Zuzanna Krawczyk

Źródło grafiki: http://www.scala-lang.org/sites/default/files/newsflash_logo.png

Hello Scala!

```
println("Hello, World!")
```

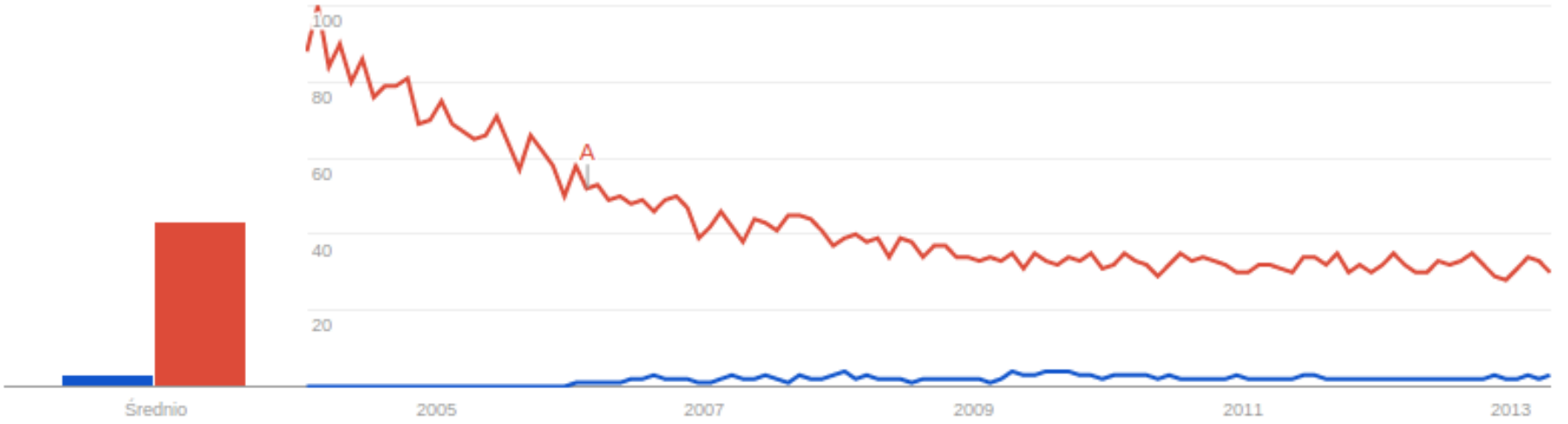
```
public class HelloWorld {

  public static void main(String[] args)
  {   System.out.println("Hello world!");
  }
}
```

Scala w punktach

- Statycznie typowana
- Działa na JVM/.NET
- Obiektowy + funkcjonalny + imperatywny paradygmat programowania
- Naturalna integracja z kodem Javy
- Jest Javą przyszłości?

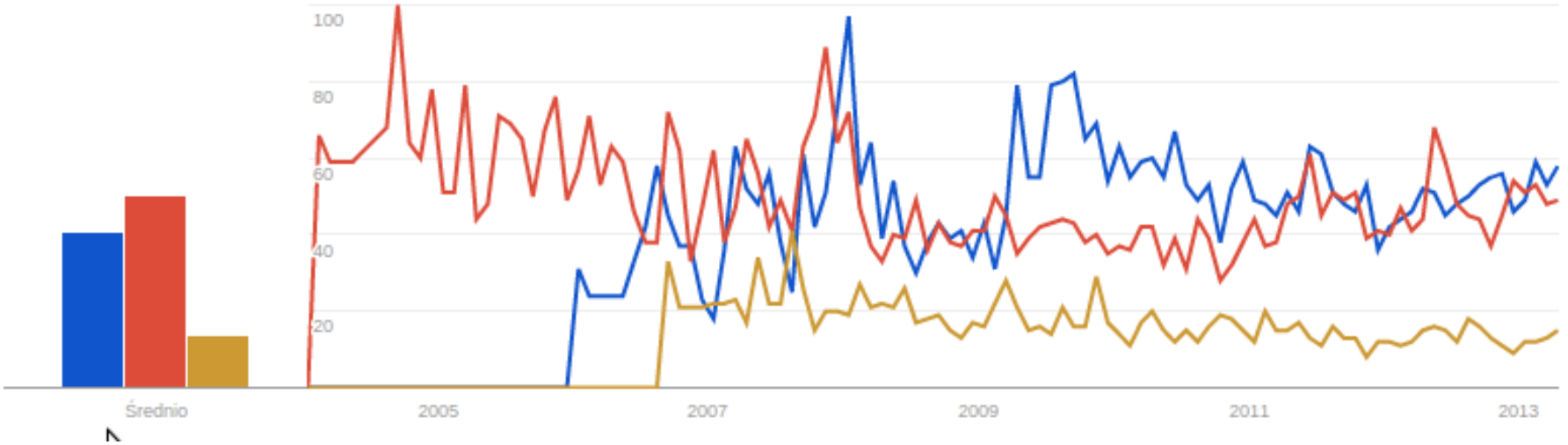
Porównanie liczby wyszukań dla haseł „*Scala language*”, „*Java language*” - Google Trends



Java

Scala

Scala, Lua, Erlang – Google Trends



Scala

Lua

Erlang

Filozofia

- Łatwość dopasowania się (skalowania :)) do dużych i małych problemów
  - Język skryptowy
  - Wsparcie dla elementów funkcyjnych – elegancki, zwięzły zapis problemów
  - Elementy obiektowe takie jak klasy , cechy, typy generyczne, polimorfizm itd. - możliwość przejrzystego tworzenia złożonych systemów

Słowa kluczowe – stałe, zmienne, funkcje ...

- Val – ewaluowana w momencie definicji, wartość nie może zostać zmieniona po przypisaniu
- Var – ewaluowana w momencie definicji, wartość zmiennej może być ponownie przypisywana
- Def – wartość jest obliczana, za każdym razem gdy jest odczytywana (definicje funkcji)
- Lazy val – wartość jest obliczana tylko w momencie pierwszego odczytu

Inferencja typów (*type inference*)

Wykorzystanie możliwości automatycznego wykrycia typów przez kompilator, na podstawie typu wyrażenia np.

```
def max(x: Int, y: Int):Int = if (x > y) x else y
```

W Scali wszystko jest obiektem

- Brak typów prymitywnych (int, double..) :
  - 2 + 3 <=> (2).+(3)
- Nawet funkcje :)
  - Przekazywanie funkcji jako argumentów
  - Zapisywanie funkcji do zmiennych
  - Zwracanie funkcji z poziomu innych funkcji
  - Funkcje anonimowe

Klasy przypadków i dopasowanie wzorców

- Klasy przypadków (*case classes*)
  - Przydatne w przypadku „drzewiastych” struktur danych
  - Wspierają dopasowanie wzorców dla obiektów
  - Tworzenie instancji klasy bez specyfikatora new
  - Wszystkie parametry klasy oznaczone jako val (dostęp jak do pól klasy)
  - ToString, hashCode, equals

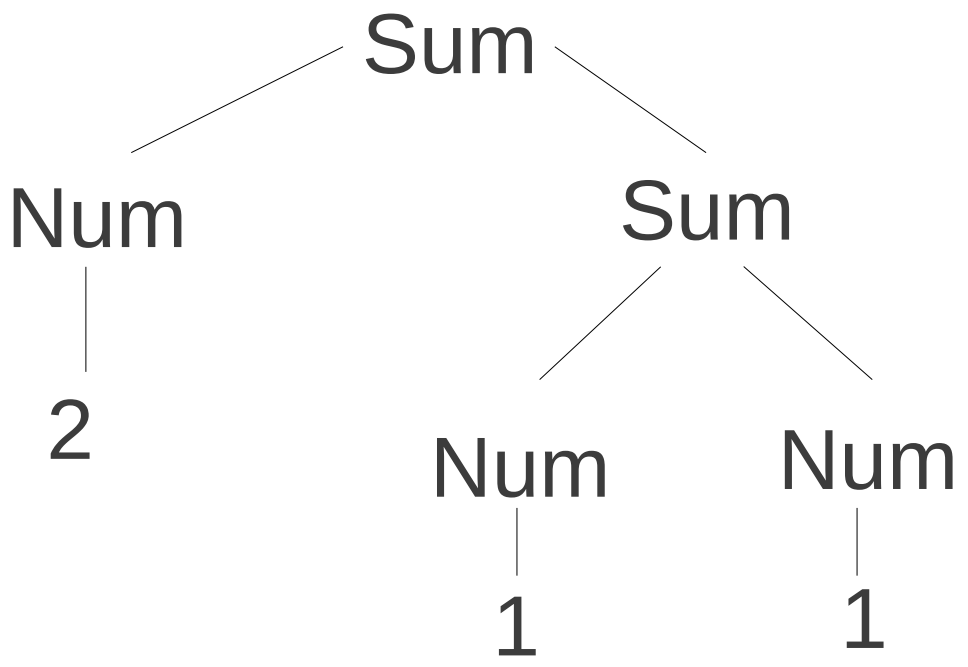
Dopasowywanie wzorców (pattern maching)

- Uogólnienie konstrukcji *switch*
- Metoda *match*
- *Match* przyjmuje zbiór klas „*case*”
- Szuka wśród nich dopasowania do wzorca
- Po znalezieniu wzorca wykonuje prawą stronę wyrażenia
- Ułatwia dodawanie nowych operacji wykonywanych na danej strukturze

Dopasowywanie wzorców - przykład

- Chcemy sparsować proste wyrażenie:

  2 + (1 + 1)



Listy i co można z nimi zrobić

- Lista jest podstawowym typem używanym w programowaniu funkcyjnym
- Raz wygenerowana lista nie może zostać zmieniona (charakterystyczne dla języków funkcyjnych)
- Scala posiada szereg metod wspierających obsługę list
- Lista składa się z głowy i reszty

Krotki (Tuples)

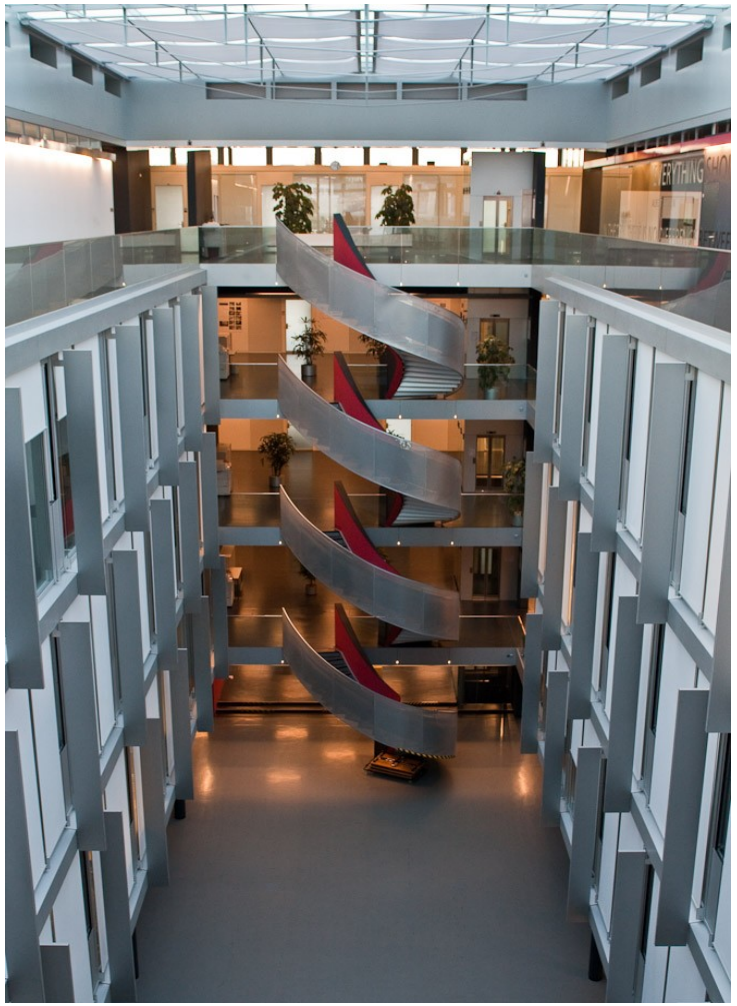
- Podobne do list – nie mogą zostać zmienione
- Jedna krotka może przechowywać różne typy danych
- Przykładowe wykorzystanie: zwracanie kilku wartości z funkcji

Cechy (Traits)

- Mogą zawierać implementacje metod, definicje pól
- Klasa może korzystać (*mix in*) z wielu cech naraz

Aktorzy

- Java → model współbieżności oparty na pamięci współdzielonej oraz zamkach
  - Zakleszczenia
  - Zjawisko hazardu
  - Trudności w testowaniu kodu
- Scala → biblioteka *Actors,* realizująca model przekazywania komunikatów



Autor: Miles Sabin, źródło: http://www.flickr.com/photos/montpelier/3957416434/

Źródło: http://www.scala-lang.org/sites/default/files/newsflash_logo.png

Schody – (wł.) scala