Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Semestrální práce ITEJA a ICSHP

Nikola Jačková

2021

**Obsah**

[Seznam obrázků 3](#_Toc68546364)

[Úvod 4](#_Toc68546365)

[1 Struktura aplikace 5](#_Toc68546366)

[1.1 LanguageLibrary 5](#_Toc68546367)

[1.1.1 Exceptions 5](#_Toc68546368)

[1.1.2 Lexer 5](#_Toc68546369)

[1.1.3 Parser 6](#_Toc68546370)

[1.1.4 Interpreter 7](#_Toc68546371)

[2 GUI 9](#_Toc68546372)

[2.1 InterpretGUI 9](#_Toc68546373)

[2.2 GrammarPopUp 10](#_Toc68546374)

[2.3 Console 10](#_Toc68546375)

[Závěr 12](#_Toc68546376)

[Přílohy 13](#_Toc68546377)

Seznam obrázků

[Obrázek 1: Třídy Expressions a Conditions 6](#_Toc68538661)

[Obrázek 2:Třídy Statements a ostatní 7](#_Toc68538662)

[Obrázek 3: Hlavní okno aplikace 9](#_Toc68538663)

[Obrázek 4: Formulář Grammar 10](#_Toc68538664)

Úvod

Tato semestrální práce vytváří interpret jednoduchého fiktivního programovacího jazyka dle vytvořené gramatiky. Aplikace nabízí grafické uživatelské rozhraní s možností zobrazení grafického výstupu zdrojového kódu v podobě želví grafiky. Aplikace byla naprogramována v jazyce C# s využitím Windows Forms a .NET Frameworku ve verzi 4.8.

# Struktura aplikace

Aplikace s názvem ITEJA\_ICSHP\_Jačková\_Nikola je rozdělena do dvou projektů – LanguageLibrary a GUI.

## LanguageLibrary

LanguageLibrary je knihovna, která obsahuje čtyři základní složky – Exceptions, Interpreter, Lexer, Parser. Každá ze složek obsahuje několik zdrojových souborů nebo dalších složek. Obsahuje třídy a rozhraní pro realizaci lexikální analýzy, sémantické analýzy a interpretaci fiktivního jazyka. Jedná se o jednoduchý jazyk, který nabízí využití základních konstrukcí – for a while cyklus, podmínku if (else), přiřazování hodnot do proměnných (textový řetězec – STRING nebo číslo – NUMBER) a využití šesti metod Print, Forward, Rotate, Backward, PenVisible a ChangePen.

Metoda Print vypíše zadaný text do konzole. Přijímá nula až nekonečno parametrů obou typů. Metoda Forward posune želví kurzor dopředu, přijímá jeden parametr typu NUMBER. Metoda Backward posune želví kurzor dozadu, přijímá jeden parametr typu NUMBER. Metoda Rotate otočí želví kurzor, přijímá jeden parametr typu NUMBER. Metoda PenVisible určuje, jestli je pero aktivní (kreslí) nebo ne, přijímá jeden parametr typu NUMBER. Metoda ChangePen určuje barvu a tloušťku nakreslené čáry, přijímá dva parametry, první typu STRING a druhý typu NUMBER.

### Exceptions

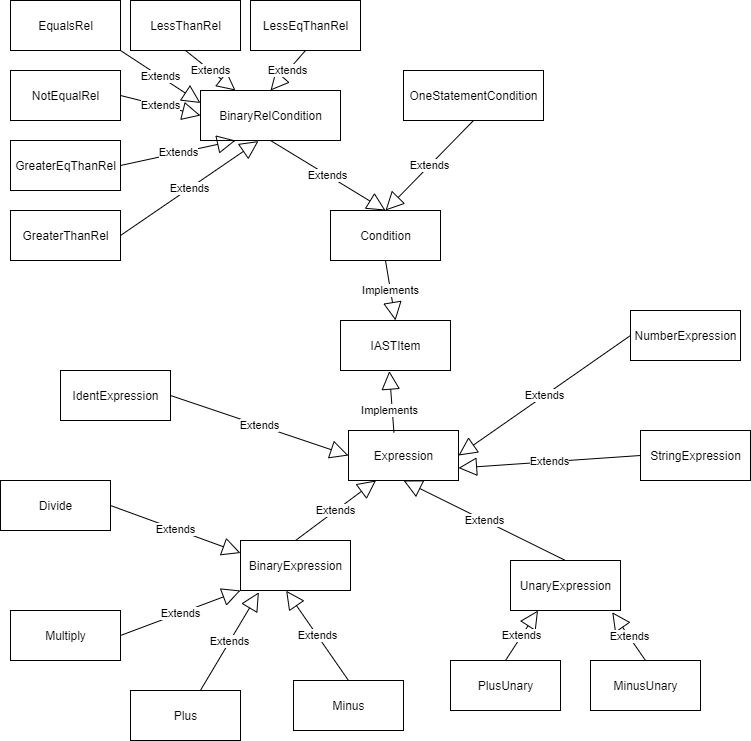
Tato složka obsahuje čtyři zdrojové soubory – InterpretException.cs, LanguageException.cs, LexerException.cs a ParserException.cs. Předkem pro zbylé tři třídy je třída LanguageException, která dědí ze třídy Exception. Tyto čtyři třídy složí pro vyvolávání vlastních výjimek.

### Lexer

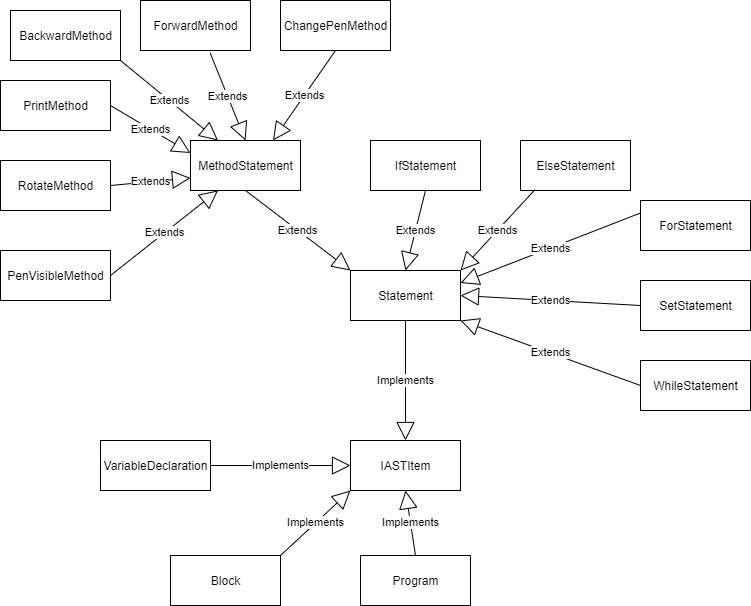
Třídy ve složce Lexer zajišťují lexikální analýzu, tj. převod vstupního textu na jednotlivé tokeny. Obsahuje další složku – Tokens, která obsahuje třídu Token a výčet TokenType. Instance třídy Token reprezentují jednotlivé tokeny a výčet TokenType udává každému tokenu předem definovaný typ např. hodnotu KEY. Ve složce Lexer se dále nachází dvě třídy – Lexer a LexerEngine. Interní LexerEngine zajišťuje samotnou lexikální analýzu a veřejná třída Lexer pak nabízí veřejné metody potřebné pro další zpracování vstupního textu – GetNextToken(), ResetEnumerator() a TokensToString(). Metoda GetNextToken() vrací objekt typu Token v pořadí, metoda ResetEnumerator() zajišťuje resetování enumerátoru, který slouží pro iteraci prvků v lexeru a metoda TokensToString() vrací textový řetězec všech tokenů. Pro vytvoření objektu je nutné předat parametr typu string.

### Parser

Třídy ve složce Parser zajišťují syntaktickou analýzu, tzn. proces analýzy posloupnosti tokenů s cílem určit jejich gramatickou strukturu vůči předem dané formální gramatice. Této analýze předchází lexikální analýza. Ve složce Parser se nachází další čtyři složky – Conditions, Expressions, Statements, Variables. Složka Parser dále obsahuje třídy Block, Parser, ParserEngine, Program a rozhraní IASTItem. Všechny třídy složky Parser a jejich podsložek podrobněji zobrazují obrázky 1 a 2.



Obrázek : Třídy Expressions a Conditions



Obrázek :Třídy Statements a ostatní

Ve složce Parser se nachází rozhraní IASTItem. IASTItem je předkem všech dílčích tříd syntaktického stromu a předepisuje metodu Accept(IVisitor visitor). Pro vytvoření instance třídy Parser je potřeba předat parametr typu Lexer.

### Interpreter

Složka Interpreter obsahuje šest položek – ExecutionContext, InterpretEngine, Interpreter, Variable, Variables a rozhraní IVisitor.

Třída Interpret obsahuje atribut typu InterpretEngine a atribut typu Parser. Pro vytvoření instance třídy Interpret je nutné předat parametr typu string. Hlavní metodou je veřejná metoda Interpret.

Třída InterpretEngine pak zajišťuje pak samotnou interpretaci předaného zdrojového kódu. Tato třída využívá delegátů pro jednotlivé metody, které jsou k dispozici. Třída obsahuje atribut typu Stack<ExecutionContext>, který slouží jako zásobník proměnných. Tato třída implementuje rozhraní IVisitor.

Objekty třídy Variable představují proměnné, které mají konkrétní typ. Třída Variables pak obsahuje atribut typu Dictionary<string, Variable> a nabízí metody pro práci s tímto atributem.

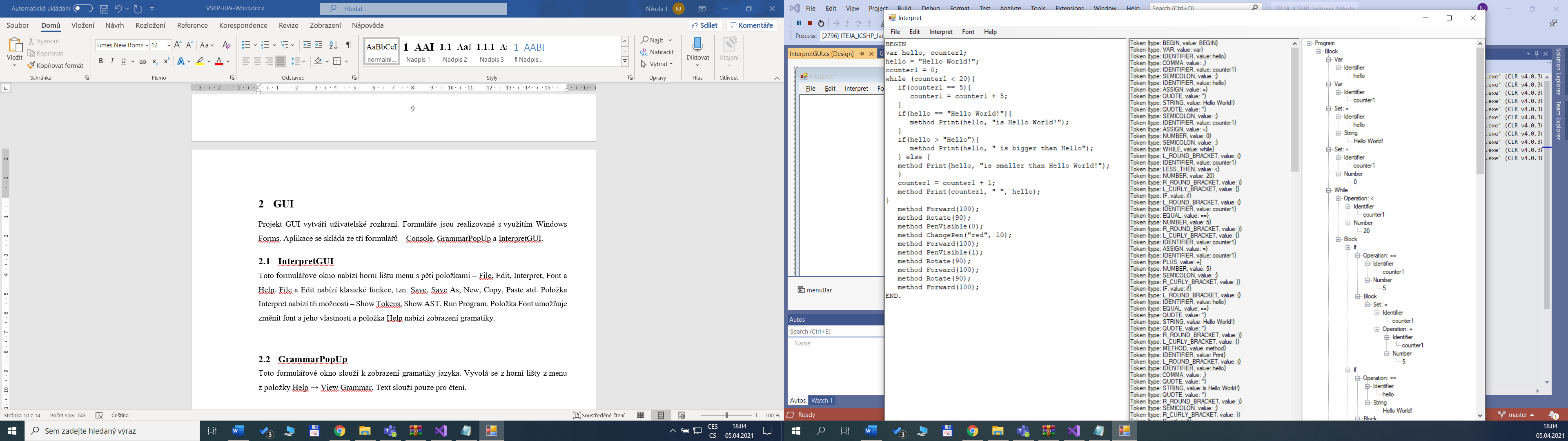
Rozhraní IVisitor je realizované dle návrhového vzoru Visitor. Pro každý typ položky syntaktického stromu určuje metodu. Třídy, které implementují toto rozhraní, musí v daných metodách určit, co se má vykonat při zpracování dané položky syntaktického stromu.

# GUI

Projekt GUI vytváří uživatelské rozhraní. Formuláře jsou realizované s využitím Windows Forms. Aplikace se skládá ze tří formulářů – Console, GrammarPopUp a InterpretGUI. Dále obsahuje čtyři třídy – LanguageLibryEngine, Program, TreeViewBuilder a Turtle.

## InterpretGUI

Toto formulářové okno nabízí horní lištu menu s pěti položkami – File, Edit, Interpret, Font a Help. File a Edit nabízí klasické funkce, tzn. Save, Save As, New, Copy, Paste atd. pro editaci textu v levém panelu. Položka Interpret nabízí tři možnosti – Show Tokens, Show AST, Run Program. Položka Font umožňuje změnit font a jeho vlastnosti a položka Help nabízí zobrazení gramatiky.

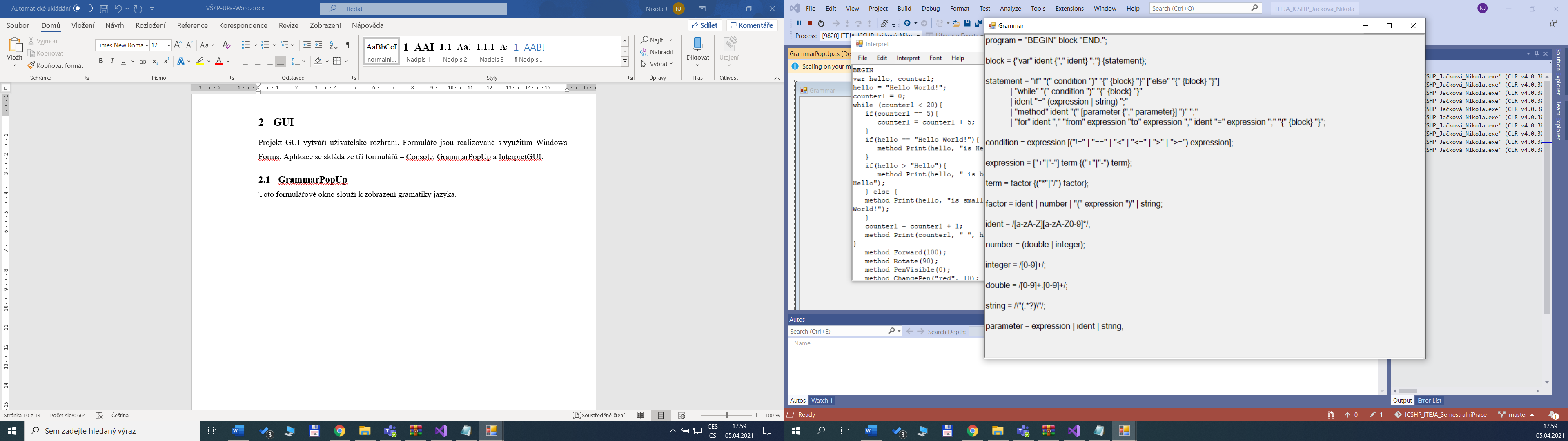


Obrázek : Hlavní okno aplikace

Hlavní okno aplikace na obrázku 3 zobrazuje v levém panelu zdrojový kód, v prostředním panelu vypisuje jednotlivé tokeny a v pravém panelu se zobrazí syntaktický strom.

## GrammarPopUp

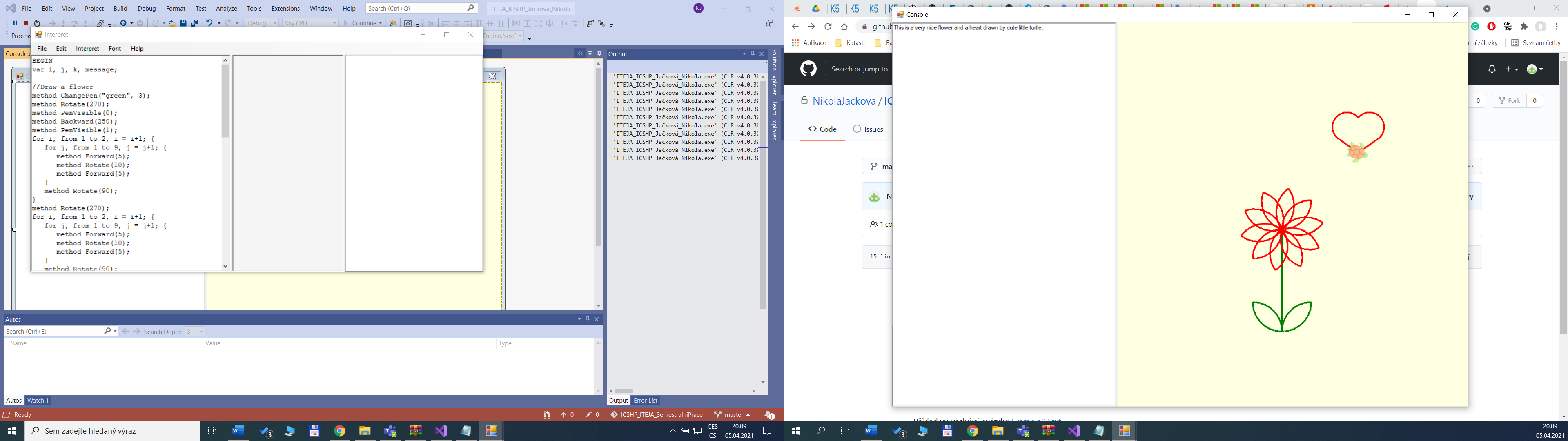
Toto formulářové okno slouží k zobrazení gramatiky jazyka. Vyvolá se z horní lišty z menu z položky Help → View Grammar. Text slouží pouze pro čtení.



Obrázek : Formulář Grammar

## Console

Tento formulář se vyvolá z horní lišty z menu z položky Interpret → Run Program. V levém panelu se zobrazí vypisovaný text, tzn. text, který vypisuje metoda Print. V pravém panelu se pak vykreslí grafické objekty dle zadaných instrukcí. Následující grafický výstup zobrazený na obrázku 5 je výstupem příkladu Example01.txt, který je součástí přílohy A.



Obrázek : Formulář Console

Závěr

Semestrální práce nabízí interpret fiktivního programovacího jazyka a následnou vizualizaci pomocí želví grafiky. Díky semestrální práci jsem porozuměla průběhu interpretace zdrojového kódu a také mě naučila základní práci s formuláři v jazyce C#.

Přílohy

[Příloha A – Název přílohy 19](#_Toc500229876)

Příloha A – Název přílohy

Popis přílohy A.