

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №1

по курсу «ООП»

Тема: Консольное приложение

Выполнила:

Петрова М. В

К3239

Проверил:

Санкт-Петербург

2025 г.

Нужно разработать консольное приложение, которое будет эмулировать вендинговый автомат, позволяющее пользователю:

- Посмотреть список доступных товаров с их ценами и количеством.
- Вставить монеты разных номиналов.
- Выбрать товар и получить его, если внесённой суммы достаточно.
- Получить сдачу (если нужно) и вернуть неиспользованные монеты при отмене операции.
- Администраторский режим для пополнения ассортимента и сбора собранных средств.

Файл **Product.cs**

хранит данные о товаре (идентификатор, название товара, цена, количество)

```
public class Product
```

```
{  
  
    public int Id { get; }  
  
    public string Name { get; }  
  
    public int PriceRub { get; }  
  
    public int Count { get; set; }  
  
    public Product(int id, string name, int priceRub, int count)  
    {  
  
        Id = id;  
  
        Name = name;  
  
        PriceRub = priceRub;  
  
        Count = count;  
    }  
}
```

```
}  
  
}
```

Файл **Program.cs** - точка входа в приложение.

Создаётся объект автомата — `VendingMachine`.

```
VendingMachine machine = new VendingMachine();
```

Дальше идёт бесконечный цикл, который крутится, пока пользователь не выберет выход. Внутри цикла программа печатает текстовое меню с пунктами:

- 1 — показать список товаров;
- 2 — вставить монету;
- 3 — купить товар;
- 4 — забрать сдачу;
- 5 — показать внесённые монеты;
- 9 — перейти в админ-режим;
- 0 — выйти из программы.

Читаем то, что ввел пользователь

```
tring command = (Console.ReadLine() ?? "").Trim();
```

Обрабатываем каждый вариант пользователя через конструкции `if else`

- 1) — Показать товары

```
machine.PrintProducts();
```

- 2) Вставить монету

```
machine.InsertCoin(denom);
```

- 3) Купить товар

```
machine.StartPurchase(productId);
```

4) Забрать сдачу

machine.TakeMoney();

5) Показать внесённые монеты

machine.ShowCoinBank();

6) Админ-режим

machine.AdminMenu();

7) Иначе — неизвестная команда

После выхода из цикла печатаем:

Console.WriteLine("До свидания!");

Файл **Vending.cs** реализует полноценную логику вендингового автомата с подтверждением операций, учётом монет и безопасной выдачей сдачи.

Данные:

Товары:

private readonly List<Product> _products = new();

Монеты автомата:

private readonly Dictionary<int, int> moneybox = new()

Только что вставленные монеты

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

Баланс и выручка

- **_userCreditRub** — деньги, которые уже лежат «внутри автомата» после прошлых покупок.

- `_lastPut` — снимок последней партии внесённых монет (нужно, чтобы вернуть при отмене).
- `CollectedRub` — сколько денег автомат собрал за всё время.
- Есть также свойства `InsertedTotalRub` (сумма монет в лотке) и `AvailableRub` (общий баланс: лоток + кредит).

Показать товары

```
public void PrintProducts()

{

    Console.WriteLine("ID   | Товар                | Цена | Остаток");

    Console.WriteLine("-----");

    foreach (var p in _products)

        Console.WriteLine($"{p.Id,-3} | {p.Name,-16} | {p.PriceRub,4} | {p.Count}");

    Console.WriteLine("-----");

    Console.WriteLine($"Ваш баланс: {AvailableRub} рублей");

}
```

Вставить монету

```
public void InsertCoin(int coinRub)

{

    if (!AllowedCoins.Contains(coinRub))

    {

        Console.WriteLine("Такой номинал не принимается.");

        return;

    }

}
```

```

        if (!_tray.ContainsKey(coinRub)) _tray[coinRub] = 0;

        _tray[coinRub]++;

        Console.WriteLine($"Принято {coinRub} руб. Ваш баланс:
{AvailableRub} рублей");
    }

```

Купить товар

Алгоритм:

1. Найти товар по ID. Если его нет или закончился — ошибка.
2. Проверить, хватает ли денег (AvailableRub).
3. Попросить подтверждение «y/n».
4. Переместить монеты из лотка в банк:

```

5.
    private void MoveInsertedToBank()
6.        int lastMovedSum = 0;
7.        foreach (var kv in _lastPut) lastMovedSum += kv.Key *
    kv.Value;
8.        int totalBefore = _userCreditRub + lastMovedSum;
9.
10.       int newCredit = totalBefore - product.PriceRub;
11.       if (newCredit < 0) newCredit = 0;
12.
13.       if (newCredit > 0 && !CanMakeChange(newCredit))
14.       {
15.           Console.WriteLine("Выдать сдачу невозможно — операция
    отменена.");
16.           ReturnCoins(_lastPut);
17.           return;
18.       }

```

```

19.
20.     product.Count--;
21.     CollectedRub += product.PriceRub;
22.     _userCreditRub = newCredit;
23.
24.     Console.WriteLine($"Покупка успешна. Ваш баланс:
    {_userCreditRub} рублей");
25. }

```

5. Посчитать новый баланс: старый кредит + внесённые монеты – цена товара.
6. Проверить, сможет ли автомат выдать сдачу (CanMakeChange). Если нет — отмена и возврат внесённых монет.
7. Если всё хорошо — уменьшаем остаток товара, увеличиваем выручку и обновляем баланс.

Забрать сдачу:

```

public void TakeMoney()
{
    if (_userCreditRub <= 0) {
        Console.WriteLine("Сдачи нет.");
        return;
    }

    var pack = MakeChange(_userCreditRub);

    if (pack == null)
    {
        Console.WriteLine("Сейчас не могу выдать сдачу на всю
сумму. Попробуйте позже.");

        return;
    }
}

```

```

    }

    Console.WriteLine("Выдача сдачи:");

    foreach (var pair in pack.OrderByDescending(p => p.Key))

        Console.WriteLine($"    {pair.Value} шт × {pair.Key} руб.");

    _userCreditRub = 0;

    Console.WriteLine("Ваш баланс: 0 рублей");

}

```

Админ-меню

Есть отдельная функция AdminMenu(). Вход по паролю admin. Внутри админ может:

- пополнить товары,
- посмотреть содержимое монетного банка,
- забрать выручку,
- пополнить монеты в банке.

```

public void AdminMenu()

{

    Console.Write("Пароль: ");

    var pass = Console.ReadLine();

    if (pass != "admin") { Console.WriteLine("Неверный пароль.");
return; }

    while (true)

    {

        Console.WriteLine("\n[АДМИН] 1) Пополнить товар  2) Монеты
в банке  3) Забрать выручку  4) Пополнить монеты  0) Выход");

```



```
        Console.WriteLine("Выбор: ");

        var cmd = Console.ReadLine();

        if (cmd == "0") break;

        switch (cmd)
        {
            case "1": RestockProducts(); break;

            case "2": PrintCoins(); break;

            case "3": Console.WriteLine($"Забрано: {CollectedRub} руб."); CollectedRub = 0; break;

            case "4": AdminAddCoins(); break;

            default: Console.WriteLine("Неизвестная команда."); break;
        }
    }
}
```

Вывод: Получено рабочее консольное приложение.

Были созданы классы и свойства, а также добавлена обработка пользовательского ввода