Programowanie w C#

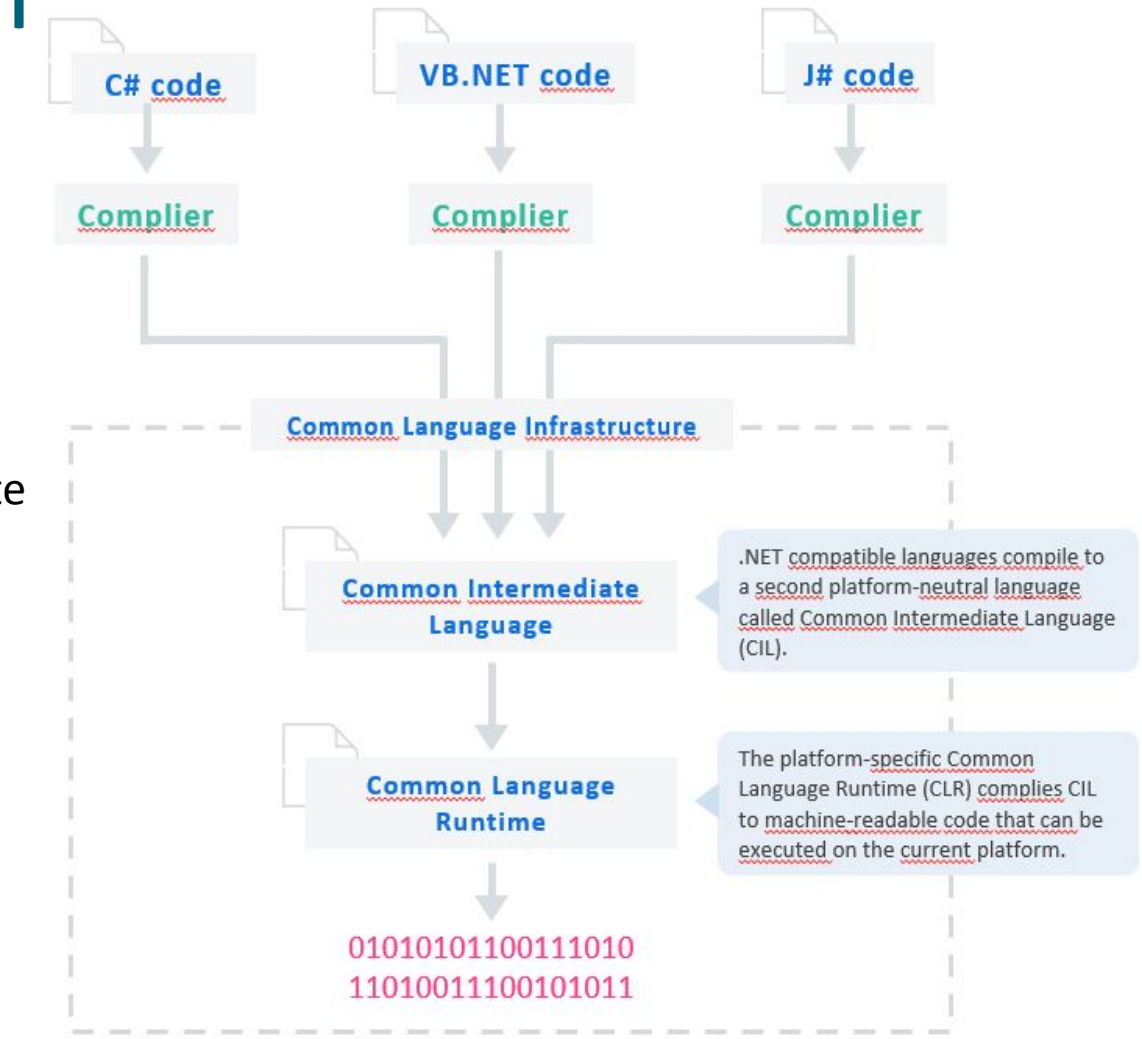
Paweł Biesiada

Architektura .NET

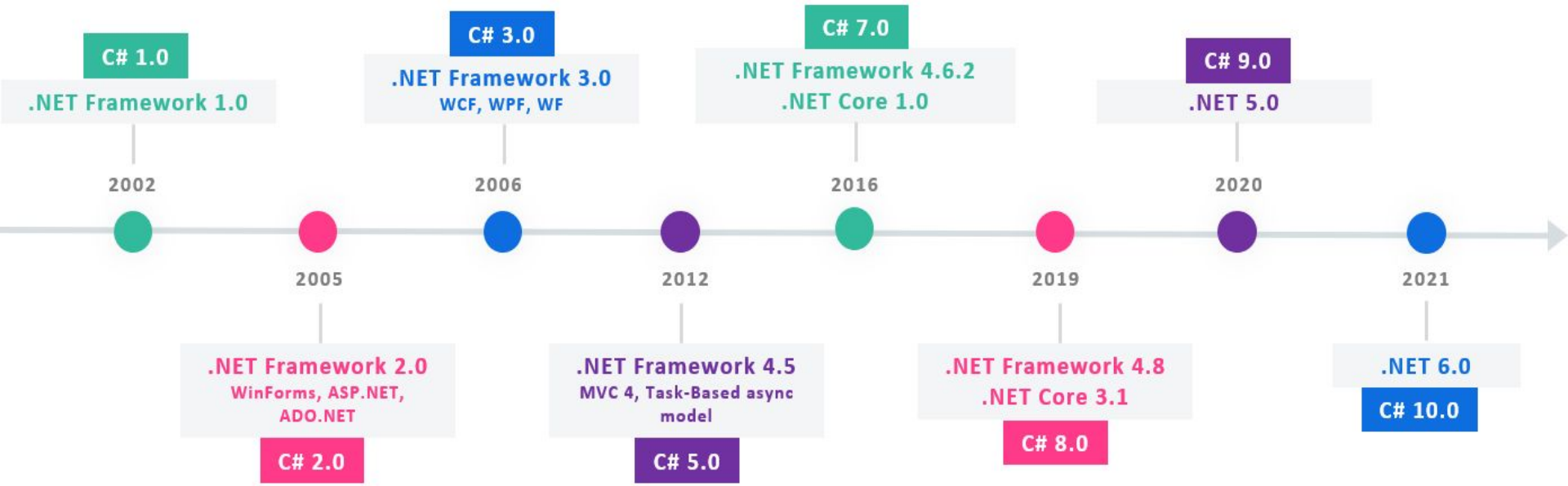
CLI Common Language Infrastructure

CIL Common Intermediate Language

CLR Common Language Runtime



Ewolucja.NET



Składowe klasy

- **Pole (field)**
- **Metoda (method)**
- **Właściwość (property)**
- Indeksator (indexer)
- **Konstruktor (constructor)**
- Destruktor (finalizer)
- Zdarzenie (event)
- Operator (operator)

Typy danych

Typy wartościowe
(mają wartość domyślną)

- Proste
  - **int**, short, long, sbyte, byte, ushort, uint, ulong, **char**, float, **double**, **decimal**, **bool**
- Wyliczeniowy – enum
- Struktura - struct
- Nullable
- Tuple

Typy referencyjne
(mogą nie mieć wartości – *null*)

- Klasa – class
  - object, string
- Interfejs – interface
- Tablice
  - array
- Delegaty - delegate

Typy danych – cd.

Type	Size in Bytes	Description	Minimum	Maximum	Example
bool	1	Named literal	false	true	
sbyte	1	Signed byte	-128	127	
byte	1	Unsigned byte	0	255	
short	2	Signed short integer	-32768	32767	
ushort	2	Unsigned short	0	65535	
int	4	Signed integer	-2147483648	2147483647	
uint	4	Unsigned integer	0	4294967295	
long	8	Signed long int	-9.2233E+18	9.2233E+18	
ulong	8	Unsigned long int	0	18446E+19	
char	2	Unicode character, contained within single quotes.	0	128	a,b,4
float	4	floating point	-3.402823E+38	3.402823E+38	3.14159
double	8	Floating point	-1.7976931E+308	1.7976931E+308	3.14159
decimal	16	Floating point, accurate to the 28th decimal place.	-7.9228E+24	7.9228E+24	
object	8+	Base type for all other types	n/a	n/a	n/a
string	20+	Immutable character array	n/a	n/a	"Hello World"
DateTime	8	Represents an instant in time, typically expressed as a date and time of day.	00:00:00 01/01/0001	23:59:59 31/12/9999	14:289:35 08/05/2010

Instrukcje

- Deklaracje – zmiennych, const
- Warunkowe – if, switch
- Pętle – for, foreach, while, do/while
- Skoku – return, yield, throw, continue, break
- Wrappery na instrukcje – using, lock

Modyfikatory dostępu (hermetyzacja)

- public – dostęp nieograniczony
- private – dostęp tylko wewnątrz klasy
- protected – dostęp wewnątrz klasy i klasach dziedziczących
- internal – dostęp w obrębie jednego assembly
- protected internal – dostęp wewnątrz klasy, klasach dziedziczących, lub w obrębie danego assembly
- private protected (C# 7.2) – dostęp wewnątrz klasy, klasach dziedziczących w obrębie danego assembly

Klasa vs struktura

class

- Typ referencyjny
- Wspiera dziedziczenie i polimorfizm
- Posiada destructor
- Może mieć zdefiniowany konstruktor bezparametrowy
- Przechowywany na stercie
- Do metody przekazujemy tylko wskaźnik na obiekt w pamięci

struct

- Typ wartościowy
- Nie można dziedziczyć ze struktury
- Nie posiada destruktora
- Zdefiniowany konstruktor musi posiadać parametry
- Przechowywany na stosie
- Do metody przekazujemy cały obiekt
- Szybki – ale dla “małych” obiektów

Elementy statyczne

- Słowo kluczowe *static*
- Zmienna statyczna
- Metoda statyczna
- Klasa statyczna

Interfejs vs klasa abstrakcyjna

Interface

- Wspiera wielokrotne dziedziczenie
- Nie posiada zmiennych klasy (field)
- Zawiera tylko sygnatury składowych klasy
- Nie posiada modyfikatorów dostępu (wszystkie domyślnie są jako public)
- Nie można zdefiniować statycznej składowej klasy (C# 8.0)

Abstract class

- Można dziedziczyć tylko z jednej klasy
- Może posiada wszystkie elementy klasy
- Może zawierać całe implementacje lub tylko sygnatury składowych klasy
- Można deklarować różne modyfikatory dostępu
- Tylko w pełni zadeklarowana składowa klasy może być statyczna

LINQ

- Language integrated query
  - Typy Generyczne (generics)
  - Delegaty (delegates)
  - Metody rozszerzające (extension methods)
  - Wyrażenia lambda (lambda expressions)
  - Metody anonimowe (anonymous methods)
  - Typy anonimowe (anonymous types)

Metody rozszerzeń

- Zamiast dziedziczenia
  - Nie zmienia definicji samej klasy
  - Można stosować do klasy do której nie mamy dostępu
- this
  - Uwaga na referencje do zmiennej this

Delegaty

- Wskaźniki na funkcje
- delegate vs event
- Dostęp do zmiennych
- Delegaty zdefiniowane
  - Action, Action<>
  - Func<>

SOLID

- S – Single responsibility principle
- O – Open/Closed principle
- L – Liskov substitution principle
- I – Interface segregation principle
- D – dependency inversion principle

Praca z plikami

- System.IO (File, Path, Directory)
- Pliki tekstowe i binarne
- Strumienie danych (Stream)
  - FileStream, MemoryStream
- Odczytywanie z pliku
  - StreamReader
- Zapis do pliku
  - StreamWtriter
- Klauzula Using

Refleksja

- Dostęp do pakietów (assembly)
- Dostęp do klas i właściwości
- Late Binding

Atrybuty

- Klasa Attribute
- Atrybuty jako metadane klasy
- Wykorzystanie własnych atrybutów

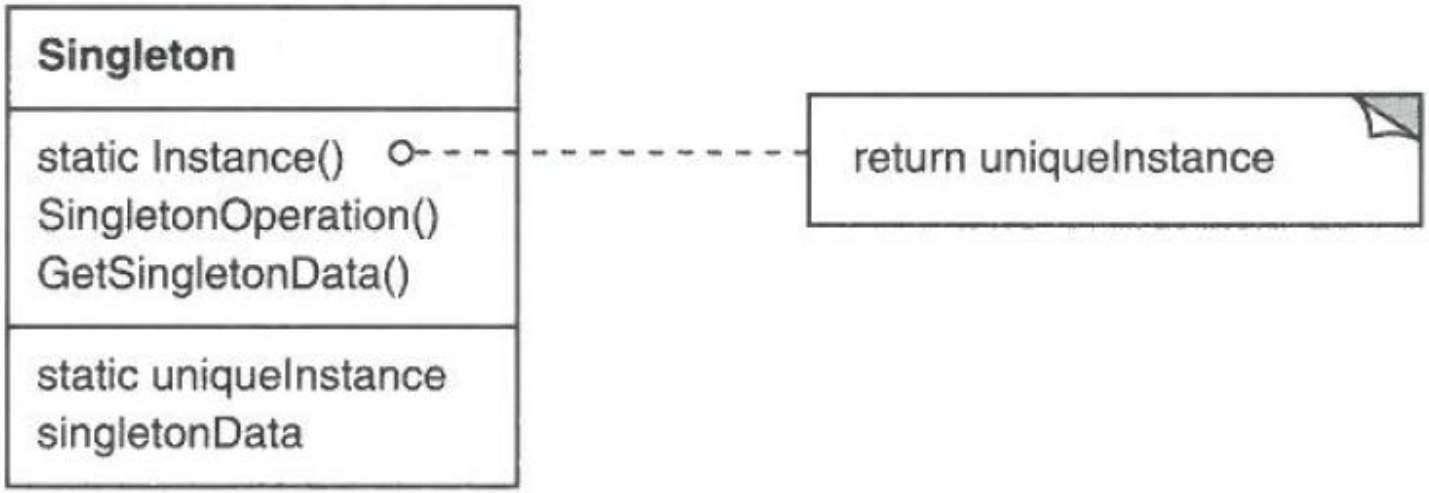
Testy jednostkowe

- Testy powinny testować pojedynczą funkcjonalność, jedna funkcjonalność może mieć wiele różnych testów
- Testy powinny być wykonywane w izolacji, bez zewnętrznej zależności. Zależności powinne być wstawiane za pomocą mocków lub stubów
- Testy powinny być powtarzalne i niezawodne
- Testy powinny być szybkie
- Testy powinny być łatwe, czytelne

Wzorce projektowe

- Kreacyjne
  - **Singleton, Fabryka Abstrakcyjna, Fabryka Metod**
- Strukturalne
  - **Fasada**
- Czynnościowe
  - **Obserwator**, Stan, **Strategia**, Polecenie, Iterator

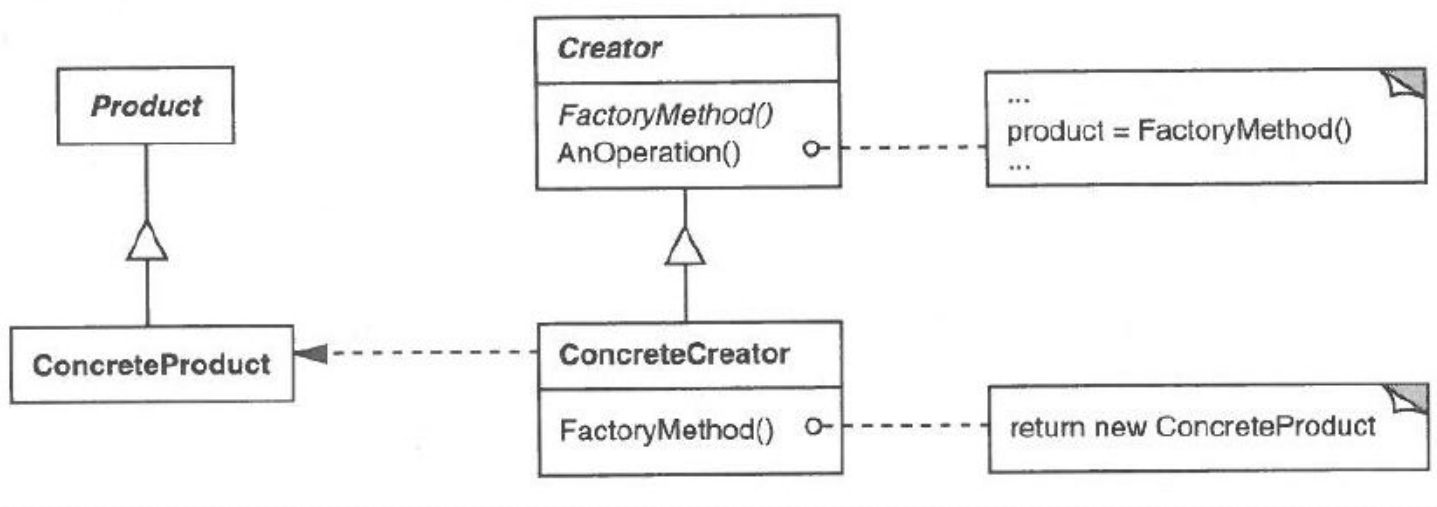
Singleton (Kreacyjny)



- Istnieje tylko jedna instancja klasy

Źródło: “Design Patterns, Elements of Reusable Object Oriented Software” E.Gamma, R. Helm

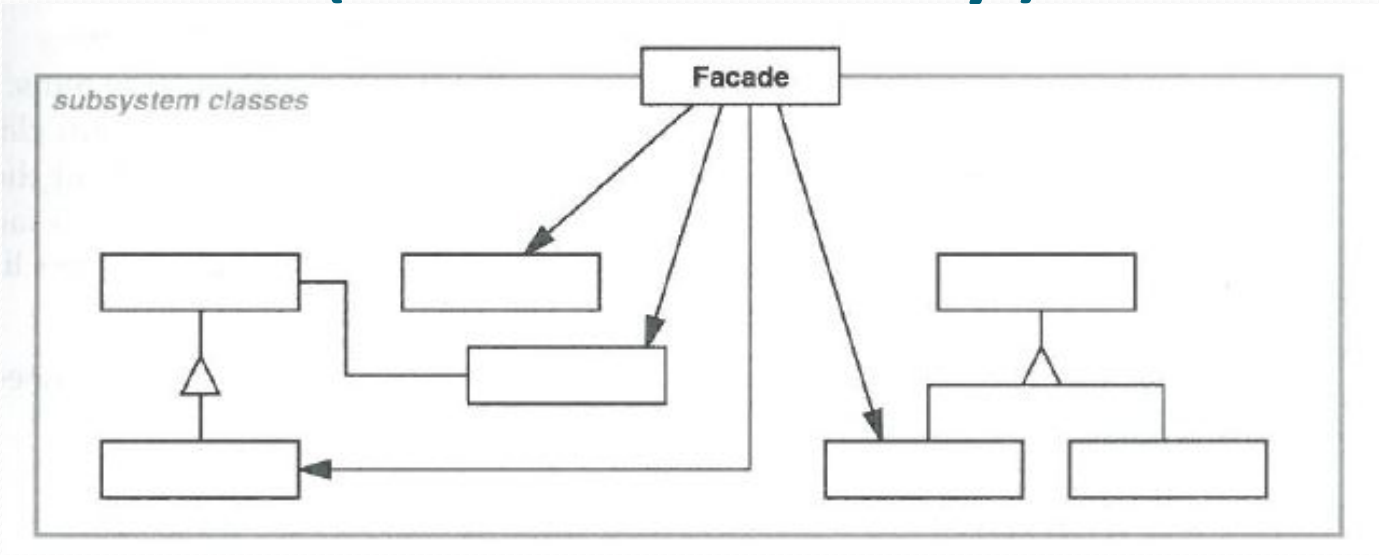
Fabryka metod (Kreacyjny)



- Określa interfejs, który zwracany jest przez metodę
- To metoda decyduje, którą implementację interfejsu zwrócić

Źródło: “Design Patterns, Elements of Reusable Object Oriented Software” E.Gamma, R. Helm

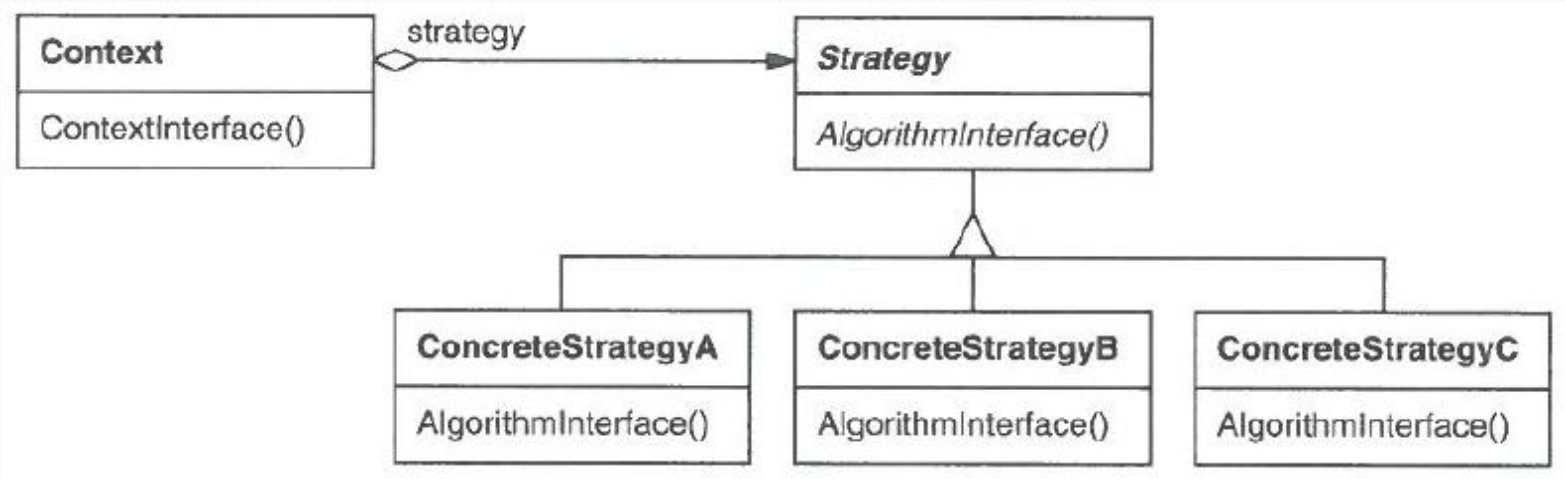
Fasada (Strukturalny)



- Fasada tworzy jeden interfejs do komunikacji ze środowiskiem zewnętrznym
- Interfejs składa się połączenia kilku klas i ich metod
- Zmniejsza skomplikowanie systemu dla odbiorcy

Źródło: “Design Patterns, Elements of Reusable Object Oriented Software” E.Gamma, R. Helm

Strategia (Czynościowy)



- Istnieje jeden interfejs wspólny dla wszystkich klas
- Jedna klasa (kontekst) używa metod tego interfejsu w swojej implementacji
- W zależności od potrzeb kontekst jest tworzony z wybraną implementacją interfejsu

Źródło: “Design Patterns, Elements of Reusable Object Oriented Software” E.Gamma, R. Helm

Wielowątkowość i asynchroniczność

- Programowanie równoległe
  - Rozdzielenie zadań *(robienie kilku rzeczy naraz na różnych wątkach)*
  - Wielowątkowość *(forma współbieżności)*
- Programowanie asynchroniczne
  - Obiekty typu *future (obietnica wykonania zadania w przyszłości – Task : z void, Task<> z wartością zwrotną)*
- *Programowanie reaktywne*
  - Reactive Extensions (Rx) *(programowanie na zdarzeniach)*
- Wady i zalety
- Kiedy NIE stosować

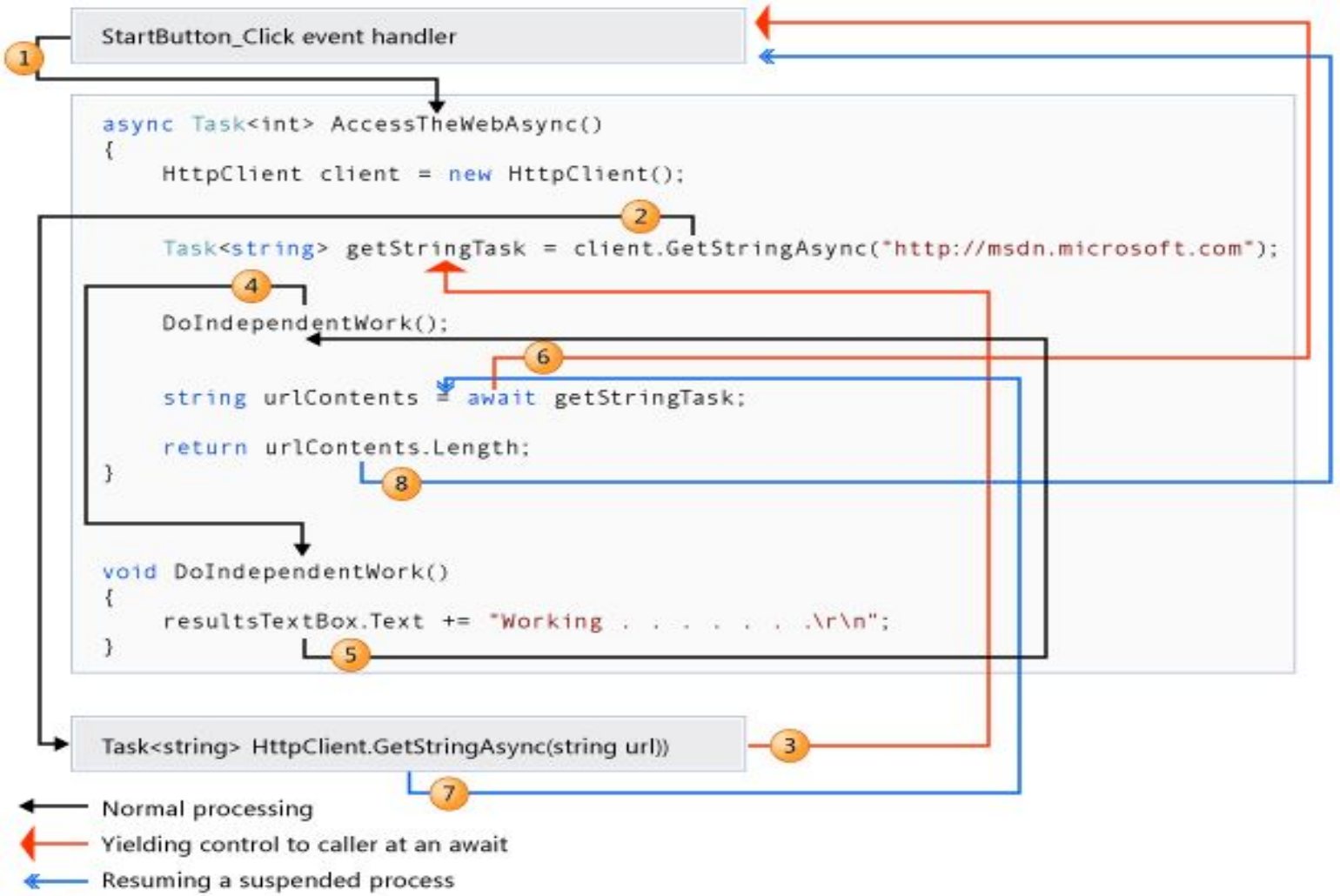
Wątki

- *Czym są wątki (wątek vs proces)*
- *Klasa System.Threading.Thread*
  - *Utworzenie, Uśpienie, Przerwanie*
- *Sekcje krytyczne*
- *Synchronizacja*
  - *Join/Abort/Sleep*
  - *ThreadPool*
  - *Interlocked*
  - *Monitor (lock)*
  - *Semafor*
  - *Bariera*

Zadania

- Wątek vs Zadanie
- Klasy *Task* i *Task<>*
  - *Wykorzystanie*
  - *Synchronizacja*
  - *async await*
- Fabryka zadań (*TaskFactory*)
- Stan zadań, przerywanie działania

async/await - przetwarzanie



Źródło: https://docs.microsoft.com/pl-pl/dotnet/csharp/programming-guide/concepts/async/

Drzewa wyrażeń - *dynamic*

- Wykorzystuje DLR (dynamic language runtime)
- Usprawnia pracę z obiektami COM, refleksją
- Słowo kluczowe dynamic zakres stosowania:
  - pole, właściwość, typ zwracany, zmienna lokalna
- Słowa kluczowego dynamic nie można używać z lambdami

Structured Language Query (SQL)

- Relational database
- Database components
  - Tables, views
  - Indexes
  - Stored procedures, triggers
- T-SQL

Typy danych - MsSQL

Data Category	Data Type	Size	Value Range
Exact numeric	Bit	1	1, 0, or NULL.
	Tinyint	1	0 to 255
	Smallint	2	-2^15 (-32,768) to 2^15-1 (32,767)
	Int	4	-2^31 (-2,147,483,648) to 2^31-1 (2,147,483,647)
	Bigint	8	-2^63 (-9,223,372,036,854,775,808) to 2^63-1 (9,223,372,036,854,775,807)
	Smallmoney	4	- 214,748.3648 to 214,748.3647
	Money	8	-922,337,203,685,477.5808 to 922,337,203,685,477.5807
	numeric[(p[,s])]	5-17	
	decimal [(p[,s])]	5-17	
Approximate numeric	Float	4-8	- 1.79E+308 to -2.23E-308, 0 and 2.23E-308 to 1.79E+308
	Real/float(24)	4	- 3.40E + 38 to -1.18E - 38, 0 and 1.18E - 38 to 3.40E + 38
Character strings	char [(N)]	N	N = 1 to 8000 non-Unicode characters bytes
	varchar [(N or max)]	N or 2^31-1	N = 1 to 8000 non-Unicode characters bytes Max = 2^31-1 bytes (2 GB) non-Unicode characters bytes
	Text	2^31-1	1 to 2^31-1 (2,147,483,647) non-Unicode characters bytes
Unicode character strings	nchar [(N)]	N	N = 1 to 4000 UNICODE UCS-2 bytes
	nvarchar [(N | max)]	N or 2^31-1	N = 1 to 4000 UNICODE UCS-2 bytes 1 to 2^31-1 (2,147,483,647) UNICODE UCS-2 bytes
	Ntext	2^30-1	Maximum size 2^30 - 1 (1,073,741,823) bytes
Binary strings	binary [(N)]	N	N = 1 to 8000 bytes
	varbinary [(N | max)]	N or 2^31-1	N = 1 to 8000 bytes Max = 0 to 2^31-1 bytes
	Image	2^31-1	0 to 2^31-1 (2,147,483,647) bytes
Other data types	Uniqueidentifier	16	xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx (hex decimal)
	Timestamp	8	binary(8) or varbinary(8)
	rowversion	8	binary(8) or varbinary(8)
	xml	2^31-1	xml([CONTENT | DOCUMENT] xml_schema_collection)
	sql_variant	8016	data type that stores values of various SQL Server-supported data types
	Hierarchyid	892	6*logA*n* bits where n is child node
	Cursor		
	Table		
	Sysname	256	
Date and time	Date	3	0001-01-01 through 9999-12-31
	time [(fractional second precision)]	3 to 5	00:00:00.0000000 through 23:59:59.9999999
	Smalldatetime	4	Date: 1900-01-01 through 2079-06-06 Time: 00:00:00 through 23:59:59
	Datetime	8	Date: January 1, 1753, through December 31, 9999 Time: 00:00:00 through 23:59:59.997
	datetime2 [(fractional seconds precision)]	6 to 8	Date: 0001-01-01 through 9999-12-31 Time: 00:00:00 through 23:59:59.9999999
	datetimeoffset [(fractional seconds precision)]	8 to 10	Date: 0001-01-01 through 9999-12-31 Time: 00:00:00 through 23:59:59.9999999 Time zone offset: -14:00 through +14:00
Spatial	Geography	2^31-1	
	Geometry	2^31-1	

Źródło: https://sqlserverrider.wordpress.com/2013/04/21/sql-server-data-type-cheat-sheet-sql-server-2012/

Operacje SQL

- **DQL** – SELECT
- **DML** – INSERT, UPDATE, DELETE
- **DDL** – CREATE, ALTER, DROP, TRUNCATE
- **DCL** – GRANT, DENY, REVOKE
- **TCL** – COMMIT, ROLLBACK, SET TRANSACTION

Operacja SELECT

- SELECT
- FROM
- JOIN
- WHERE
- GROUP BY
- HAVING
- ORDER BY

Entity Framework

- Object-relational Mapping (ORM)
- Setup models
  - Database first
  - Model first (.NET Framework)
  - Code first
- ContextDb, SetDb
- Lazy loading

WinForms

- Technologia
- User interface (UI)
- Kontrolki
  - zdarzenia
  - używanie wątków

LINQ to SQL

- Object-relational Mapping (ORM)
- Attributes
- Lazy loading

Podsumowanie