ООП в C#

(лямбда-выражения, библиотеки классов, chatbot: постановка задачи)

Андрей Голяков

Лямбда-выражения

Экземпляр делегата можно инициализировать лямбда-выражением.

Отличительной чертой лямбд является оператор `=>` , который делит выражение на левую часть с параметрами и правую с телом метода.

Например, если определён так класс-делегат

```
delegate int DoCalculation(int number1, int number2);
```

то экземпляр может быть создан как обычным приравниванием метода с необходимой сигнатурой

```
DoCalculation action1 = Sum; // assuming this method defined
```

а может быть записан в более короткой форме лямбда-выражения

```
DoCalculation action2 = (int x, int y) => x * y;
```

Лямбда-операторы

Если подходить строго, это именно лямбда-выражение:

```
DoCalculation action2 = (int x, int y) => x * y;
```

Но это может быть и лямбда-оператор, если мы заключим его в блочные скобки:

```
DoCalculation action3 =
    (int x, int y) =>
    {
        int z = x * y;
        return z;
    };
```

Лямбда-выражения и лямбда-операторы

Допускается не указывать типы аргументов, ведь компилятор и так знает тип и сигнатуру вашего делегата:

```
DoCalculation action2 = (x, y) => x * y;
```

и

```
DoCalculation action3 =
    (x, y) =>
    {
        int z = x * y;
        return z;
    };
```

Лямбда-выражения и лямбда-операторы

В случае если имеется лишь один аргумент то можно опустить обрамляющие его скобки:

```
// you can omit parentheses in case of the single parameter
Action<string> action4 = x => Console.WriteLine(x);
action4("test");
```

Если в сигнатуре делегата нет аргументов, то необходимо указать пустые скобки:

```
// you should use empty parentheses in case of absense ot the parameters
Func<float> getPi = () => MathF.PI;
Console.WriteLine(getPi());
```

Самостоятельная работа

Переписать расчёт периметра и площади окружности на использование лямбда-выражений вместо методов класса.

Добавить функцию вычисление диаметра и также вывести результат расчёта диаметра на экран.

Библиотеки классов

Библиотека классов определяет типы и методы, которые могут быть вызваны из любого приложения или других библиотек.

Если вы создадите библиотеку классов, вы сможете по своему усмотрению распространять ее как независимый компонент или включить в состав одного или нескольких приложений.

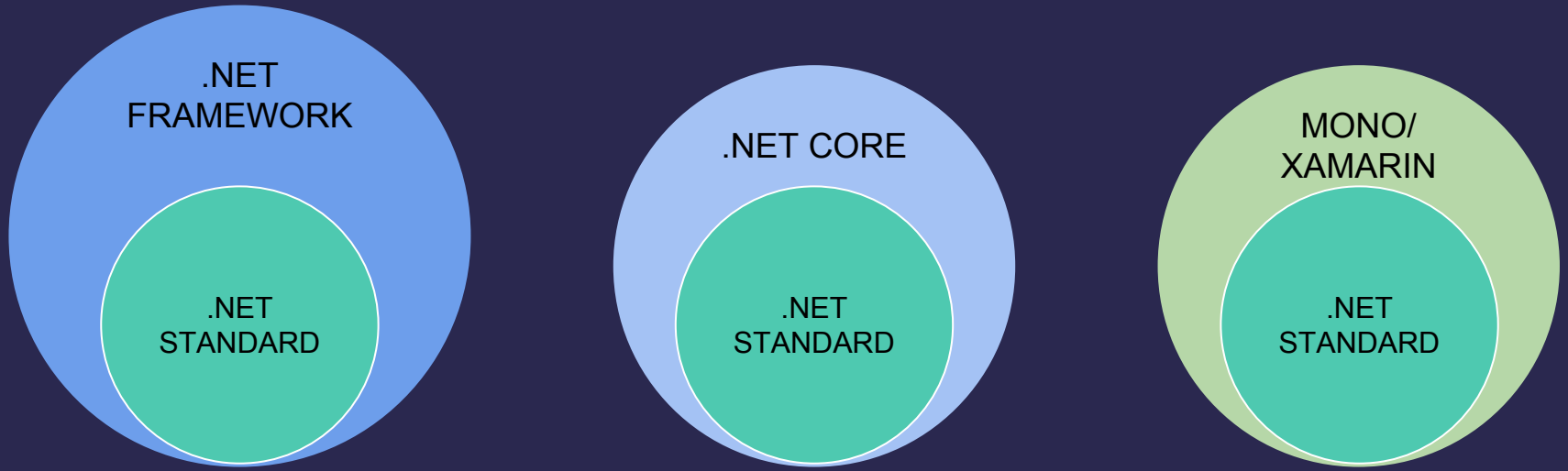
Чтобы создать библиотеку классов необходимо выбрать соответствующий тип проекта - Class Library.

Существует несколько возможностей создать библиотеку классов:

- Class Library (.NET Core)
- Class Library (.NET Standard)

.NET Standard

.NET Standard Library — это формальный набор спецификаций общих интерфейсов других платформ: .NET Core, .NET Framework, Mono/Xamarin и остальных.



Библиотека классов под .NET Standard 2.0

Библиотеку классов, предназначенную для .NET Standard 2.0, можно вызывать из любой реализации .NET, которая поддерживает эту версию .NET Standard. Для .NET Standard 2.0 это:

- .NET Core 2.0 и выше
- .NET Framework 4.6.1 (рекомендуется 4.7.2 и выше)
- Mono 5.4
- Xamarin.iOS 10.14
- Xamarin.Mac 3.8
- Xamarin.Android 8.0
- Универсальная платформа Windows 10.0.16299
- Unity 2018.1

Настройка зависимостей

Чтобы подключить библиотеку классов к проекту, в котором планируется использовать её классы необходимо в Solution Explorer открыть контекстное меню на пункте Dependencies (зависимости) и выбрать пункт Add Reference (добавить ссылку).

В открывшемся диалоговом окне выбрать слева пункт Projects (проекты) и отметить галочками необходимые проекты в солюшене и нажать OK.

Самостоятельная работа

За основу берётся код самостоятельной работы, выполненной последней на 15 ом уроке: классы Circle и CircleOperation.

- Класс Circle вынести в отдельную сборку с именем Calculator.Figure типа .NET Standard
- Класс CircleOperation вынести в отдельную сборку с именем Calculator.Operation типа .NET Core
- Создать консольное приложение .NET Core в которое поместить логику расчёта параметров окружности используя внешние классы.
- Добавить в соответствующие библиотеки классы Square и SquareOperation для описания квадрата и расчёта его периметра и площади.
- В консольном приложении также рассчитать и вывести параметры квадрата.

Chatbot: Постановка задачи

Разработать программу, которую можно было бы зарегистрировать в качестве бота одного из чат сервисов.

Основная функциональность

- Принимать в сообщении будильник: сообщение и время срабатывания
- В назначенное время посылать в ответ сообщение-напоминание.

Дополнительные требования

- Сообщения должны оставаться в хранилище программы даже после срабатывания.
- В первой версии приложения, хранилище будет in-memory коллекцией, однако оно должно быть написано так, чтобы обеспечить легкую замену другой реализацией в будущих версиях.

Chatbot: Постановка задачи

Разработать программу, которую можно было бы зарегистрировать в качестве бота одного из чат сервисов.

Основная функциональность

- Принимать в сообщении будильник: сообщение и время срабатывания
- В назначенное время посылать в ответ сообщение-напоминание.

Дополнительные требования

- Сообщения должны оставаться в хранилище программы даже после срабатывания.
- В первой версии приложения, хранилище будет in-memory коллекцией, однако оно должно быть написано так, чтобы обеспечить легкую замену другой реализацией в будущих версиях.

Домашнее задание

1. Подумать, какие можно выделить интерфейсы и классы для решения задачи чат бота?
2. Как они будут взаимодействовать между собой?

Спасибо за внимание.