

}

private void button2\_Click(object sender, EventArgs e)

{

NameProduxtTextBox.Clear();

AmountProducttextBox.Clear();

BuyingPriceProducttextBox.Clear();

SellingPriceProducttextBox.Clear();

}

private void Form1\_Leave(object sender, EventArgs e)

{

if (

string.IsNullOrEmpty(NameProduxtTextBox.Text)

||

string.IsNullOrEmpty(AmountProducttextBox.Text)

||

string.IsNullOrEmpty(BuyingPriceProducttextBox.Text)

||

string.IsNullOrEmpty(SellingPriceProducttextBox.Text)

)

{

button1.Enabled = false;

}

else

{

button1.Enabled = true;

}

}

private void richTextBox1\_TextChanged(object sender, EventArgs e)

{

if (

string.IsNullOrEmpty(NameProduxtTextBox.Text)

||

string.IsNullOrEmpty(AmountProducttextBox.Text)

||

string.IsNullOrEmpty(BuyingPriceProducttextBox.Text)

||

string.IsNullOrEmpty(SellingPriceProducttextBox.Text)

)

{

button1.Enabled = false;

}

else

{

button1.Enabled = true;

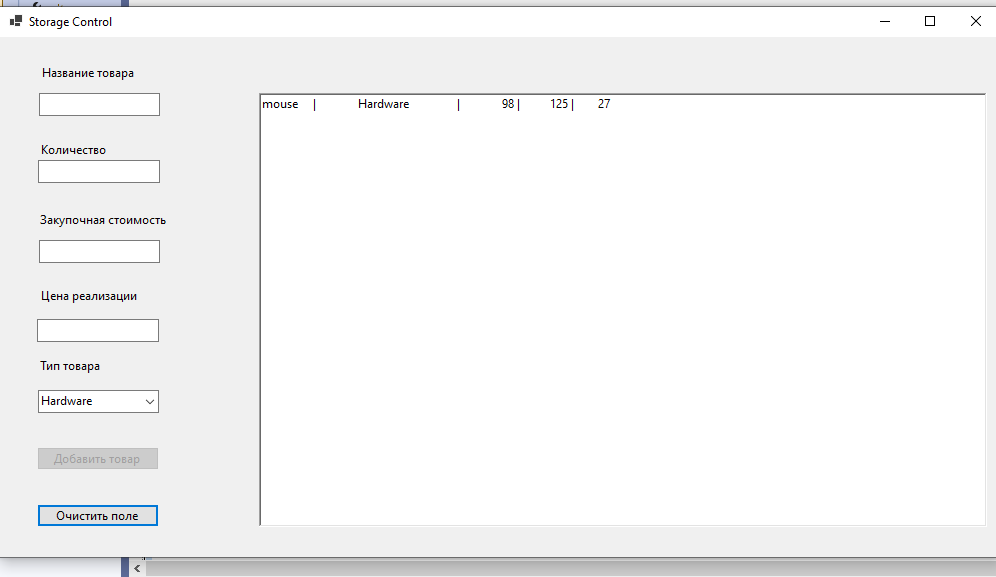
}

}

}

3. Далее с помощью кнопки «Добавить товар» мы можем добавить товар на наш склад, в поле отображения товара мы увидим все информацию о нем, а также прибыль от его продажи.





4. Если нажать на кнопку «Очистить поле» то мы можем очистить заполненные поля товара, и напечатать информацию в них заново.