Day 00.5

## Общие требования

* Убедитесь, что на вашем компьютере установлен [SDK для разработки на .NET 5](https://dotnet.microsoft.com/download) и вы используете именно его.
* Помните, ваш код будут читать! Обратите особое внимание на оформление вашего кода и именование переменных. Придерживайтесь общепринятых стандартов [C# Coding Conventions](https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions).
* Самостоятельно выберите удобную для себя IDE.
* Программа должна иметь возможность запуска через командную строку dotnet.
* В каждом из заданий указаны входные параметры и формат ответа на выходе. Необходимо придерживаться их.
* В начале каждого задания приведен список разрешенных языковых конструкций.
* Если затрудняетесь в решении задачи, обратитесь с вопросами к другим участникам бассейна, интернету, Google, посмотрите на StackOverflow.
* С основными возможностями языка C# можно ознакомиться в [официальной спецификации](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/language-specification/introduction).
* Вы демонстрируете все решение, верный результат работы программы — лишь один из способов проверки ее корректной работы. Поэтому когда необходимо получить определенный вывод в результате работы ваших программ, запрещено показывать предрассчитанный результат.
* Обращайте особое внимание на термины, выделенные **bold** шрифтом: их изучение пригодится вам как в выполнении текущего задания, так и в вашей дальнейшей карьере .NET разработчика.
* Have fun :)

## Требования к заданиям дня

* Нельзя использовать nuget-пакеты.
* Весь код заданий должен находиться в одном консольном приложении, созданном на основе стандартного шаблона .NET SDK.
* Каждому классу должен соответствовать отдельный файл.
* Используйте **top-level-statements** и **var**.
* Название решения (и его отдельного каталога) - d{*xx*}, где *xx* - цифры текущего дня. Название проекта то же.

## Что нового

* Классы, свойства и методы
* Конструкторы классов
* Лямбда-операторы
* Переопределение методов
* Инкапсуляция
* Статические классы
* Методы-расширения

## Интро

В который раз возвращаясь из магазина с продуктами, где снова провели в очереди около полутора вечностей, вы завороженно размышляете о природе этих мест. Магазинов, собственно. И их очередей. Отчего появляются последние? От малого ли количества работающих касс, из-за нерасторопных кассиров или людей с большим количеством товара в корзине. Почему всегда кажется, что соседняя касса работает быстрее?

Эта загадка природы никак не дает вам покоя. И вы решаете если не разобраться в ней, то по крайней мере смоделировать с помощью силы, которая всегда приходила вам на помощь в затруднениях.

С помощью программирования.



[xkcd](https://xkcd.com/208/)

# 00. Склад

Начнем с сердца магазина. Создайте класс *Storage* и опишите его **свойства**. Главное для нас, что склад определяется целочисленным количеством товаров (условимся, одинаковых), которые в нем хранятся.

Количество товаров может меняться, но в момент создания склада оно определяется его вместимостью - стартовой величиной. Пусть количество товаров задается в **конструкторе класса**. Склад не может быть создан без вместимости, значит у него не должно быть **конструктора без параметров**.

Добавьте в класс метод *IsEmpty*, который будет возвращать информацию о том, что товары на складе закончились. Используйте [лямбда-оператор](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/lambda-operator).

# 01. Покупатель

Как насчет покупателей? Создайте класс *Customer* и опишите его свойства. Сделайте так, чтобы одного покупателя можно было отличить от другого: пусть у него будет имя и порядковый номер, под которым он попал в магазин. Так как классы наследуются от **типа Object** и получают доступ к его методам, [переопределите метод ToString()](https://docs.microsoft.com/en-us/dotnet/api/system.object.tostring?view=net-5.0) так, чтобы он возвращал строковую информацию с номером покупателя и его именем.

Имя и номер - неизменные величины. Сделайте эти **автосвойства** [закрытыми](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/restricting-accessor-accessibility) для редактирования извне и задающимися только из **конструктора класса**. Покупатель не может быть создан без них, значит у него не должно быть **конструктора без параметров**.

Попробуйте создать двух покупателей с одинаковыми именами и номерами:

var customer1 = new Customer("Andrew", 1);

var customer2 = new Customer("Andrew", 1);

Выведите в консоль значение выражения customer1 == customer2. Почему ответ именно такой?

Измените класс *Customer* [так](https://docs.microsoft.com/en-us/dotnet/api/system.object.equals?view=net-5.0), чтобы в консоль вывелось true.

# 02. Корзина

Возьмите класс *Customer* из предыдущего задания, давайте добавим ему свойство, отражающее количество товаров в его корзине. При создании покупателя (в реальном мире - появлении его в магазине) товаров у него ноль.

Чтобы не давать пользователям ваших классов менять значение, как заблагорассудится, сделайте поле с количеством товаров не изменяемым извне.

Вместо этого добавьте покупателю метод заполнения корзины *FillCart*: пусть метод принимает максимальную вместимость корзины и заполняет его количество товаров рандомным числом (минимум 1, максимум вместимость корзины).

Так мы реализовали логику заполнения корзины в одном месте и закрыли ее удобным методом с читаемым и понятным названием. Добро пожаловать в **инкапсуляцию**.

Создайте 3 пользователей, заполните им корзины вместимостью 15 товаров. Выведите в консоль информацию о пользователе и количество товаров в их корзине.

# 03. Касса

Создайте класс *CashRegister* и опишите его свойства. Пусть у кассы будет “название” - какой-то заголовок, который позволит покупателям однозначно понять, о какой кассе речь. Переопределите методы класса так, чтобы ToString() возвращал заголовок и чтобы две кассы с одинаковым заголовком были равными.

У кассы него не должно быть **конструктора без параметров**.

Еще у кассы не должно быть очередей. Но мы живем в реальном мире и именно для реального мира хотим построить модель, поэтому давайте реализуем очереди.

Добавьте классу *CashRegister* коллекцию *Customers* так, чтобы это была [очередь](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.queue-1?view=net-5.0) типа “**первым поступил - первым обслужен**”.

Подумайте, нужно ли инкапсулировать это свойство?

# 04. Магазин

Создайте класс *Store* и опишите его свойства. У магазина должен быть склад и набор касс.

Реализуйте конструктор класса так, чтобы он принимал на входе целочисленную вместимость склада и целочисленное количество касс. Заполните свойства класса в конструкторе, используя эти входные данные. Пусть каждая касса создается с заголовком, соотвествующим ее порядковому номеру.

Подумайте, есть ли смысл закрывать от редактирования **сеттер** коллекции касс?

Добавьте свойство *IsOpen*, возвращающее true, если на складе ещё остались товары.

# 05. Очереди

Как вы встаете в очередь? В кассу с наименьшим количеством покупателей или в кассу с наименьшим количеством покупок? Реализуем обе ситуации.

Создайте **статический** класс *CustomerExtensions* с двумя **extension методами** для объекта *Customer*: оба метода будут принимать в себя коллекцию касс (объектов *CashRegister*)и возвращать ту кассу, которую выберет покупатель*.* Первый метод - кассу с наименьшим количеством покупателей, второй - с наименьшим количеством товаров среди всех покупателей в очереди этой кассы.

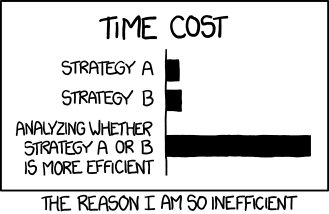
Не забудьте назвать методы так, чтобы было понятно, что именно они делают.

# 06. So how does it work?

Создайте магазин с 3 кассами и складом вместимостью 40. Создайте 10 разных покупателей. Реализуйте цикл: пусть каждый из покупателей заполнит корзину и встанет в очередь. Цикл должен работать, пока магазин открыт и в нем есть покупатели. Учтите, что заполнение корзины покупателя товарами должно убирать их со склада. Если на складе осталось меньше товаров, чем в корзине пользователя, ему нужно дать столько, сколько осталось на складе.

Сделайте это для двух разных случаев выбора очереди (задание 05) и в каждом выведите в цикле в консоль имя пользователя и его номер, количество товаров, выбранную им кассу и количество людей на кассе, количество товаров в корзинах очереди.

Вместимость корзины - 7 товаров.



[xkcd](https://xkcd.com/1445/)