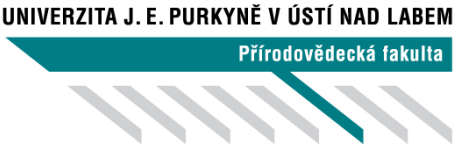
Univerzita Jana Evangelisty Purkyně
v Ústí nad Labem
Přírodovědecká fakulta



Tacit programming - návrh doménově specifického jazyka a implementace jeho interpretu

BAKALÁŘSKÁ PRÁCE

Vypracoval: Oleg Musijenko

Vedoucí práce: Mgr. Jiří Fišer, Ph.D.

Studijní program: Aplikovaná informatika

Studijní obor: Informační systémy

Ústí nad Labem 2022

Zde bude vloženo zadání bakalářské práce!!

Zadání bakalářské je možné získat až v okamžiku, kdy je práce schválena ve STAGu vedoucím práce, vedoucím katedry a garantem oboru a nelze tak učinit elektronicky.

Požádejte o něj vedoucího katedry informatiky. Zadání vám bude doporučeno v podobě elektronicky podepsaného PDF, které vložíte na místo toho listu (dvojstránky) některým z nástrojů pro práci s PDF dokumenty.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a použil jen pramenů, které cituji a uvádím v přiloženém seznamu literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., ve znění zákona č. 81/2005 Sb., autorský zákon, zejména se skutečností, že Univerzita Jana Evangelisty Purkyně v Ústí nad Labem má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona,s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Jana Evangelisty Purkyně v Ústí nad Labem oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

V Ústí nad Labem dne 18. ledna 2022 Podpis:

Děkuji vedoucímu práce Mgr. Jiřímu Fišerovi, Ph.D.
za neocenitelné rady a pomoc při tvorbě bakalářské práce.

Abstrakt

Tacit programming - návrh doménově specifického jazyka a implementace jeho interpretu

Abstrakt shrnuje základní motivaci práce (kontext), hlavní cíl a následně jednotlivé autorské kroky k jeho splnění (co bylo uděláno od úvodních rešerší, přes návrh, implementaci k případnému nasazení. Minimální rozsah je 800 znaků (maximální půl strany).

Klíčová slova

seznam klíčových slov (obecných termínů vystihujících téma práce) v počtu dva až deset

---

Abstract

Tacit programming - design of a domain specific language and implementation of it's interpreter

Translation of Czech abstract.

Key words

Translation of czech key words.

Obsah

Úvod

1 Tacit programming

Tacit programming je styl programování, kde nevyužíváme parametry funkcí, ale funkce řetězíme, nebo kompozicujeme. Ukažme si jednoduchý a intuitivní příklad v javascriptu.

```
fetch("APIURL")
.then(x => fancyFunction(x))
.then(x => console.log(x))
.catch(e => throw Error(e))
```

Po získání dat provedeme transformaci dat (definice fancyFunction není v tomto kontextu důležitá, stačí pouze vědět, že vrací Promise/Slib) a poté zapíšeme výsledek do Konzole. Toto je jeden ze způsobů, jak můžeme na sebe řetězit funkce zpětného volání (”Callbacks”).

Toto je zcela běžná praxe javascript programátorů, ale bohužel má jednu malou nevýhodu. Tvoří se zde zbytečná anonymní funkce (”arrow function nebo-li šipková”) a pokud bychom prohlubovali čím dál víc zásobník volání, mohou nám tyto anonymní funkce zabírat paměť.

```
fetch("APIURL")
.then(fancyFunction)
.then(console.log)
.catch(throw Error)
```

Zde se nachází kód, který dělá stejné instrukce, jako ten předchozí. Rozdíl je ten, že je zapsán jako *beztečkový ”point free”*. Takto programátor předchází potřebě dělat wrappovací funkce.

Na následujícím přikladě si ukážeme, jak funguje **currying** a proč souvisí s tacit programováním.

```
const curry = (f) => a => b => f(a,b);

const sayHello = (a, b) = `Hello ${a} from ${b}`;

const applyToFunctionArray = (input,...args) => args.map(a => a(input))

const partiallyAppliedData = ["A", "B", "C"].map(curry(sayHello));
// [(b) => "Hello A from ${b}",
//  (b) => "Hello B from ${b}",
//  (b) => "Hello C from ${b}"]
```

```
const partiallyAppliedData2 = ["A", "B", "C"].map(curry(sayHello)(1));
// ["Hello A from 1",
//  "Hello B from 1",
//  "Hello C from 1"]
```

Curry funkce zařizuje, že máme pro každý argument vlastní funkci. V čem je toto výhodné? Například je zde uvedené pole, které se skládá z částečně aplikovaných funkcí. Takto může programátor u předchozí ukázky naiterovat odpověď ze serveru do objektu, které je závislé na třeba na uživatelském vstupu.

Zajímavější část je u *partiallyAppliedData2*.

1.1 Principy a odlišnosti od klasického procedurálního paradigmatu

1.2 Rešerše existujících implementací

2 DSL - principy a využití

3 Návrh vlastního DSL

4 Implementace interpretu navrženého DSL

5 Ověření použitelnosti (testování funkčnosti, praktické příklady využití)

6 Závěr

7 Citace

Katuščák, Drobíková a Papík c2008 Lattner 2008

Bibliografie

Katuščák, Dušan, Barbora Drobíková a Richard Papík (c2008). *Jak psát závěrečné a kvalifikační práce. jak psát bakalářské práce, diplomové práce, dizertační práce, specializační práce, habilitační práce, seminární a ročníkové práce, práce studentské vědecké a odborné činnosti, jak vytvořit bibliografické citace a odkazy a citovat tradiční a elektronické dokumenty*. Nitra: Enigma. ISBN: 978-80-89132-70-6.

Lattner, Chris (2008). *Intro to LLVM*. Erice, Sicily: Chris Lattner. URL: https://llvm.org/pubs/2008-10-04-ACAT-LLVM-Intro.pdf.