

*САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ*

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №0

по курсу «ООП»

Тема: Торговый автомат

Выполнила:

Федорова М. В.

K3239

Проверил:

Санкт-Петербург

2024 г.

1. Общее описание проекта

Данный проект представляет собой консольное приложение, имитирующее работу вендингового автомата. Приложение позволяет пользователю взаимодействовать с автоматом для приобретения товаров и предоставляет административный режим для управления ассортиментом и денежными средствами.

****Функциональные возможности пользователя:****

- Просмотр списка доступных товаров с их ценами и количеством
- Внесение монет различных номиналов (1, 0.5, 0.2, 0.1 рубля)
- Выбор товара и его получение, при условии достаточной внесенной суммы
- Получение сдачи (если необходимо) и возврат неиспользованных монет при отмене операции

****Административный режим:****

- Пополнение ассортимента товаров
- Загрузка монет в автомат
- Сбор накопленных денежных средств из автомата

2. Структура проекта и описание кода

2.1. Program.cs – Точка входа в приложение

Основной файл приложения, который инициализирует торговый автомат и обрабатывает пользовательский ввод.

****Инициализация конфигурации:****

```
var configuration = new ConfigurationBuilder()  
    .SetBasePath(Directory.GetCurrentDirectory())  
    .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)  
    .AddJsonFile("products.json", optional: false, reloadOnChange: true)  
    .Build();
```

Создается объект конфигурации, который загружает настройки из JSON файлов. appsettings.json содержит пароль администратора, а products.json - данные о товарах.

****Загрузка списка товаров:****

```
var products = new List<Product>();  
var productsSection = configuration.GetSection("Products");  
foreach (var productSection in productsSection.GetChildren())  
{  
    var id = int.Parse(productSection["Id"] ?? "0");  
    var name = productSection["Name"] ?? "";  
    var priceKop = int.Parse(productSection["PriceKop"] ?? "0");  
    var quantity = int.Parse(productSection["Quantity"] ?? "0");  
    products.Add(new Product(id, name, priceKop, quantity));  
}
```

Код читает секцию "Products" из конфигурации и создает объекты Product для каждого товара, парся значения ID, названия, цены в копейках и количества.

****Создание кошелька автомата:****

```
var machineWallet = new Wallet(new Dictionary<int, int>
{
    [100] = 10,
    [50] = 10,
    [20] = 10,
    [10] = 10
});
```

Создается кошелек автомата с начальным количеством монет каждого номинала (100, 50, 20, 10 копеек).

****Основной цикл работы приложения:****

```
while (true)
{
    Console.WriteLine();
    Console.WriteLine($"Balance: {Money.Format(vm.CurrentBalanceKop)}");
    Console.WriteLine("1) Show products");
    Console.WriteLine("2) Insert coin (1, 0.5, 0.2, 0.1 RUB)");
    Console.WriteLine("3) Buy product");
    Console.WriteLine("4) Cancel / return coins");
    Console.WriteLine("5) Admin mode");
    Console.WriteLine("0) Exit");
    Console.Write("Choose: ");
```

Бесконечный цикл, который отображает меню пользователя и обрабатывает выбор через switch конструкцию.

2.2. Класс VendingMachine (machine.cs)

Основной класс торгового автомата, управляющий всеми операциями.

****Основные поля****

```
private readonly List<Product> _products;  
private readonly Wallet _machineWallet;  
private readonly Wallet _sessionWallet;  
private readonly string _adminPassword;
```

* `_products` - список товаров в автомате

* `_machineWallet` - кошелек автомата с монетами

* `_sessionWallet` - временный кошелек для текущей сессии пользователя

* `_adminPassword` - пароль для доступа к административным функциям

****Конструктор****

```
public VendingMachine(List<Product> products, Wallet machineWallet, string adminPassword)  
{  
    _products = products;  
    _machineWallet = machineWallet;  
    _sessionWallet = new Wallet();  
    _adminPassword = adminPassword;  
}
```

Инициализирует автомат с переданными товарами, кошельком и паролем.
Создает новый пустой кошелек для сессии.

****Метод InsertCoin** - внесение монеты**

```
public bool InsertCoin(int denomKop)  
{  
    if (!Money.IsSupportedCoin(denomKop)) return false;  
    _sessionWallet.Add(denomKop);  
    return true;  
}
```

Проверяет, поддерживается ли номинал монеты, и если да - добавляет ее в сессионный кошелек

****Метод Purchase**- покупка товара**

```
{
    var product = _products.FirstOrDefault(p => p.Id == productId);
    if (product == null) return (false, "Product not found");
    if (!product.HasStock) return (false, "Out of stock");
    if (CurrentBalanceKop < product.PriceKop) return (false, "Insufficient balance");

    foreach (var kv in _sessionWallet.Snapshot())
        _machineWallet.Add(kv.Key, kv.Value);

    var changeKop = CurrentBalanceKop - product.PriceKop;
    if (!Calc.TryMakeChange(changeKop, _machineWallet, out var change))
    {
        foreach (var kv in _sessionWallet.Snapshot())
            _machineWallet.TryRemove(kv.Key, kv.Value);
        return (false, "Cannot make change for this transaction");
    }

    product.ConsumeOne();

    foreach (var kv in change)
        _machineWallet.TryRemove(kv.Key, kv.Value);

    ClearSession();

    var msg = changeKop == 0 ? "Enjoy your product!" :
        $"Enjoy your product! Change: {DescribeCoins(change)} ({Money.Format(changeKop)})";
    return (true, msg);
}
```

1. Находит товар по ID
2. Проверяет наличие товара и достаточность средств
3. Перемещает монеты из сессионного кошелька в основной
4. Рассчитывает сдачу
5. Если сдачу выдать нельзя - возвращает монеты
6. Уменьшает количество товара на 1
7. Выдает сдачу из основного кошелька
8. Очищает сессию

****Метод Cancel**- отмена операции**

```
public (bool ok, string message) Cancel()
{
    var coins = _sessionWallet.Snapshot().Where(kv => kv.Value > 0)
        .ToDictionary(k => k.Key, v => v.Value);
    var sum = CurrentBalanceKop;
    ClearSession();
    var msg = coins.Count == 0 ? "Nothing to return" :
        $"Returned: {DescribeCoins(coins)} ({Money.Format(sum)})";
    return (true, msg);
}
```

Возвращает все внесенные в текущей сессии монеты и очищает сессионный кошелек.

****2.3. Класс Product (product.cs)****

Модель товара, хранящая информацию о товарах в автомате.

****Свойства****

```
public int Id { get; }
public string Name { get; }
public int PriceKop { get; }
public int Quantity { get; private set; }
```

* `Id` – уникальный идентификатор товара (только чтение)

* `Name` – название товара (только чтение)

* `PriceKop` – цена в копейках (только чтение)

* `Quantity` – количество товара (чтение/запись только внутри класса)

****Конструктор****

```
public Product(int id, string name, int priceKop, int quantity)
{
    Id = id;
    Name = name;
    PriceKop = priceKop;
    Quantity = quantity;
}
```

Инициализирует все свойства товара переданными значениями.

****Методы управления количеством****

```
public bool HasStock => Quantity > 0;

public void AddStock(int amount) => Quantity += amount;
public void ConsumeOne() => Quantity--;
```

* `HasStock` –проверяет, есть ли товар в наличии

* `AddStock` – добавляет указанное количество товара

* `ConsumeOne` – уменьшает количество на 1 (при покупке)

****2.4. Класс Money (money.cs)****

Статический класс для работы с денежными единицами.

****Константы и массивы****

```
public const int Ruble = 100;  
  
public static readonly int[] SupportedCoins = new[] { 100, 50, 20, 10 };
```

* `Ruble` – константа, представляющая 1 рубль в копейках

* `SupportedCoins = { 100, 50, 20, 10 }` массив поддерживаемых номиналов монет

****Метод форматирования****

```
public static string Format(int amountKop)  
{  
    return string.Format(CultureInfo.InvariantCulture, "{0:0.00} RUB", amountKop / 100.0);  
}
```

* `Format` – Преобразует сумму в копейках в строку формата "X.XX RUB".

****Проверка поддержки монеты****

```
public static bool IsSupportedCoin(int kop) => Array.Exists(SupportedCoins, c => c == kop);
```

Проверяет, поддерживается ли указанный номинал монеты.

****2.5. Класс Wallet (wallet.cs)****

Кошелек для хранения и управления монетами.

****Поле для хранения монет****

```
private readonly Dictionary<int, int> _coins = new();
```

Словарь, где ключ - номинал монеты, значение - количество монет этого номинала.

****Конструкторы****

```
public Wallet()
{
    foreach (var d in Money.SupportedCoins)
        _coins[d] = 0;
}

public Wallet(IDictionary<int, int> initial)
{
    foreach (var d in Money.SupportedCoins)
        _coins[d] = initial.TryGetValue(d, out var c) ? c : 0;
}
```

- Первый конструктор создает пустой кошелек
- Второй инициализирует кошелек с заданными значениями

****Методы управления монетами****

```
public int GetCount(int denom) => _coins.TryGetValue(denom, out var c) ? c : 0;

public void Add(int denom, int count = 1)
{
    if (!_coins.ContainsKey(denom)) _coins[denom] = 0;
    _coins[denom] += count;
}

public bool TryRemove(int denom, int count = 1)
{
    if (GetCount(denom) < count) return false;
    _coins[denom] -= count;
    return true;
}
```

- GetCount - возвращает количество монет указанного номинала
- Add - добавляет указанное количество монет
- TryRemove - пытается удалить монеты, возвращает false если недостаточно

****Расчет общей суммы****

```
public int TotalKop()
{
    var sum = 0;
    foreach (var kv in _coins)
        sum += kv.Key * kv.Value;
    return sum;
}
```

Суммирует стоимость всех монет в кошельке.

****2.6. Класс Calc (calc.cs)****

Статический класс для математических расчетов, в основном для выдачи сдачи.

****Метод расчета сдачи****

```
public static bool TryMakeChange(  
    int amountKop,  
    Wallet machineWallet,  
    out Dictionary<int, int> change)  
{  
    change = new Dictionary<int, int>();  
    var remaining = amountKop;  
  
    foreach (var denom in Money.SupportedCoins)  
    {  
        if (remaining <= 0) break;  
  
        var need = remaining / denom;  
        var avail = machineWallet.GetCount(denom);  
        var take = System.Math.Min(need, avail);  
  
        if (take > 0)  
        {  
            change[denom] = take;  
            remaining -= take * denom;  
        }  
    }  
  
    return remaining == 0;  
}
```

1. Инициализирует словарь для сдачи
2. Проходит по номиналам монет от большего к меньшему
3. Для каждого номинала рассчитывает, сколько монет нужно и сколько доступно
4. Берет минимальное из этих значений
5. Уменьшает оставшуюся сумму
6. Возвращает true, если всю сдачу удалось выдать

3. Использование принципов ООП

Инкапсуляция

- * Внутренние данные защищены через `private` поля
- * Внешний доступ осуществляется через публичные методы

Абстракция

- * Пользователь взаимодействует с высокоуровневыми функциями
- * Детали реализации скрыты внутри классов

Полиморфизм

- * Общие методы обрабатывают объекты через абстрактные интерфейсы и классы
- * Используется переопределение поведения

4. Обработка ошибок

- * Проверка пользовательского ввода
- * Проверка наличия товара и достаточности средств
- * Информативные сообщения при неудачных действиях
- * Предотвращение выхода программы из-за исключений

5. Вывод

Проект успешно реализует эмуляцию работы торгового автомата с поддержкой пользовательского и административного интерфейса. Он демонстрирует грамотное использование принципов ООП, разделение ответственности между классами и устойчивую обработку ошибок.