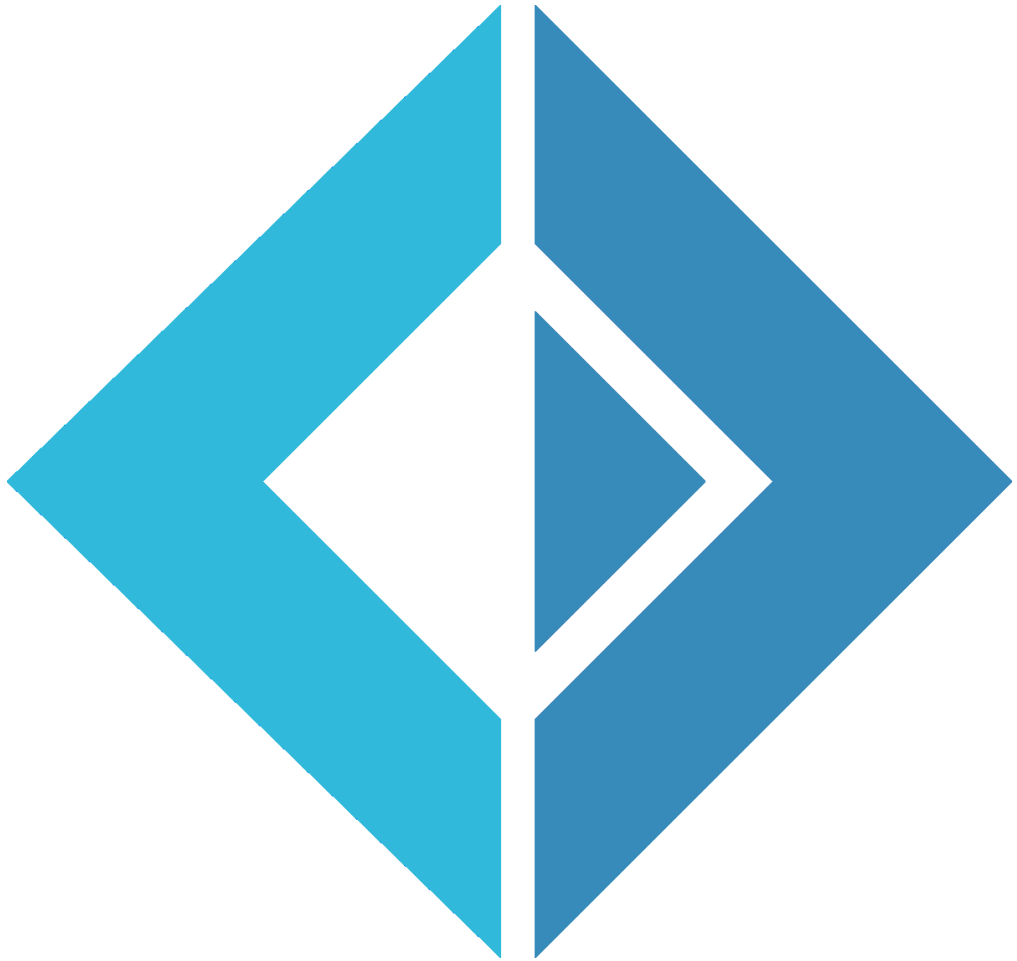
PV178 – Lab 13

Karel Jiránek

Agenda

- F#

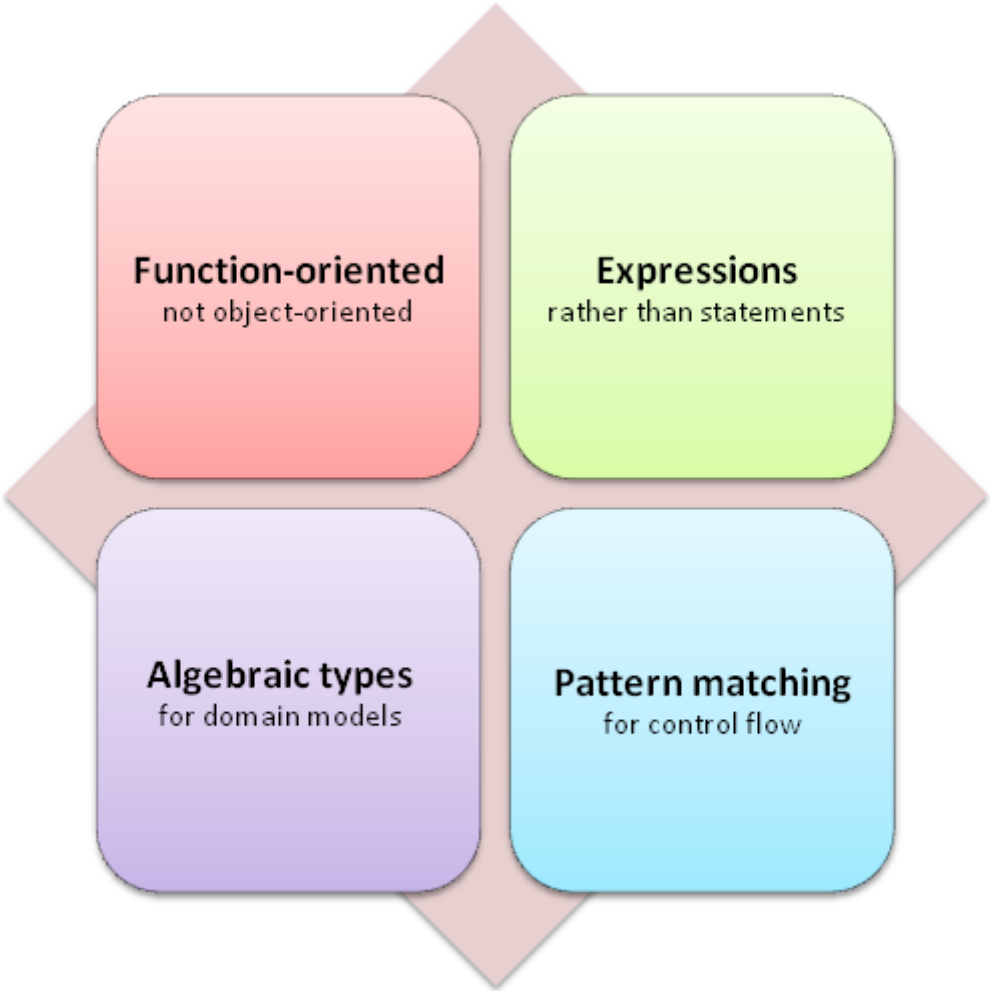


Obecně

- Vyvíjený od 2005 (dnes ve verzi 6.0)
- Open source
- Silně typovaný
- Součást .NET světa
- Spojuje více paradigmat – spojuje OOP principy a funkcionální programování
- „Funkce první“ (first class citizen)
- Neexistuje zde `null`
- Vestavěná immutabilita
- Striktní odsazování příkazů (podobně jak Python)

Principy

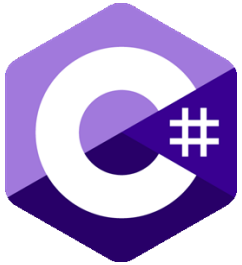
- 4 základní principy



Orientace na funkce

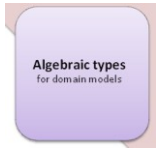
```
let square x = x * x

let result = [1..10] |> List.map square;

let execFunction aFunc aParam = aFunc aParam
let result2 = execFunction square
```

Výrazy



```
string firstArg = null;
string secondArg = null;
if (args.Length > 1)
{
    firstArg = args[0];
    secondArg = args[1];
}
```

```
let firstArg, secondArg =
        if argv.Length > 1 then
            argv.[0], argv.[1]
        else
            "", ""
```



Algebraické typy

- **Produkty** typů

```
type IntAndBool = {intPart: int; boolPart: bool}
let x = {intPart = 1; boolPart = false}
```

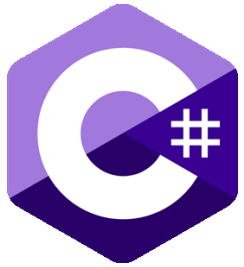
- **Součty** typů

```
type IntOrBool =
        | IntChoice of int
        | BoolChoice of bool
let y = IntChoice 42
let z = boolChoice true
```

Pattern matching jako řízení toku kódu

```
type Color =
        | Red = 0
        | Green = 1
        | Blue = 2

let printColorName (color:Color) =
        match color with
        | Color.Red -> printfn "Red"
        | Color.Green -> printfn "Green"
        | Color.Blue -> printfn "Blue"
        | _ -> ()

printColorName Color.Red
printColorName Color.Green
printColorName Color.Blue
```

C# vs F#



VS

VS

Piping

```
// Definice (definuje přímo F#)
let (|>) x f = f x


// Použití
let items = [0..10]
let squaresBiggerThan3 =
     items
     |> Seq.map (fun x -> x * x)
     |> Seq.filter (fun x -> x > 3)
```

Compose

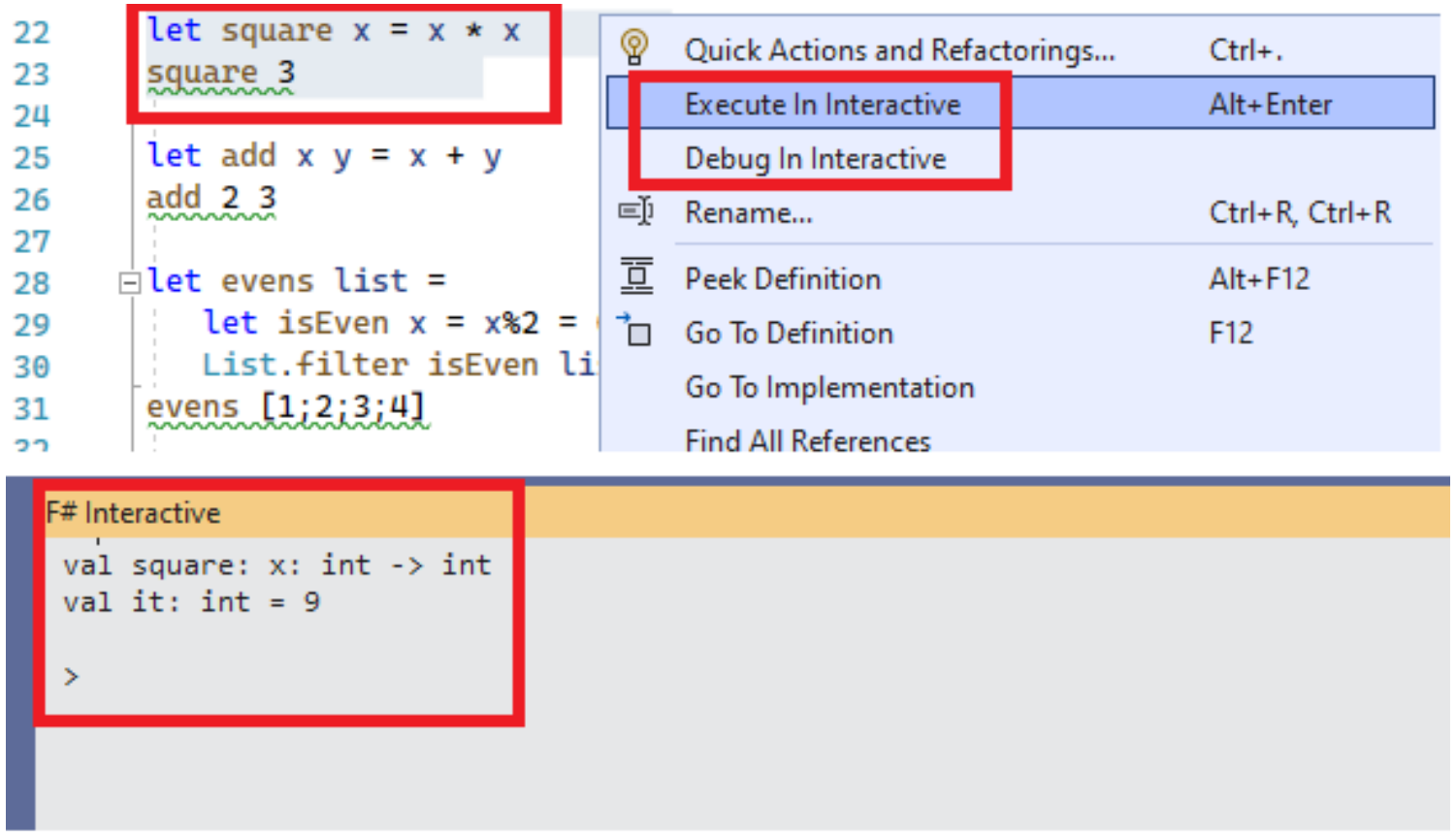
```fsharp
// Definice (definuje přímo F#)
let (>>) f g x = g (f x)


// Použití
let length (s: string) = s.Length
let makeString x = new string('a', x)

let composedFunc = length >> makeString

composedFunc "1234" // Výsledek je "aaaa"
```

Výhody F#

- Units of measure – validace jednotek při kompilaci (např.: fyzikálních)
- Interaktivní mód – podobný interaktivnímu modu v Pythonu
- Kód je dobře testovatelný – díky kódu bez vedlejších efektů (díky immutabilitě)
- Interoperabilita – snadné využití přímo v C# (a obráceně)
- Méně chyb – z podstaty funkcionálního programování
- Pattern matching
- **REPL**

Výhody F# - REPL

- „ Read-Eval-Print-Loop“



Příklady

- Syntax.fs – syntaxe jazyka
- Types.fs – typy (typ v F# ≅ třída v C#)
- Piping.fs – příklad pipingu
- Compose.fs – příklad skládání funkcí
- CSharpInterop – příklad použití knihovny ze C#

Samostatná práce

Postup 1

1. Otevřte soubor `IndividualWork`,
2. Změňte typ tak aby dědil od `IndividualWorkBase`
3. Implementujte, za pomocí poděděných metod, ve vytvořené třídě public metody

3.1 `kibiBytesToKiloBytes`

3.2 `kibiBytesToBytes`

3.3 `kibiBytesToBits`

4. Implementujte metody

4.1 `getBestStudent` – vybere z kolekce studentů studenta s nejvíce body

4.2 `printPersonsWithJob` – vytisikne kolekci lidí (persons), u každé osoby vypíše jestli se jedná o učitele nebo studenta (využijte pattern matching)

!!! Pro testování odkomentujte metody v souboru Program.fs !!!

Nápověda

- Pro implementaci `getBestStudent` využijte Seq.MaxBy
- Využijte příklady ze `Syntax.fs`
- Pro implementaci `printPersonsWithJob` využijte pattern matching
- Pro implementaci `printPersonsWithJob` využijte operátor `:?`

!!! Pro testování odkomentujte metody v souboru Program.fs !!!

Zdroje

- https://www.tutorialspoint.com/fsharp
- https://fsharpforfunandprofit.com/posts/key-concepts/
- https://docs.microsoft.com/en-us/dotnet/fsharp/language-reference/units-of-measure
- https://docs.microsoft.com/en-us/dotnet/fsharp/tools/fsharp-interactive/
- https://www.codemag.com/article/1605061