Platformy Technologiczne

Jarosław Kuchta

Porównanie języków C++, C#, Java

C++ vs. C# vs. Java (1)

- Idea:
  - C++: obiektowa wersja języka C (1985)
  - Java: język przenośny “Write Once, Run Anywhere” (1991)
  - C#: konkurencja dla języka Java (2000)
- Twórcy:
  - C++: Bjarne Stroustrup
  - Java: James Gosling, Mike Sheridan, Patrick Naughton (Sun Microsystems)
  - C#: Microsoft (Anders Hejlsberg, główny architekt Delphi)
- Standardy:
  - C++: ISO/IEC 14882:2017
  - Java: ISO/IEC TR 13066-6:2014
  - C#: ECMA-334, ISO/IEC 23270:2018
- Paradygmaty:
  - C++: proceduralny, funkcjonalny, obiektowy, ogólny, imperatywny
  - C#: strukturalny, funkcjonalny, obiektowy, ogólny, imperatywny, deklaratywny, refleksyjny, sterowany zdarzeniami, sterowany zadaniami, współbieżny
  - Java: structured, funkcjonalny, obiektowy, ogólny, imperatywny, refleksyjny, współbieżny

C++ vs. C# vs. Java (2)

- Kompatybilność:
  - C++:
    - wiele standardów: C++98, C++03, C++TR1, C++11, C++14, C++17
    - brak kompatybilności kodu z różnych kompilatorów
      - dekoracja nazw – rozwiązywanie problemów unikatowych identyfikatorów
      - różna obsługa wyjątków
  - C#:
    - niewielka kompatybilność źródłowa z C++ (tylko z C++/CLI)
    - kompatybilność z Common Language Infrastructure (C++/CLI, F#, VB.NET, Managed JScript, Windows PowerShell, …)
  - Java:
    - brak kompatybilności z innymi językami
- Przenośność:
  - C++: przenośne wraz z bibliotekami standardowymi
  - C#:
    - do działania w środowisku .NET Framework lub .NET Core
    - wiele frameworków .NET używających biblioteki natywne (C, C++)
    - Mono – wolna, otwarto-źródłowa, przenośna implementacja .NET
    - otwarcie na społeczność w 2010
  - Java:
    - duża przenośność (JVM)

C++ vs. C# typowanie

- C++:
  - silne typowanie
  - statyczne sprawdzanie typów
  - system typów oparty o nazwy
  - częściowe wnioskowanie typów
  - wskaźniki mogą zastępować typy
- C#:
  - silne typowanie
  - brak zmiennych i metod globalnych – tylko w klasach
  - typy wartościowe i referencyjne
  - **string** – typ referencyjny bez modyfikacji instancji (szybkie kopiowanie)
  - **array** – typ referencyjny (tylko tablice dynamiczne)
  - **boxing** i **unboxing** zmieniają wartości na referencje i odwrotnie
  - typy obiektowe z licznikiem referencji
  - niejawna deklaracja typów (**var**)
  - niejawne **tablice typowane**(new[])
  - większe bezpieczeństwo typów
    - jawna i niejawna **konwersja typów**
    - **poszerzanie** typów całkowitych
    - domyślnie **podwójna precyzja** operacji zmiennoprzecinkowych
    - typ **bool** niekompatybilny z **int**
    - **typy wyliczeniowe** niekompatybilne z **int** (oprócz 0)
  - **składowe wyliczeniowe** dostępne przez nazwę typu
  - **kowariancja** i **kontrawariancja** typów uogólnionych
  - typ **dynamic** dla współdziałania z innymi aplikacjami
  - nullowane typy wartościowe

C++ vs. C# (2)

- Metaprogramowanie
  - C++:
    - zaawansowany preprocessing
    - metaprogramowanie przez szablony
  - C#:
    - prosty preprocessing
    - atrybuty
    - typy uogólnione
- Dziedziczenie
  - C++:
    - dziedziczenie wielokrotne
    - rzutowanie typów w dół (dynamic_cast)
    - rzutowanie typów w górę (static_cast)
  - C#:
    - pojedyncze dziedziczenie
    - jawna i niejawna implementacja interfejsów

C++ vs. C# (2)

- Metody:
  - C++:
    - statyczne metody dla operacji klas (nie instancji)
    - metody wirtualne i abstrakcyjne
    - polimorfizm statyczny i dynamiczny
    - parametry przekazywane przez wartość, wskaźnik lub referencję
    - domyślne wartości parametrów
    - lista parametrów nieokreślonej długości
    - operatory nadpisywane w klasach
  - C#:
    - wszystkie funkcje muszą być zadeklarowane jako metody klas
    - statyczne metody w klasach statycznych zamiast funkcji globalnych
    - metody wirtualne i abstrakcyjne
    - parametry wejściowe, **ref** i **out**
    - parametry wejściowe przekazywane przez wartość lub referencję
    - domyślne wartości parametrów
    - **params** jako tablica reprezentująca pozostałe parametry
    - metody rozszerzeniowe – statyczne metody w klasie statycznej w parametrem **this**
    - silnie typowane referencje do funkcji (delegaty)
    - jawne i niejawne nadpisywane operatory konwersji
    - asynchroniczne wywoływanie metod
    - sekcje krytyczne przez instrukcję lock

Java vs. C#
Typy danych

	Java	C#
Typy wartościowe	Nie; tylko typy prymitywne	Tak
Typy referencyjne	Tak	Tak
Typy całkowite ze znakiem	Tak; 8, 16, 32, 64 bitowe	Tak; 8, 16, 32, 64 bitowe
Typy całkowite bez znaku	Nie; ale wsparcie dla niektórych operacji	Tak; 8, 16, 32, 64 bitowe
Typ dziesiętny o dowolnym rozmiarze	Typ referencyjny, bez operatorów	Dostarczane przez inne firmy
Typ całkowity o dowolnym rozmiarze	Typ referencyjny, bez operatorów	Tak
Tabele	Tak	Tak
Typ Boolean	Tak	Tak
Typ Character	Tak	Tak
Liczby zespolone	Dostarczane przez inne firmy	Dostarczane przez inne firmy
Typy daty/czasu	Tak; referencyjne	Tak; wartościowe
Typy wyliczeniowe	Tak; referencyjne	Tak; wartościowe
Obliczenia dziesiętne wysokiej precyzji	Patrz typ dziesiętny o dowolnym rozmiarze	Typ decimal 128-bit (28 digits)
Liczby zmiennoprzecinkowe IEEE 754 32-bitowe	Tak	Tak
Liczby zmiennoprzecinkowe IEEE 754 64-bitowe	Tak	Tak
Nullowane typy wartościowe	Nie	Tak
Wskaźniki	Nie	W kodzie niebezpiecznym
Łańcuchy znaków	Niezmienne typy referencyjne, Unicode	Niezmienne typy referencyjne, Unicode
Adnotacje / atrybuty typów	Tak	Tak
Drzewo typów z jednym typem nadrzędnym	Nie	Tak
Krotki	Dostarczane przez inne firmy	Tak

Java vs. C#
Typy wyliczeniowe

- Java:
  - typ wyliczeniowy jest klasą, a jego wartościami są obiekty (instancje) tej klasy
  - zbiory wyliczeń i kolekcje map zapewniają możliwość łączenia wielu wartości wyliczenia w połączoną wartość.
- C#:
  - Wyliczenia w C # są niejawnie wyprowadzane z typu Enum (typ wartości).
  - Zestaw wartości wyliczenia C # jest definiowany przez typ bazowy, który może być typem liczby całkowitej ze znakiem lub bez znaku o 8, 16, 32 lub 64 bitach.
  - Definicja wyliczenia definiuje nazwy wybranych wartości całkowitych.
  - Wyliczenia mapowane bitowo, w których rzeczywista wartość może być kombinacją wartości wyliczonych bitowo lub połączonych razem.

Java vs. C#
Typy referencyjne

	Java	C#
Garbage collection	Tak	Tak
Słabe referencje (Weak references)	Tak	Tak
Kolejkowanie referencji (przez GC)	Tak	Tak
Referencje przesuwalne	Tak	Tak
Phantom references	Tak	Nie
Klasy proxy	Tak	Tak

Java vs. C#
Tablice i kolekcje

	Java	C#
Uogólnione typy danych	Tak	Tak
Jednowymiarowe, indeksowane od zera	Tak	Tak
Wielowymiarowe, prostokątne	Nie	Tak
Tablice tablic	Tak	Tak
Jednowymiarowe, indeksowane nie od zera	Nie	Niekiedy
Unifikacja tablic i kolekcji	Nie	Tak
Mapy / słowniki	Tak	Tak
Słowniki sortowane	Tak	Tak
Zbiory	Tak	Tak
Zbiory sortowane	Tak	Tak
Listy / wektory	Tak	Tak
Kolejki / stosy	Tak	Tak
Kolejki priorytetowe	Tak	Tak
Zbiory z powtarzającymi się elementami	Dostarczane przez inne firmy	Tak
Kolekcje optymalizowane dla współbieżności	Tak	Tak

Specyficzne cechy C#

- **C#:**
  - **właściwości** (get & set)
  - **zdarzenia** (add & remove)
  - metody **async**

C# jako open-source

- 2010: .NET Compiler Platform ("Roslyn")
- 2014: ASP.NET
- 2014: .NET Core (Windows, Linux, macOS, bez C++/CLI)
- 2016: Mono (Windows, macOS, Linux) -> .NET Foundation
- 2016: Xamarin (Android, iOS, Windows)
- 2016: Microsoft Build (licencja MIT)