Dokumentacja projektu

Interpreter kod Java na bytecode

Łukasz Graczyk

# Język programowania: Python

# Źródło: własne, część metod aplikacji okienkowej pozyskana z dokumentacji PySimpleGui i tkinter

Wykorzystane komponenty:

import os

import shlex

from tkinter import Tk

import wx

import PySimpleGUI as sg

import string

import variables as vr 🡨 plik własny

import re 🡨 regex

def ShowFontDialog():

Pokazuje okienko z możliwością wyboru fontu w interfejsie notatnika.

def ShowPrintDialog():

Pokazuje okienko z możliwością wydruku zawartości interfejsu notatnika.

def ShowPageSetupDialog():

Pokazuje okienko z możliwością ustawienia strony/szerokości/rozmiaru.

def rgb2hex(r, g, b):

Formatuje wartość RGB w zapisie 0-255 do formatu hex.

def open\_file\_binary()

Otwiera plik ze ścieżki podanej przez użytkownika i pokazuje w interfejsie binarnym.

def save\_file(file\_name: str):

Zapisuje plik, plik zawiera zawartość interfejsu notatnika kodu javy

def save\_file\_binary(file\_name: str):

Zapisuje plik, plik zawiera zawartość interfejsu notatnika kodu bytecodu

def save\_as\_binary() -> str:

Zapisuje plik do formatu .class, plik zawiera zawartość interfejsu notatnika kodu bytecodu

def save\_as() -> str:

Zapisuje plik do formatu .txt, plik zawiera zawartość interfejsu notatnika kodu javy

def get\_word\_count():

Zwraca ilość wyrazów zawartych w interfejsie notatnika wpisanym na górnym interfejsie użytkownika

def validate\_text() -> bool:

Sprawdza poprawność tekstu pisanego w oknie interfejsu użytkownika

def characters\_without\_spaces():

Liczy wyrazy bez białych znaków

def get\_line\_count():

Zwraca ilość linii

def ShowMessageBox(title: str, message: str):

Pokazuje okienko z wiadomością

def AboutNotepadPyPlus():

Pokazuje okienko z wiadomością zawierającą informację na temat programu

def ConvertStart():

Pobiera zawartość okna interfejsu, dzieli zawartość na linie, przekazuje do Do metody poniżej (LineConver) sprawdza poprawność kodu

def LineConvert(words):

Pobrana linia zostaje przetworzona przez funkcje I zagnieżdżenia, trafia do poszczególnych metod przedstawionych poniżej, dodatkowo dodawane są linie odpowiedzialne

def integerParse(words):

def floatParse(words):

def doubleParse(words):

def stringParse(words):

def charParse(words):

def boolParse(words):

def forParser(words):

def randomParse(words):

def tabIntParse(words):

Parsuje linie pobraną jako argument, zgodną ze składnią Javy na bytecode. Sprawdza poprawność napisanego kodu pod względem ilości linii, podanych argumentów, znaków występujących w deklaracji, w razie błędu wyrzuca wyjątek i kończy działanie metody, pokazując wiadomość ERROR

def intTryParse(value):

def floatTryParse(value):

Sprawdza poprawność wartości przekazywanych do integer parse/ float parse za pomocą sprawdzania typów w wyjątku

def randomOperations(variable, words,randomVariable):

Przetwarza argumenty podane przez użytkownika do metody Random javy

def variableIntegerTabOperations(variable,words):

Pozwala pobierać odpowiednie argumenty I nadpisywać dane wartości w tablicy wcześniej zdefiniowanej

def variableIntegerOperations(variable,words):

def variableFloatsOperations(variable,words):

def variableDoublesOperations(variable,words):

pozwala przeprowadzić operacje arytmetyczne podawane przez użytkownika do wcześniej zadeklarowanych zmiennych

def dodajEtykiete():

def dodajLinie():

Dodaje etykietę I linie do kodu bajtowego w razie potrzeby wygenerownia większej ilości etykiet, tablica etykiet wywołuje funkcję przed kolejnym dodaniem etykiety i powiększa ja o następne litery z alfabetu, następnie wykorzystuje liczby, zapis liczb jest liczony następująco: 0, 01, 012, etc.

def AddNumbers():

Dodaje etykiety bytecodu

TYPY - literał:

# INT

<literal-int>::=<typ-literalu><nazwa-zmiennej><znak-rownosci><znak-przed-liczba><liczba-calkowita>„;”

<liczba-calkowita> ::= <cyfra> | <liczba-calkowita> <cyfra>

<cyfra> ::= 0|1|2|3|4|5|6|7|8|9

<typ-literalu>::=”int”

<nazwa-zmiennej>::=<litera>|<litera><nazwa-zmiennej>

<litera>::="A"|"B"|"C"|"D"|"E"|"F" ... |"Z" |"a"|"b"|"c"|"d"|"e"|"f" ... |"z"

<znak-rownosci>::=”=”

<znak-przed-liczba>::=”-„|”+”

# FLOAT

<literal-float>::=<typ-literalu><nazwa-zmiennej><znak-rownosci><znak-przed-liczba><liczba-rzeczywista>”;”

<typ-literalu>::=”float”

<nazwa-zmiennej>::=<litera>|<litera><nazwa-zmiennej>

<litera>::="A"|"B"|"C"|"D"|"E"|"F" ... |"Z" |"a"|"b"|"c"|"d"|"e"|"f" ... |"z"

<znak-rownosci>::=”=”

<znak-przed-liczba>::=”-”|”+”

<liczby-rzeczywiste> ::= <liczba-calkowita>”.”<ułamek>

<ułamek> ::= <cyfra> | <ułamek><cyfra>

<liczba-calkowita> ::= <cyfra> | <liczba-calkowita> <cyfra>

<cyfra> ::= 0|1|2|3|4|5|6|7|8|9

# BOOL

<literal-bool>::=<typ-literalu><nazwa-zmiennej><znak-rownosci><typ-bool> „;”

<liczba-calkowita> ::= <cyfra> | <liczba-calkowita> <cyfra>

<typ-literalu>::=”bool”

<nazwa-zmiennej>::=<litera>|<litera><nazwa-zmiennej>

<litera>::="A"|"B"|"C"|"D"|"E"|"F" ... |"Z" |"a"|"b"|"c"|"d"|"e"|"f" ... |"z"

<znak-rownosci>::=”=”

<typ-bool>::=”true„|”false”|0|1

# CHAR

<literal-char>::=<typ-literalu><nazwa-zmiennej><znak-rownosci><litera> „;” <typ-literalu>::=”char”

<nazwa-zmiennej>::=<litera>|<litera><nazwa-zmiennej>

<litera>::="A"|"B"|"C"|"D"|"E"|"F" ... |"Z" |"a"|"b"|"c"|"d"|"e"|"f" ... |"z"

<znak-rownosci>::=”=”

# STRING

<literal-string>::=<typ-literalu><nazwa-zmiennej><znak-rownosci><ciag-znakow> „;” <typ-literalu>::=”string”

<nazwa-zmiennej>::=<litera>|<litera><nazwa-zmiennej>

<ciag-znakow>::=<litera>|<litera><ciag-znakow>

<litera>::="A"|"B"|"C"|"D"|"E"|"F" ... |"Z" |"a"|"b"|"c"|"d"|"e"|"f" ... |"z"

<znak-rownosci>::=”=”

# DOUBLE

<literal-double>::=<typ-literalu><nazwa-zmiennej><znak-rownosci><znak-przed-liczba><liczba-rzeczywista>”;”

<typ-literalu>::=”double”

<nazwa-zmiennej>::=<litera>|<litera><nazwa-zmiennej>

<litera>::="A"|"B"|"C"|"D"|"E"|"F" ... |"Z" |"a"|"b"|"c"|"d"|"e"|"f" ... |"z"

<znak-rownosci>::=”=”

<znak-przed-liczba>::=”-”|”+”

<liczby-rzeczywiste> ::= <liczba-calkowita>”.”<ułamek>

<ułamek> ::= <cyfra> | <ułamek><cyfra>

<liczba-calkowita> ::= <cyfra> | <liczba-calkowita> <cyfra>

<cyfra> ::= 0|1|2|3|4|5|6|7|8|9

# TABLICA

<literal-tablica>::=<typ-literalu><nazwa-zmiennej>”[”<liczba-calkowita>”]”<znak-rownosci>”new ”<typ-literalu>”[„liczba-calkowita>„];”

<liczba-calkowita> ::= <cyfra> | <liczba-calkowita> <cyfra>

<cyfra> ::= 0|1|2|3|4|5|6|7|8|9

<typ-literalu>::=”int”

<nazwa-zmiennej>::=<litera>|<litera><nazwa-zmiennej>

<litera>::="A"|"B"|"C"|"D"|"E"|"F" ... |"Z" |"a"|"b"|"c"|"d"|"e"|"f" ... |"z"

<znak-rownosci>::=”=”

<znak-przed-liczba>::=”-„|”+”

# DOSTEP DO TABLICY

<*array access*> ::= <*expression name*> [ <*expression*> ] | <*primary no new array*> [ <*expression*>]

<*expression name*> ::= <*identifier*> | <*ambiguous name*> . <*identifier*>

<*postincrement expression*> ::= <*postfix expression*> ++

<*ambiguous name*>::= <*identifier*> | <*ambiguous name*>. <*identifier*>

<*primary no new array*> ::= <*literal*> | this | ( <*expression*> ) | <*class instance creation expression*> | <*field access*> | <*method invocation*> | <*array access*>

<*postfix expression*> ::= <*primary*> | <*expression name*> | <*postincrement expression*> | <*postdecrement expression*>

<*literal*> ::= <*integer literal*> | <*floating-point literal*> | <*boolean literal*> | <*character literal*> | <*string literal*> | <*null literal*>

<*expression*> ::= <*assignment expression*>

<*method invocation*> ::= <*method name*> ( <*argument list*>? ) | <*primary*> . <*identifier*> ( <*argument list*>? ) | super . <*identifier*> ( <*argument list*>? )

<*primary*> ::= <*primary no new array*> | <*array creation expression*>

<*field access*> ::= <*primary*> . <*identifier*> | super . <*identifier*>

<*primary*> ::= <*primary no new array*> | <*array creation expression*>

<*array creation expression*> ::= new <*primitive type*> <*dim exprs*> <*dims*>? | new <*class or interface type*> <*dim exprs*> <*dims*>?

<*dim exprs*> ::= <*dim expr*> | <*dim exprs*> <*dim expr*>

<*dim expr*> ::= [ <*expression*> ]

<*dims*> ::= [ ] | <*dims*> [ ]

<*class instance creation expression*> ::= new <*class type*> ( <*argument list*>? )

<*argument list*> ::= <*expression*> | <*argument list*> , <*expression*>

<*class type list*> ::= <*class type*> | <*class type list*> , <*class type*>

# IF STATEMENT / FOR

<if then statement>::= if ( <expression> ) <statement>

<expression> ::= <assignment expression>

<statement> ::= <statement without trailing substatement> | <labeled statement> | <if then statement> | <if then else statement> | <for statement>

<for statement> ::= for ( <for init>? ; <expression>? ; <for update>? ) <statement>

<for statement no short if> ::= for ( <for init>? ; <expression>? ; <for update>? ) <statement no short if>

<for init> ::= <statement expression list> | <local variable declaration>

<for update> ::= <statement expression list>

# WYKONANIE OPERACJI ARYTMETYCZNEJ

<operacja>::=<nazwa-zmiennej><znak-rownosci><nazwa-zmiennej><znak-operacji><nazwa-zmiennej>

<nazwa-zmiennej>::=<litera>|<litera><nazwa-zmiennej>

<znak-rownosci>::=”=”

<znak-operacji>::=+|-|\*|/|%

# RANDOM

<deklaracja-random>::= <nazwa-zmiennej><znak-rownosci>”new Random();”

<znak-rownosci>::=”=”

<nazwa-zmiennej>::=<litera>|<litera><nazwa-zmiennej>

<litera>::="A"|"B"|"C"|"D"|"E"|"F" ... |"Z" |"a"|"b"|"c"|"d"|"e"|"f" ... |"z"

# WYKORZYSTANIE RANDOM

<wykorzystanie-random>::=<literal-int><znak-rownosci>”random.nextint(”<cyfra>”);”

<literal-int>::=<typ-literalu><nazwa-zmiennej><znak-rownosci><znak-przed-liczba><liczba-calkowita>„;”

<liczba-calkowita> ::= <cyfra> | <liczba-calkowita> <cyfra>

<cyfra> ::= 0|1|2|3|4|5|6|7|8|9

<typ-literalu>::=”int”

<nazwa-zmiennej>::=<litera>|<litera><nazwa-zmiennej>

<litera>::="A"|"B"|"C"|"D"|"E"|"F" ... |"Z" |"a"|"b"|"c"|"d"|"e"|"f" ... |"z"

<znak-rownosci>::=”=”

<znak-przed-liczba>::=”-„|”+”

# DEKLARACJA ZMIENNYCH W PROGRAMIE:

### dopuszczalne typy:

bool ‘name’ = ‘value’;

int ‘name’ = ‘value’;

float ‘name’ = ‘value’;

double ‘name’ = ‘value’;

string ‘name’ = ‘value’;

char ‘name’ = ‘value’;

bool ‘name’=‘value’;

int ‘name’=‘value’;

float ‘name’=‘value’;

double ‘name’=‘value’;

string ‘name’=‘value’;

char ‘name’=‘value’;

### Operacje na zmiennych:

‘name\_of\_value’ = ‘name\_of\_value’ ‘+ / \* % - ‘ ‘name\_of\_value’

### PĘTLA FOR:

for ( int ‘name’ = ‘value’ ; ‘name’ [<=/==/>=/!=] ‘value’ ; ‘name’[++/--] ) {

[operations] };

### RANDOM:

Random ‘name’ = new Random();

int ‘name’= random.nextInt(‘value’);

### WYPISYWANIE:

System.out.print( "[string\_value]" );

System.out.println( "[string\_value]");

System.out.print( ‘name\_of\_value’ );

System.out.println( ‘name\_of\_value’ );

### Operacje na tablicach:

int[] ‘name’ = new tab[‘value’];

‘name’ [‘value’] = ‘value’;