Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по Лабораторной работе №1

по дисциплине «**Объектно-Ориентированное Программирование**»

Автор: Конанчук Иван Алексеевич

Факультет: ФПИн

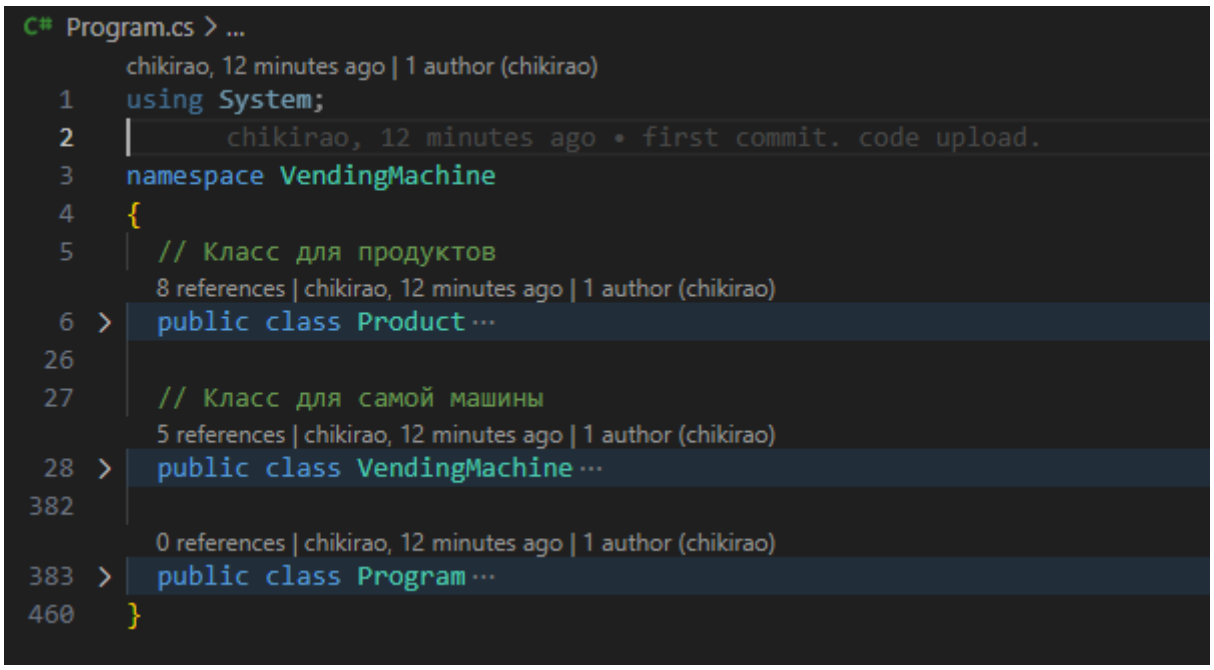
Группа: К3240

Санкт-Петербург, 2025

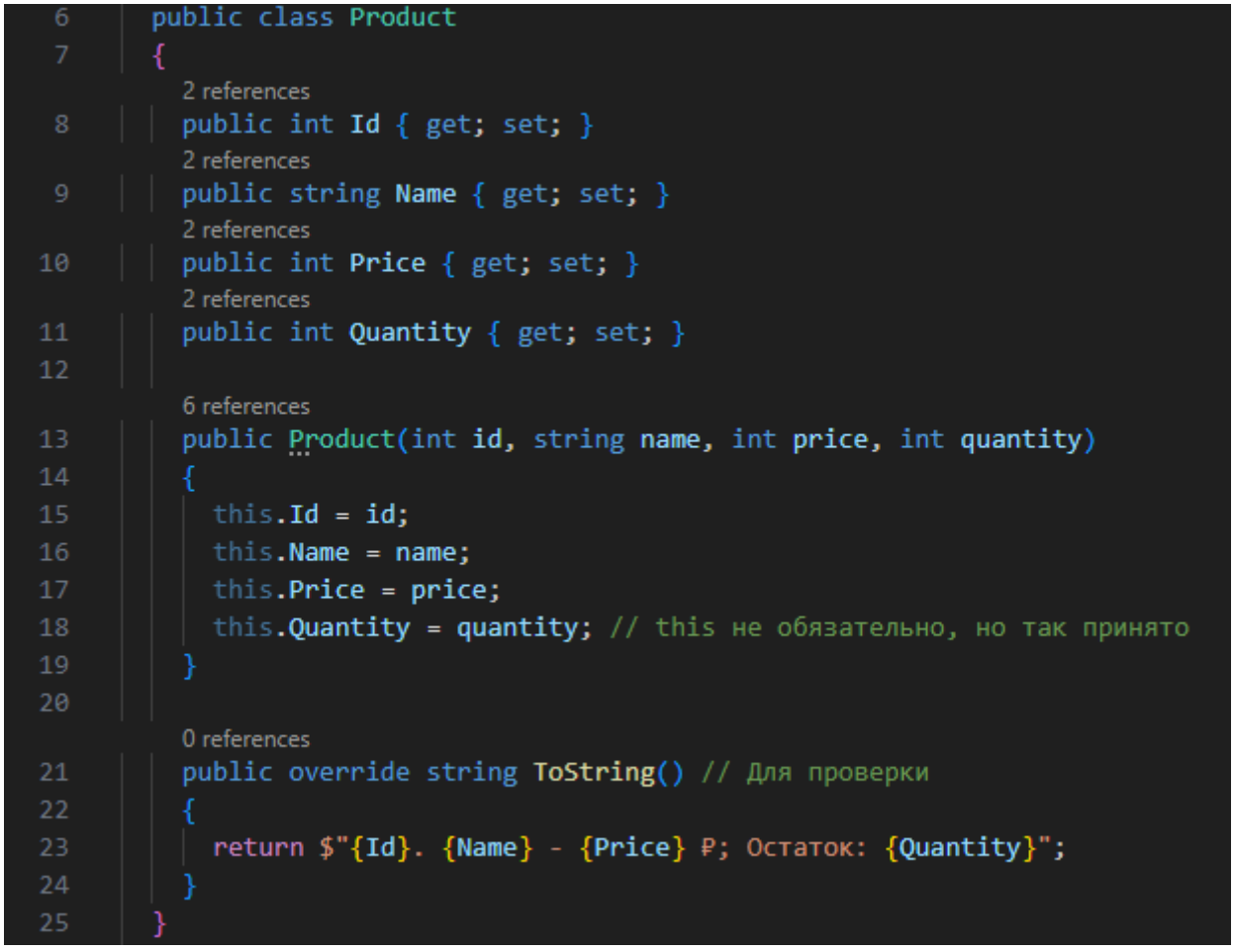
В данной лабораторной нам нужно сделать консольное приложение, имитирующее логику вендингового автомата.

Мной было принято решение реализовывать всю логику приложения в одном файле Program.cs.

Внутри файла логика разделена на три основных класса:



Начал я с класса продукта:



У класса продукта я реализовал 4 автоматических свойства:

1. ID – для нумерации продуктов и доступу к ним по номеру.
2. Имя.
3. Цена.
4. Количество в наличии.

Для проверки работы класса заоверрайдил базовый метод ToString который по дефолту возвращает имя класса.

Дальше я сделал свойства классу Вендинговой машины:

```
// Класс для самой машины
5 references | chikirao, 20 minutes ago | 1 author (chikirao)
public class VendingMachine
{
    // Разрешенные номиналы
    4 references
    private static readonly int[] AllowedDenominations = [10, 5, 2, 1]; // collection expressions с C# 12, в старых версиях new[] { 10, 5, 2, 1 }

    // Нынящний набор товаров
    14 references
    private readonly List<Product> _products = []; // new List<Product>()

    // Сколько монет вставил пользователь
    6 references
    private readonly Dictionary<int, int> _insertedCoins = []; // new Dictionary<int, int>();

    // Вставленные монеты в рублях
    10 references
    public int InsertedRubles { get; private set; } = 0;

    // Счетчик выручки
    7 references
    private int _revenue = 0;

    // Пароль от админки
    1 reference
    private const string AdminPass = "1235";
```

В соответствии с комментариями:

1. AllowedDenominations – Список разрешенных номиналов монет. Я решил использовать в лабораторных только полные рубли, без копеек. Этот список доступен только для чтения.
2. _products – Приватное свойство для хранения всех продуктов.
3. Dictionary – Словарь для хранения внесенных монеток. Каждому номиналу соответствует своё количество.
4. InsertedRubles – Переменная счетчик для внесенных монет в рублях в общем. Приватный сет для на всякий случай для безопасного редактирования.
5. _revenue – Приватная переменная счетчик для хранения собранной выручки.
6. AdminPass – Приватная константа для хранения пароля от админки. Для учебы создана как обычная str переменная, в реальном автомате можно хранить её более безопасно, в переменной окружения, например, и шифровать его как-то.

Дальше добавляем базовые товары для примера. Чтобы не добавлять каждый раз при тесте.

```
    // Генератор стартовых товаров для примера
    1 reference
    public VendingMachine()
    {
        _products.Add(new Product(1, "Вода", 5, 8));
        _products.Add(new Product(2, "Сникерс", 20, 6));
        _products.Add(new Product(3, "Баунти", 15, 5));
        _products.Add(new Product(4, "Чипсы", 13, 4));
        _products.Add(new Product(5, "Банан", 7, 7));
    }
```

После, чтобы каждый раз не возвращаться каждый раз и не копировать символ рубля, хорошей практикой будет сделать форматор, который автоматически будет этот символ добавлять.

```
58      // Форматирование суммы под вид "n ₽"
        15 references
59      public static string FormatRub(int rubles)
60      {
61          return $"{rubles} ₽";
62      }
```

По поводу сдачи, у меня было два варианта. Сделать выдачу сдачи в рублях простым выводом, либо каким-то образом выдавать сдачу монетами. Но для этого надо разбить сдачу по монетам. Вспомнив то, чему нас учили на дисциплине Алгоритмы в прошлом семестре, я решил реализовать выдачу монет жадным алгоритмом. (Он идет от самого большого номинала, пытаясь выдать монету этим номиналом, и, если на такой номинал не хватает, идет к меньшему.) Не знаю как это реализовано в реальных автоматах, но, не думаю, что для этой задаче нужен алгоритм сложнее.

```
64      // Размен сдачи, жадный алгоритм
        1 reference
65      private Dictionary<int, int> MakeChange(int change)
66      {
67          var result = new Dictionary<int, int>();
68          if (change <= 0) return result;
69
70          int remaining = change;
71
72          foreach (var coin in AllowedDenominations)
73          {
74              if (remaining <= 0) break;
75
76              int take = remaining / coin;
77              if (take > 0)
78              {
79                  result[coin] = take;
80                  remaining -= take * coin;
81              }
82          }
83
84          return result;
85      }
```

Реализовал эти методы сначала, потому что они будут использоваться дальше в коде.

Пока перейдем в класс Program, в нем будет функция Main и логика программы вывода.

Я вынес в отдельные функции программы 6 команд:

```
0 references | chikirao, 38 minutes ago | 1 author (chikirao)
public class Program
{
    0 references
    public static void Main()
    {
        Console.OutputEncoding = System.Text.Encoding.UTF8; // Чтоб символ рубля выводился нормально
        var machine = new VendingMachine(); // Создаем сам объект вендингого автомата

        while (true)
        {
            Console.WriteLine("\nВЕНДИНГОВЫЙ АВТОМАТ: ");
            Console.WriteLine("  1. - Показать список товаров.");
            Console.WriteLine("  2. - Вставить монету.");
            Console.WriteLine("  3. - Купить товар.");
            Console.WriteLine("  4. - Отмена, возврат монет.");
            Console.WriteLine("  5. - Войти в админ панель (пароль обязателен).");
            Console.WriteLine("  0. - Выход из сессии.");
            Console.Write("ВВОД: ");
            var command = Console.ReadLine();
```

При первых тестах были проблемы с некоторыми символами, поэтому меняем output консоли на UTF8.

Вернемся к классу VendingMachine. Раз уж команды определены, можно реализовывать методы. Первый – вывод всех продуктов.

```
// Вывод продуктов
4 references
public void ShowProducts()
{
    Console.WriteLine("\nСписок товаров:");
    foreach (var product in _products)
    {
        Console.WriteLine($"  {product.Id}. {product.Name} - {product.Price}Р. Доступно: {product.Quantity}");
    }
}
```

Циклом проходится по всем продуктам и выводит.

Второй – вспомогательный метод для метода вставки монет и третий – сам метод вставки.

```
// Показать разрешенные номиналы
1 reference
public void ShowAllowedDenominations()
{
    Console.WriteLine("Разрешенные номиналы: " + string.Join(", ", AllowedDenominations.Select(FormatRub)));
}

// Вставка монет
1 reference
public bool InsertCoin(int coin)      chikirao, 38 minutes ago • first commit. code upload.
{
    if (!AllowedDenominations.Contains(coin))
    {
        Console.WriteLine("  Такой номинал не принимается!");
        return false;
    }

    InsertedRubles += coin;
    _insertedCoins[coin] = _insertedCoins.GetValueOrDefault(coin) + 1;

    Console.WriteLine($"Принято {FormatRub(coin)}. Текущая сумма монет: {FormatRub(InsertedRubles)}.");
    return true;
}
```

Монета проверяется на номинал и, если всё хорошо, добавляется к вставленным монетам а номинал прибавляется к общей сумме рублей.

Далее возврат монет и отмена операции. Эти две функции я объединил в одну команду:

```
119     // Отмена операции, возврат средств
        1 reference
120     public Dictionary<int, int> CancelXRefund()
121     {
122       if (InsertedRubles == 0 || _insertedCoins.Count == 0)
123       {
124         Console.WriteLine("  Вы не вставили монеты, возвращать нечего.");
125         return [];
126       }
127
128       var refund = new Dictionary<int, int>(_insertedCoins); // Сохраняем копию
129       Console.WriteLine($"Отмена операции...\nВозврат средств: {FormatRub(InsertedRubles)}.");
130       _insertedCoins.Clear();
131       InsertedRubles = 0;
132
133       return refund;
134     }
```

Теперь основной и крупнейший метод – покупка товара. Часть 1:

```
136     // Покупка товара
        1 reference
137     public bool Purchase(int productId)
138     {
139       // Отменяем покупку если 0
140       if (productId == 0)
141       {
142         Console.WriteLine("  Покупка отменена!");
143         return false;
144       }
145
146       // Поиск продукта по ID
147       var product = _products.Find(p => p.Id == productId);
148       if (product == null)
149       {
150         Console.WriteLine("Товар не найден.");
151         return false;
152       }
153
154       // Проверка есть ли в автомате
155       if (product.Quantity <= 0)
156       {
157         Console.WriteLine("Товар закончился.");
158         return false;
159       }
160
161       // Проверка достаточно ли денег
162       int deficit = product.Price - InsertedRubles;
163       if (deficit > 0)
164       {
165         Console.WriteLine($"  Недостаточно средств!\n  Необходимо доплатить: {FormatRub(deficit)}.");
166         return false;
167       }
```

Тут, как будет указано позже, покупка отменяется при вводе 0, ищется товар по ID, проверяется сток и достаточно ли денег.

Часть 2:

```
169         // Подсчет сдачи
170         int changeCalc = InsertedRubles - product.Price;
171         if (changeCalc > 0)
172         {
173             var changeToGive = MakeChange(changeCalc);
174 
175             Console.WriteLine($"Сдача: {FormatRub(changeCalc)}.");
176             Console.WriteLine("Состав сдачи:");
177 
178             foreach (var coin in AllowedDenominations)
179             {
180                 if (changeToGive.TryGetValue(coin, out var cnt) && cnt > 0)
181                 {
182                     Console.WriteLine($" * {FormatRub(coin)} - {cnt} шт.");
183                 }
184             }
185         }
```

Тут считается сдача.

Часть 3:

```
187         // Выдаем товар
188         product.Quantity--;
189         _revenue += product.Price;
190         Console.WriteLine($"Выдан товар: {product.Name}.");
191 
192         // Очистка и выход
193         _insertedCoins.Clear();
194         InsertedRubles = 0;
195 
196         return true;
197     }
```

Системное сообщение и выход.

Дальше идет админ панель:

```
199     // Секция АДМИН ПАНЕЛИ
        1 reference
200     public void AdminPanel()
201     {
202       Console.Write("Введите ПИНкод: ");
203       var pin = Console.ReadLine();
204       if (pin != AdminPass)
205       {
206         Console.WriteLine("  Неверный ПИНкод.");
207         return;
208       }
209
210       while (true)
211       {
212         Console.WriteLine("\n  АДМИН-ПАНЕЛЬ");
213         Console.WriteLine(" 1. - Показать товары.");
214         Console.WriteLine(" 2. - Пополнить остаток товара.");
215         Console.WriteLine(" 3. - Создать новый товар.");
216         Console.WriteLine(" 4. - Собрать выручку.");
217         Console.WriteLine(" 0. - Выйти в режим покупателя.\n");
218
219         Console.Write("ВВОД: ");
220         var command = Console.ReadLine();
221         Console.WriteLine();
```

Для админки я прописал 6 команд.

Логика обработки команд:

```
223             if (command == "0") break;
224
225             switch (command)
226             {
227                 case "1":
228                     ShowProducts();
229                     Console.WriteLine($"Текущая выручка: {FormatRub(_revenue)}");
230                     break;
231
232                 case "2":
233                     AdminRestockProduct();
234                     break;
235
236                 case "3":
237                     AdminNewProduct();
238                     break;
239
240                 case "4":
241                     AdminRevenue();
242                     break;
243
244                 default:
245                     Console.WriteLine("  Неизвестная комманда!");
246                     break;
247             }
248         }
249     }
```

Дальше методы, первый – Пополнение количества товара:

```
251     // Админка: пополнить товары
        1 reference
252     private void AdminRestockProduct()
253     {
254       ShowProducts();
255       Console.Write("Введите ID товара для пополнения: ");
256       if (!int.TryParse(Console.ReadLine(), out var id))
257       {
258         Console.WriteLine("  Некорректный ID.");
259         return;
260       }
261
262       // Поиск товара
263       var product = _products.FirstOrDefault(p => p.Id == id);
264       if (product == null)
265       {
266         Console.WriteLine("  Товар не найден.");
267         return;
268       }
269
270       // Пополняем на конкретное число
271       Console.Write("Кол-во пополнения: ");
272       if (!int.TryParse(Console.ReadLine(), out var cnt) || cnt <= 0)
273       {
274         Console.WriteLine("  Некорректное кол-во!");
275         return;
276       }
277
278       // Увеличиваем кол-во
279       product.Quantity += cnt;
280       Console.WriteLine("Добавлено.");
281       Console.WriteLine($"Остаток товара сейчас \"{product.Name}\" - {product.Quantity} шт.");
282     }
```

Следующий метод – Создание нового продукта. Часть 1 – ввод параметров:

```
284     // Создать новый продукт
        1 reference
285     private void AdminNewProduct()
286     {
287       // Ввод названия
288       Console.Write("Введите название товара: ");
289       var name = (Console.ReadLine() ?? "").Trim(); // На всякий обрезаем пробелы вокруг
290       if (string.IsNullOrWhiteSpace(name))
291       {
292         Console.WriteLine("  Вы ввели пустое название.\nТовар не создан!");
293         return;
294       }
295
296       // Ввод цены
297       Console.Write("Введите цену в рублях: ");
298       if (!int.TryParse(Console.ReadLine(), out var price) || price <= 0)
299       {
300         Console.WriteLine("  Некорректная цена.");
301         return;
302       }
303
304       // Ввод колличества
305       Console.Write("Введине кол-во товара: ");
306       if (!int.TryParse(Console.ReadLine(), out var quantity))
307       {
308         Console.WriteLine("  Колличество введено некорректно!");
309       }
```

Часть 2, назначение ID:

```
311         // Назначаем ID.
312         int newid;
313         if (_products.Count == 0)
314         {
315           newid = 1;
316         }
317         else
318         {
319           int maxid = _products[0].Id;
320
321           for (int i = 1; i < _products.Count; i++)
322           {
323             if (_products[i].Id > maxid)
324             {
325               maxid = _products[i].Id;
326             }
327           }
328
329           newid = maxid + 1;
330         }
331
332         _products.Add(new Product(newid, name, price, quantity));
333         Console.WriteLine($"Товар добавлен id:{newid} - {name}, {FormatRub(price)}. {quantity} шт.");
334       }
```

Если продуктов 0, то ID задается 1, если уже есть продукты, то новый ID – максимальный + 1.

Следующая и последняя функция – Сбор выручки:

```
337     private void AdminRevenue()
338     {
339       if (_revenue <= 0)
340       {
341         Console.WriteLine("  Выручка отсутствует!");
342         return;
343       }
344
345       Console.WriteLine($"  Выручка к сбору: {FormatRub(_revenue)}");
346       Console.Write("Введите сумму для вывода (0 для отмены): ");
347
348       var revenueInput = (Console.ReadLine() ?? "").Trim();
349
350       if (!int.TryParse(revenueInput, out var revenueToCollect) || revenueToCollect < 0)
351       {
352         Console.WriteLine("  Сумма введена некорректно!");
353         return;
354       }
355
356       if (revenueToCollect == 0)
357       {
358         Console.WriteLine("  Вывод отменен.");
359         return;
360       }
361
362       if (revenueToCollect > _revenue)
363       {
364         Console.WriteLine("  Введенная сумма больше доступной!");
365         return;
366       }
367
368       _revenue -= revenueToCollect;
369       Console.WriteLine($"Вы забрали: {FormatRub(revenueToCollect)}.\nВ автомате осталось: {FormatRub(_revenue)}.");
370     }
371   }
```

На этом заканчивается класс VendingMachine, поэтому переходим обратно в Program.

Реализация логики обработки команд по аналогии с админкой:

```
373     public class Program
375       public static void Main()
388           Console.WriteLine("  0. - Выход из сессии.");
389           Console.Write("ВВОД: ");
390           var command = Console.ReadLine();
391
392           if (command == "0") break;
393
394           switch (command)
395           {
396             case "1":
397               machine.ShowProducts();
398               Console.WriteLine($"Текущая внесенная сумма: {VendingMachine.FormatRub(machine.InsertedRubles)}.");
399               break;
400
401             case "2":
402               machine.ShowAllowedDenominations();
403               Console.Write("Введите номинал монеты: ");
404               if (!int.TryParse(Console.ReadLine(), out var coin) || coin <= 0)
405               {
406                 Console.WriteLine("  Данный номинал не принимается!");
407                 break;
408               }
409               machine.InsertCoin(coin);
410               break;
411
412             case "3":
413               machine.ShowProducts();
414               Console.WriteLine($"\nВнесено (руб): {VendingMachine.FormatRub(machine.InsertedRubles)}.");
415               Console.Write("Введите ID товара (0 для отмены): ");
416               if (!int.TryParse(Console.ReadLine(), out var productId))
417               {
418                 Console.WriteLine("  Некорректный ID!");
419                 break;
420               }
421               machine.Purchase(productId);
422               break;
423
424             case "4":
425               var refund = machine.CancelXRefund();
426               if (refund.Count > 0)
427               {
428                 Console.WriteLine("Возврат монет: ");
429                 foreach (var coinD in refund.Keys.OrderByDescending(i => i))
430                 {
431                   Console.WriteLine($"  {VendingMachine.FormatRub(coinD)} - {refund[coinD]} шт.");
432                 }
433               }
434               break;
435
436             case "5":
437               machine.AdminPanel();
438               break;
439
440             default:
441               Console.WriteLine("  Такой команды нет!");
442               break;
443           }
444         }
```

Вся логика обработки команд помещена в цикл бесконечный while (true):

```
380         while (true)
381         {
382           Console.WriteLine("\nВЕНДИНГОВЫЙ АВТОМАТ: ");
```

Соответственно выход из цикла означает выход из сессии командой 0:

```
445
446           // Если цикл прекратился - пользователь закончил сессию
447           Console.WriteLine("Выход...\nПока 💋💋💋");
448         }
449       }
450     }
```

Из будущих улучшений нашей системы могу придумать очистку терминала при вводе, например введена команда Показать товары, и терминал очищается, видно только товары, потом на ввод выход обратно в меню итд.

Так же отсутствует troubleshooting из-за того, что все операции у нас виртуальные, в реальности я думаю присутствует какая-то асинхронность, логи, авто-перезапуски.

Помимо этого, можно имитировать таймаут, чтобы система входила в сонный режим если пользователь долго не вводит команды.

В отличие от реального автомата, у нас ненастоящие деньги, «безграничный» запас монет для сдачи (хотя в реальности на сдачу с покупки монет может не хватить).

Ну и unit тестирование тоже лишним в реальной системе не будет.

Таким образом, мы реализовали логику работы вендингового автомата, и овладели базовыми знаниями языка C#.