# **Листинг программы**

## **GUI.py (Графический интерфейс программы)**

import subprocess  
from tkinter import \*  
from tkinter import filedialog, colorchooser  
from PIL import Image, ImageTk  
from Lexer import Lexer  
from SyntaxAnalyzer import Syntax\_analyzer  
  
current\_file = None  
value = True  
  
  
# Функции для работы программы  
# Функция "Открытие Word-файла"  
def open\_word(file\_path):  
 try:  
 subprocess.Popen(['start', 'winword', file\_path], shell=True)  
 except Exception as e:  
 print(f'Произошла ошибка: {e}')  
  
  
# Функция "Открыть справку - о программе"  
def open\_guide():  
 try:  
 subprocess.Popen(['start', 'winword', './Files\_tools/8. Справка (О программе).docx'], shell=True)  
 except Exception as e:  
 print(f'Произошла ошибка: {e}')  
  
  
# Функция "Создание файла"  
def create\_file():  
 global text\_editor  
 text\_editor = Text(window,  
 width=40,  
 height=15,  
 background='white',  
 font=('Calibri', 14),  
 bd=0,  
 highlightthickness=1,  
 highlightbackground='black'  
 )  
 text\_editor.place(x=10, y=102)  
  
  
# Функция "Открыть файл"  
def open\_file():  
 create\_file()  
 file\_path = filedialog.askopenfilename(filetypes=[("Text files", "\*.txt"), ("Python files", "\*.py")])  
 if file\_path:  
 with open(file\_path, "r") as file:  
 file\_content = file.read()  
 text\_editor.delete("1.0", END)  
 text\_editor.insert("1.0", file\_content)  
  
  
# Функция "Сохранить файл"  
def save\_file():  
 global current\_file  
 if current\_file:  
 with open(current\_file, "w") as file:  
 user\_input\_text = text\_editor.get("1.0", END)  
 file.write(user\_input\_text)  
 else:  
 save\_file\_as()  
  
  
# Функция "Сохранить файл как"  
def save\_file\_as():  
 global current\_file  
 file\_path = filedialog.asksaveasfilename(defaultextension=".txt",  
 filetypes=[("Text files", "\*.txt"),  
 ("All files", "\*.\*")])  
 if file\_path:  
 with open(file\_path, "w") as file:  
 user\_input\_text = text\_editor.get("1.0", END)  
 file.write(user\_input\_text)  
 current\_file = file\_path  
  
  
# Функция "Увеличение шрифта"  
def increase\_font():  
 current\_font = text\_editor['font']  
 font\_specs = str(current\_font).split()  
 size = int(font\_specs[-1])  
 new\_size = size + 2  
 text\_editor.configure(font=('Calibri', new\_size))  
  
  
# Функция "Уменьшение шрифта"  
def decrease\_font():  
 current\_font = text\_editor['font']  
 font\_specs = str(current\_font).split()  
 size = int(font\_specs[-1])  
 if size > 2:  
 new\_size = size - 2  
 text\_editor.configure(font=('Calibri', new\_size))  
 else:  
 pass  
  
  
# Функция "Очистить окно редактора"  
def delete\_text():  
 text\_editor.delete('1.0', 'end')  
 console.configure(text='')  
  
  
# Функция "Выбор цвета для шрифта"  
def change\_color\_font():  
 color = colorchooser.askcolor(title='Выбор цвета текста')  
 if color[1]:  
 text\_editor.configure(fg=color[1])  
  
  
# Функция "Смена темы GUI"  
def theme\_color():  
 global value  
 if value == True:  
 dark\_color\_button = '#82877b'  
 window.configure(background='#212121') # темная тема  
 console.configure(background='#2c2c2c',  
 highlightthickness=1,  
 highlightbackground='white')  
 console\_label.configure(background='#FFFFFF')  
 text\_editor\_label.configure(background='#FFFFFF')  
 text\_editor.configure(background='#2c2c2c',  
 highlightthickness=1,  
 highlightbackground='white',  
 fg='#ff8000')  
 button\_create\_file.configure(background=dark\_color\_button)  
 button\_open\_file.configure(background=dark\_color\_button)  
 button\_save\_file.configure(background=dark\_color\_button)  
 button\_font\_up.configure(background=dark\_color\_button)  
 button\_font\_down.configure(background=dark\_color\_button)  
 button\_clear\_text.configure(background=dark\_color\_button)  
 button\_color\_text.configure(background=dark\_color\_button)  
 button\_theme.configure(background=dark\_color\_button)  
 button\_start.configure(background=dark\_color\_button)  
  
 value = False  
  
 else:  
 white\_color\_button = 'light grey'  
 window.configure(background='#ecf0f1')  
 console.configure(background='white',  
 highlightthickness=1,  
 highlightbackground='black'  
 )  
 console\_label.configure(background='#ecf0f1')  
 text\_editor\_label.configure(background='#ecf0f1')  
 text\_editor.configure(background='white',  
 highlightthickness=1,  
 highlightbackground='black',  
 fg='#000000')  
 button\_create\_file.configure(background=white\_color\_button)  
 button\_open\_file.configure(background=white\_color\_button)  
 button\_save\_file.configure(background=white\_color\_button)  
 button\_font\_up.configure(background=white\_color\_button)  
 button\_font\_down.configure(background=white\_color\_button)  
 button\_clear\_text.configure(background=white\_color\_button)  
 button\_color\_text.configure(background=white\_color\_button)  
 button\_theme.configure(background=white\_color\_button)  
 button\_start.configure(background=white\_color\_button)  
 value = True  
  
  
# Функция "Запуск анализатора"  
def syntax\_analyzer():  
 input\_code = text\_editor.get("1.0", END)  
 tokens = Lexer().add\_tokens(input\_code)  
 output = Syntax\_analyzer().analyzer(tokens)  
 print(f'Входные токены: {tokens}')  
 if not output:  
 console.config(text='Исходное выражение:' '\n' f'{input\_code}' '\n' 'Ошибок не обнаружено',  
 justify='left',  
 anchor='nw',  
 foreground='black')  
 else:  
 console.config(text='\n'.join(map(lambda x: f'ERROR: {x}', output)),  
 justify='left',  
 anchor='nw',  
 foreground='red')  
  
  
# Основное окно программы  
window = Tk()  
window.title('Объявление целочисленной константы с инициализацией на Java')  
window.geometry('855x500')  
window.configure(background='#ecf0f1')  
window.iconphoto(False, PhotoImage(file='./Icons/Icon\_1.png'))  
window.wm\_attributes('-alpha', 1)  
window.resizable(width=False, height=False)  
  
# "Меню компилятора"  
menu = Menu(window)  
# Пункт "Файл" меню компилятора  
file\_item = Menu(menu, tearoff=0)  
file\_item.add\_command(label='Создать', command=create\_file)  
file\_item.add\_separator()  
file\_item.add\_command(label='Открыть', command=open\_file)  
file\_item.add\_separator()  
file\_item.add\_command(label='Сохранить', command=save\_file)  
file\_item.add\_separator()  
file\_item.add\_command(label='Сохранить как', command=save\_file\_as)  
file\_item.add\_separator()  
file\_item.add\_command(label='Выход')  
file\_item.add\_separator()  
menu.add\_cascade(label='Файл', menu=file\_item)  
  
# Пункт "Инструменты" меню компилятора  
tools\_item = Menu(menu, tearoff=0)  
files\_tools = [  
 ('Постановка задачи', './Files\_tools/1. Постановка задачи.docx'),  
 ('Порождающая грамматика', './Files\_tools/2. Порождающая грамматика.docx'),  
 ('Классификация Хомского', './Files\_tools/3. Классификация Хомского.docx'),  
 ('Метод анализа', './Files\_tools/4. Метод.docx'),  
 ('Диагностика и нейтрализация', './Files\_tools/5. Диагностика и нейтрализация.docx'),  
 ('Тестирование', './Files\_tools/6. Тестирование.docx'),  
 ('Листинг программы', './Files\_tools/7. Листинг программы.docx')  
]  
for file, file\_path in files\_tools:  
 tools\_item.add\_command(label=file, command=lambda path=file\_path: open\_word(path))  
 tools\_item.add\_separator()  
  
menu.add\_cascade(label='Инструменты', menu=tools\_item)  
  
# Пункт "Справка" меню компилятора  
help\_item = Menu(menu, tearoff=0)  
help\_item.add\_command(label='О программе', command=open\_guide)  
help\_item.add\_separator()  
menu.add\_cascade(label='Справка', menu=help\_item)  
  
# Пункт "Выход" меню компилятора  
for \_ in range(43):  
 menu.add\_separator()  
menu.add\_command(label='Выход', command=window.quit)  
window.config(menu=menu)  
  
# Разграничение GUI  
separator\_canvas = Canvas(window, width=853, height=2, bg='grey', highlightthickness=0)  
separator\_canvas.place(x=1, y=0)  
  
separator\_canvas = Canvas(window, width=853, height=2, bg='grey', highlightthickness=0)  
separator\_canvas.place(x=1, y=65)  
  
# Кнопки "Панели инструментов"  
# Кнопка "Создать файл"  
photo\_create\_file = Image.open('./Icons/file\_create.png')  
img\_create\_file = ImageTk.PhotoImage(photo\_create\_file)  
button\_create\_file = Button(window, image=img\_create\_file, width=25, height=25, background='light grey',  
 command=create\_file)  
button\_create\_file.place(x=60, y=20)  
  
# Кнопка "Открыть файл"  
photo\_open\_file = Image.open('./Icons/file\_open.png')  
img\_open\_file = ImageTk.PhotoImage(photo\_open\_file)  
button\_open\_file = Button(window, image=img\_open\_file, width=25, height=25, background='light grey', command=open\_file)  
button\_open\_file.place(x=105, y=20)  
  
# Кнопка "Сохранить файл"  
photo\_save\_file = Image.open('./Icons/file\_save.png')  
img\_save\_file = ImageTk.PhotoImage(photo\_save\_file)  
button\_save\_file = Button(window, image=img\_save\_file, width=25, height=25, background='light grey', command=save\_file)  
button\_save\_file.place(x=150, y=20)  
  
# Кнопка "Увеличение шрифта"  
photo\_font\_up = Image.open('./Icons/size\_up.png')  
img\_font\_up = ImageTk.PhotoImage(photo\_font\_up)  
button\_font\_up = Button(window, image=img\_font\_up, width=25, height=25, background='light grey', command=increase\_font)  
button\_font\_up.place(x=270, y=20)  
  
# Кнопка "Уменьшение шрифта"  
photo\_font\_down = Image.open('./Icons/size\_down.png')  
img\_font\_down = ImageTk.PhotoImage(photo\_font\_down)  
button\_font\_down = Button(window, image=img\_font\_down, width=25, height=25, background='light grey',  
 command=decrease\_font)  
button\_font\_down.place(x=315, y=20)  
  
# Кнопка "Очистка экрана"  
photo\_clear\_text = Image.open('./Icons/delete\_text.png')  
img\_clear\_text = ImageTk.PhotoImage(photo\_clear\_text)  
button\_clear\_text = Button(window, image=img\_clear\_text, width=25, height=25, background='light grey',  
 command=delete\_text)  
button\_clear\_text.place(x=360, y=20)  
  
# Кнопка "Очистка экрана"  
photo\_color\_text = Image.open('./Icons/color\_font.png')  
img\_color\_text = ImageTk.PhotoImage(photo\_color\_text)  
button\_color\_text = Button(window, image=img\_color\_text, width=25, height=25, background='light grey',  
 command=change\_color\_font)  
button\_color\_text.place(x=405, y=20)  
  
# Кнопка "Тема"  
photo\_theme = Image.open('./Icons/theme.png')  
img\_theme = ImageTk.PhotoImage(photo\_theme)  
button\_theme = Button(window, image=img\_theme, width=25, height=25, background='light grey', command=theme\_color)  
button\_theme.place(x=450, y=20)  
  
# Кнопка "Старт"  
photo\_start = Image.open('./Icons/start\_2.png')  
img\_start = ImageTk.PhotoImage(photo\_start)  
button\_start = Button(window, image=img\_start, width=25, height=25, background='light grey', command=syntax\_analyzer)  
button\_start.place(x=600, y=20)  
  
# Окно редактирования и консоль  
# Консоль  
# Вывод надписи "Консоль"  
console\_label = Label(window,  
 text='Консоль (окно вывода сообщений)',  
 font='Calibri 13',  
 background='#ecf0f1')  
console\_label.place(x=450, y=75)  
# Окно консоли  
console = Label(window,  
 width=40,  
 height=15,  
 background='white',  
 font='Calibri 14',  
 bd=0,  
 highlightthickness=1,  
 highlightbackground='black',  
 wraplength=400)  
console.place(x=445, y=102)  
  
# Вывод надписи "Окно редактирования"  
text\_editor\_label = Label(window,  
 text='Окно редактирования',  
 font='Calibri 13',  
 background='#ecf0f1')  
text\_editor\_label.place(x=30, y=75)  
  
window.mainloop()

## **Lexer.py (Лексер синтаксического анализатора)**

import re  
  
  
class Lexer:  
 def \_\_init\_\_(self):  
 self.grammar = ['final', 'int', 'identifier', '=', 'digit', ';']  
 self.terminal\_symbol = {'final', 'int', '=', ';'}  
 self.identifier = r'[a-zA-Z][a-zA-Z0-9\_]\*'  
 self.digit = r'\d+'  
 self.tokens = []  
  
 def check\_tokens(self, tokens):  
 self.tokens = ['-' if i not in tokens else i for i in self.grammar]  
 return self.tokens  
  
 def add\_tokens(self, input\_code):  
 index = 1  
 for word in re.findall(r'\w+|\S', input\_code):  
 if word in self.terminal\_symbol:  
 self.tokens.append(word)  
 index += 1  
 elif re.fullmatch(self.identifier, word):  
 self.tokens.append('identifier')  
 index += 1  
 elif re.fullmatch(self.digit, word):  
 self.tokens.append('digit')  
 index += 1  
 else:  
 self.tokens.append('no match')  
  
 self.check\_tokens(self.tokens)  
  
 return self.tokens

## **SyntaxAnalyzer.py (Синтаксический анализатор)**

class Syntax\_analyzer:  
 def \_\_init\_\_(self):  
 self.grammar = ['final', 'int', 'identifier', '=', 'digit', ';']  
 self.errors = []  
  
 def add\_error(self, error):  
 return self.errors.append(error)  
  
 def analyzer(self, tokens):  
 if tokens[0] != self.grammar[0]:  
 self.add\_error('некорректное начало модификатора, ожидалось "final"')  
  
 if tokens[1] != self.grammar[1]:  
 self.add\_error('некорректно определен тип переменной, ожидалось "int"')  
  
 if tokens[2] != self.grammar[2]:  
 self.add\_error('некорректно заданная переменная')  
  
 if tokens[3] != self.grammar[3]:  
 self.add\_error('отсутствует знак присваивания (=) после переменной')  
  
 if tokens[4] != self.grammar[4]:  
 self.add\_error('некорректное выражение в правой части метода')  
  
 if tokens[5] != self.grammar[5]:  
 self.add\_error('пропущено окончание выражения (;)')  
  
 return self.errors