

**Московский государственный технический
университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5. Группа 31

Отчёт по работе по моделям данных
«Программа-симулятор торгового автомата на с#»



Авторы
Анна Зенгер
Сергей Кочкин
Максим Ромашёв
Родион Афанасьев

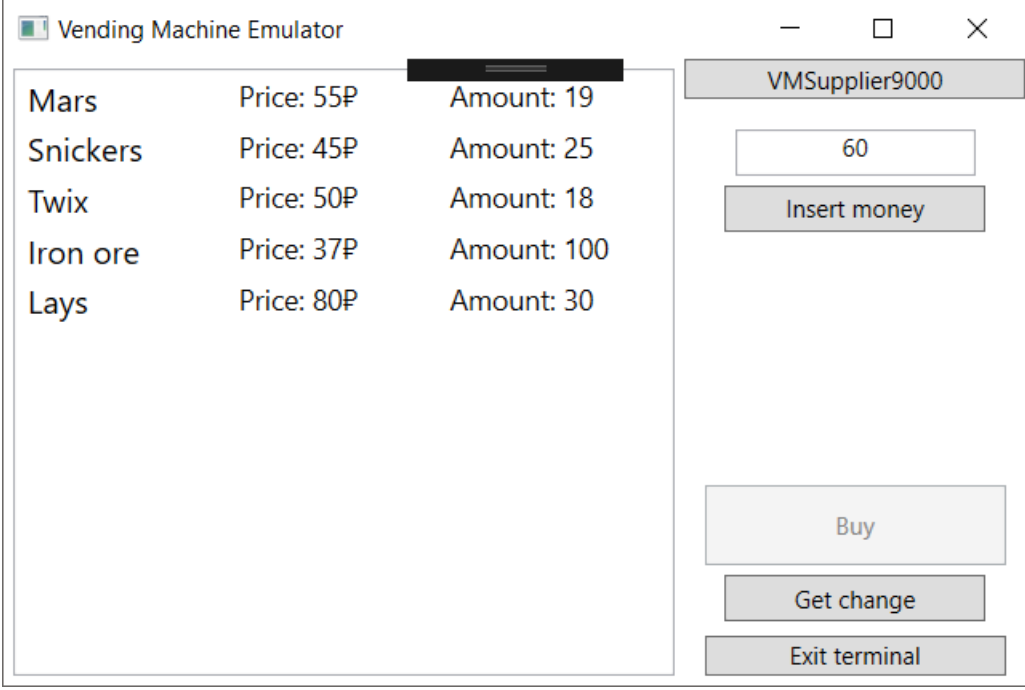
г. Москва, 2017 г.

Цель работы: создание виртуальной модели исполняемого устройства, которое бы послужило бы основой для реальной физической модели, изучение особенностей языка с#, получение познаний в среде интернет технологий.

Программа представляет из себя симулятор реально существующего аппарата по продаже продуктов питания. Проект состоит из двух частей: пользовательского интерфейса и всего, что с этим связано и логики, описываемой в классах.

Примеры интерфейса программы:

Основное окно



The screenshot shows a window titled "Vending Machine Emulator". On the left, there is a table of products with their prices and amounts. On the right, there are several buttons for interacting with the machine.

Product	Price	Amount
Mars	55P	19
Snickers	45P	25
Twix	50P	18
Iron ore	37P	100
Lays	80P	30

Buttons on the right side of the window:

- VMSupplier9000
- 60 (input field)
- Insert money
- Buy
- Get change
- Exit terminal

Окно добавление продуктов

The 'Supplier9000' window is titled 'Supplies' and contains a list of products: Mars, Snickers, Twix (selected), Iron ore, and Lays. To the right, under 'Banknotes'n'Coins', is a list of denominations: 1000, 500, 100, 50, 10, 5, 2, and 1. In the center, there are two input fields: 'Amount:' with the value '18' and 'Price (for products):' with the value '50'. Below these are three buttons: 'Task A', 'Add new local product' (highlighted with a dashed border), and 'Delete'.

Supplies	Amount:	Price (for products):	Banknotes'n'Coins
Mars	18	50	1000
Snickers			500
Twix			100
Iron ore			50
Lays			10
			5
			2
			1

Buttons: Task A, Add new local product, Delete

Ввод денег и получение сдачи

The 'A...' window features a display showing '60' and a vertical stack of buttons for inserting money: 'Insert 1P', 'Insert 2P', 'Insert 5P', 'Insert 10P', 'Insert 50P', 'Insert 100P', 'Insert 500P', and 'Insert 1000P'. To the right, a smaller dialog box displays the message 'Change is given: 10Px2 5Px1 2Px29' and an 'OK' button.

Buttons: Insert 1P, Insert 2P, Insert 5P, Insert 10P, Insert 50P, Insert 100P, Insert 500P, Insert 1000P

Change is given: 10Px2 5Px1 2Px29

OK

Данные программы хранятся в json формате для дальнейшего взаимодействия с пользователем между сессиями.

Далее приводятся куски основного кода. Весь код доступен в Github репозитории по QR-коду выше.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Data.Entity;
using System.IO;

namespace Vending_Terminal_Software_Classes
{
    public class Vending_Machine_Emulator
    {
        public string TerminalLocation;

        public Context context = new Context();

        int BiggestCoin;

        public DataBase Data = new DataBase();

        List<Banknotes_n_Coins> Changelist = new List<Banknotes_n_Coins>(); string
ChangeString;

        public CurrentProduct pCopy { get; set; }

        public event Action MainRefresh;

        public event Action SupplierRefresh;

        public event Action<string> MessageBox;

        public Vending_Machine_Emulator()
        {

        }

        public void Buy()
        {
            context.CurrentStates.First(n => n.Location == TerminalLocation).Money -=
pCopy.Price;

            var Product = context.CurrentStates
                .Include(cs => cs.Product)
                .First(n => n.Location == TerminalLocation)
                .Product.Find(n => n.Code == context.Info
                    .First(m => m.Name == pCopy.Name).Code);

            Product.Amount -= 1;

            context.CurrentStates
                .Include(cs => cs.Sales)
                .First(n => n.Location == TerminalLocation)
                .Sales.Add(new Sale { Code = Product.Code, Date = DateTime.Now });

            context.SaveChanges();

            MR();
        }
    }
}
```

```

public void Change() // Usual change method
{
    ChangeList.Clear();

    if (GiveFive() == true)
    {
        ComplicatedChange();
    }

    while (context.CurrentStates.First(n => n.Location == TerminalLocation).Money
> 0)
    {
        var Coin = context.CurrentStates
            .Include(cs => cs.Banknotes_n_Coins)
            .First(n => n.Location == TerminalLocation)
            .Banknotes_n_Coins.Find(n => n.Cost <= context.CurrentStates
            .First(m => m.Location == TerminalLocation).Money & n.Amount > 0 &
n.CanBeChange == true);

        Coin.Amount -= 1;
        context.CurrentStates.First(n => n.Location == TerminalLocation).Money -=
Coin.Cost;

        // Given coins counter

        if (ChangeList.Exists(n => n.Cost == Coin.Cost))
        {
            var tmp = ChangeList.Find(n => n.Cost == Coin.Cost);
            tmp.Amount += 1;
        }
        else
        {
            ChangeList.Add(new Banknotes_n_Coins { Cost = Coin.Cost, Amount = 1
});
        }
    }

    context.SaveChanges();

    // ChangeString creator

    ChangeList.Sort(new MoneyComparer());

    ChangeString = "Change is given:";

    ChangeList.ForEach(delegate (Banknotes_n_Coins n) { ChangeString += " " +
n.Cost + "P" + n.Amount; });

    MR();

    MessageBox?.Invoke(ChangeString);
}

public void ComplicatedChange() // This change method gives only 1 5P coin
{
    context.CurrentStates.First(n => n.Location == TerminalLocation).Money -= 5;

    var Coin = context.CurrentStates.Include(cs => cs.Banknotes_n_Coins).First(n
=> n.Location == TerminalLocation).Banknotes_n_Coins.Find(n => n.Cost == 5);
    Coin.Amount -= 1;

    ChangeList.Add(new Banknotes_n_Coins { Cost = 5, Amount = 1 });

    while (context.CurrentStates.First(n => n.Location == TerminalLocation).Money
> 0)

```

```

        {
            Coin = context.CurrentStates.Include(cs => cs.Banknotes_n_Coins).First(n
=> n.Location == TerminalLocation).Banknotes_n_Coins.Find(n => n.Cost <=
context.CurrentStates.First(m => m.Location == TerminalLocation).Money & n.Amount > 0 &
n.CanBeChange == true & n.Cost != 5);

            Coin.Amount -= 1;
            context.CurrentStates.First(n => n.Location == TerminalLocation).Money -=
Coin.Cost;

            // Given coins counter

            if (ChangeList.Exists(n => n.Cost == Coin.Cost))
            {
                var tmp = ChangeList.Find(n => n.Cost == Coin.Cost);
                tmp.Amount += 1;
            }
            else
            {
                ChangeList.Add(new Banknotes_n_Coins { Cost = Coin.Cost, Amount = 1
});
            }
        }

        public void MoneyAdd(int value)
        {
            var Coin = context.CurrentStates.Include(cs => cs.Banknotes_n_Coins).First(n
=> n.Location == TerminalLocation).Banknotes_n_Coins.Find(n => n.Cost == value);

            Coin.Amount += 1;
            context.CurrentStates.First(n => n.Location == TerminalLocation).Money +=
value;

            context.SaveChanges();

            MR();
        }

        public bool ChangeCount(int tag)
        {
            BiggestCoin = context.CurrentStates
                .Include(cs => cs.Banknotes_n_Coins)
                .First(n => n.Location == TerminalLocation)
                .Banknotes_n_Coins.Find(n => n.CanBeChange == true)
                .Cost;

            if (tag <= BiggestCoin)
                return true;

            int tmp = 0;

            using (var context = new Context())
            {
                var Change = context.CurrentStates.Include(cs =>
cs.Banknotes_n_Coins).First(n => n.Location ==
TerminalLocation).Banknotes_n_Coins.FindAll(n => n.CanBeChange == true);

                foreach (Banknotes_n_Coins B in Change)
                {
                    tmp += B.Cost * B.Amount;
                }

                if (tag + context.CurrentStates.First(n => n.Location ==
TerminalLocation).Money <= tmp)

```

```

        return true;
    else
        return false;
    }
}

public bool GiveFive() // Check amount of 1P coins and if change is odd
{
    if (context.CurrentStates.Include(cs => cs.Banknotes_n_Coins).First(n =>
n.Location == TerminalLocation).Banknotes_n_Coins.Find(n => n.Cost == 1).Amount == 0 &
context.CurrentStates.First(n => n.Location == TerminalLocation).Money % 2 == 1)
        return true;
    else
        return false;
}

public bool CanBeBought()
{
    try
    {
        if (pCopy.Price <= context.CurrentStates.First(n => n.Location ==
TerminalLocation).Money & pCopy.Amount > 0)
        {
            if (context.CurrentStates.Include(cs => cs.Banknotes_n_Coins).First(n
=> n.Location == TerminalLocation).Banknotes_n_Coins.Find(n => n.Cost == 1).Amount == 0 &
context.CurrentStates.First(n => n.ID == 1).Money - pCopy.Price < 5)
                return false;
            return true;
        }
        return false;
    }
    catch { return false; }
}

public void SR()
{
    SupplierRefresh?.Invoke();
}

public void MR()
{
    MainRefresh?.Invoke();
}
}
}

```