**Код программы**

**flashcards.py**

import tkinter as tk

from tkinter import ttk, messagebox

from typing import Protocol, Iterable

import matplotlib.pyplot as plt

from src.flashcards import Flashcard

class IFlashcardSource(Protocol):

def next(self): pass

@property

def current\_word(self): pass

@property

def current\_translation(self): pass

def correct\_answer(self): pass

def wrong\_answer(self): pass

def save\_progress(self): pass

def reset\_answers(self): pass

def clear\_word(self): pass

@property

def progress(self) -> Iterable: pass

@property

def total\_correct(self) -> int: pass

@property

def total\_wrong(self) -> int: pass

def try\_add\_flashcard(self, flashcard): pass

class AddWordGUI:

def \_\_init\_\_(self, root, flashcard\_source: IFlashcardSource):

self.\_\_root = root

self.\_\_flashcard\_source = flashcard\_source

tk.Label(self.\_\_root, text="Слово на английском:").pack(pady=10)

self.\_\_word\_entry = tk.Entry(self.\_\_root, font=("Arial", 14))

self.\_\_word\_entry.pack(pady=5)

tk.Label(self.\_\_root, text="Перевод:").pack(pady=10)

self.\_\_translation\_entry = tk.Entry(self.\_\_root, font=("Arial", 14))

self.\_\_translation\_entry.pack(pady=5)

self.\_\_save\_button = tk.Button(self.\_\_root, text="Сохранить", command=self.\_\_try\_save\_word)

self.\_\_save\_button.pack(pady=5)

def \_\_try\_save\_word(self):

word = self.\_\_word\_entry.get().strip()

translation = self.\_\_translation\_entry.get().strip()

if not word or not translation:

messagebox.showerror("Ошибка", "Заполните оба поля")

return

if not self.\_\_flashcard\_source.try\_add\_flashcard(Flashcard(word, translation)):

messagebox.showerror("Ошибка", f"Слово '{word}' уже есть в словаре")

return

messagebox.showinfo("Успех", f"Слово '{word}' добавлено в словарь")

self.\_\_root.destroy()

class FlashcardGUI:

def \_\_init\_\_(self, root, flashcard\_source: IFlashcardSource):

self.\_\_flashcard\_source = flashcard\_source

self.\_\_root = root

ttk.Label(self.\_\_root, text="Угадалка от нас", font=("Arial", 24, "bold"), anchor="center").pack(pady=20)

self.\_\_word = tk.StringVar()

self.\_\_word\_label = ttk.Label(self.\_\_root, textvariable=self.\_\_word, font=("Arial", 30, "bold"), anchor="center")

self.\_\_word\_label.pack(pady=10)

self.\_\_entry = ttk.Entry(self.\_\_root, font=("Arial", 16), justify="center")

self.\_\_entry.pack(pady=10)

self.\_\_show\_word\_button = ttk.Button(self.\_\_root, text="Показать слово", command=self.show\_word)

self.\_\_show\_word\_button.pack(pady=5)

self.\_\_check\_button = ttk.Button(self.\_\_root, text="Проверить перевод", command=self.\_\_check\_translation)

self.\_\_check\_button.pack(pady=5)

self.\_\_message\_label = ttk.Label(self.\_\_root, text="", font=("Arial", 16), anchor="center")

self.\_\_message\_label.pack(pady=10)

self.\_\_stats = tk.StringVar(value="Правильно: 0 | Неправильно: 0")

self.\_\_stats\_label = ttk.Label(self.\_\_root, textvariable=self.\_\_stats, font=("Arial", 14))

self.\_\_stats\_label.pack(pady=10)

self.\_\_show\_progress\_button = ttk.Button(self.\_\_root, text="Посмотреть прогресс", command=self.\_\_show\_progress)

self.\_\_show\_progress\_button.pack(pady=5)

self.\_\_add\_word\_button = ttk.Button(self.\_\_root, text="Добавить новое слово", command=self.\_\_add\_word)

self.\_\_add\_word\_button.pack(pady=5)

def \_\_add\_word(self):

add\_window = tk.Toplevel(self.\_\_root)

add\_window.title("Добавить слово")

add\_window.geometry("300x200")

AddWordGUI(add\_window, self.\_\_flashcard\_source)

def show\_word(self):

self.\_\_flashcard\_source.next()

self.\_\_word.set(self.\_\_flashcard\_source.current\_word or "")

self.\_\_entry.delete(0, tk.END)

def \_\_check\_translation(self):

if not self.\_\_flashcard\_source.current\_word:

messagebox.showerror("Ошибка", "Сначала выберите слово!")

return

user\_input = self.\_\_entry.get().strip().lower()

if user\_input == self.\_\_flashcard\_source.current\_translation:

self.\_\_correct\_answer()

else:

self.\_\_wrong\_answer()

def \_\_correct\_answer(self):

self.\_\_message\_label.config(text="Правильно!", foreground="green")

self.\_\_flashcard\_source.correct\_answer()

self.\_\_update\_stats()

self.show\_word()

def \_\_wrong\_answer(self):

self.\_\_message\_label.config(

text=f"Ошибка! Правильный перевод: {self.\_\_flashcard\_source.current\_translation}",

foreground="red",

)

self.\_\_flashcard\_source.wrong\_answer()

self.\_\_update\_stats()

def \_\_update\_stats(self):

"""Обновляет статистику."""

self.\_\_stats.set(f"Правильно: {self.\_\_flashcard\_source.total\_correct} | Неправильно: {self.\_\_flashcard\_source.total\_wrong}")

def clear\_word(self):

self.\_\_flashcard\_source.clear\_word()

self.\_\_word.set("")

def reset\_answers(self):

self.\_\_flashcard\_source.reset\_answers()

self.\_\_update\_stats()

def save\_progress(self):

self.\_\_flashcard\_source.save\_progress()

def \_\_show\_progress(self):

if not self.\_\_flashcard\_source.progress:

messagebox.showinfo("Прогресс", "Нет данных для отображения")

return

correct = [entry["correct"] for entry in self.\_\_flashcard\_source.progress]

wrong = [entry["wrong"] for entry in self.\_\_flashcard\_source.progress]

plt.figure(figsize=(10, 5))

plt.plot(correct, label="Правильные", color="green", marker="o")

plt.plot(wrong, label="Неправильные", color="red", marker="x")

plt.title("Прогресс обучения")

plt.xlabel("Тесты")

plt.ylabel("Количество ответов")

plt.legend()

plt.grid(True)

plt.show()

**testmode.py**

from tkinter import ttk, messagebox, StringVar

class Timer:

def \_\_init\_\_(self, root, on\_tick, on\_end, time\_sec=30):

self.\_\_root = root

self.\_\_limit = time\_sec

self.\_\_time = 0

self.\_\_timer = None

self.\_\_on\_tick = on\_tick

self.\_\_on\_end = on\_end

@property

def time(self):

return self.\_\_time

@property

def limit(self):

return self.\_\_limit

def start(self):

self.\_\_time = self.\_\_limit

self.\_\_timer = self.\_\_root.after(1000, self.\_\_tick)

def \_\_tick(self):

if self.\_\_time > 0:

self.\_\_time -= 1

self.\_\_on\_tick()

self.\_\_timer = self.\_\_root.after(1000, self.\_\_tick)

return

self.\_\_on\_end()

self.\_\_root.after\_cancel(self.\_\_timer)

class TestModeGUI:

def \_\_init\_\_(self, root, on\_start, on\_end, time=30):

self.\_\_root = root

self.\_\_on\_start = on\_start

self.\_\_on\_end = on\_end

self.\_\_time = StringVar()

self.\_\_timer\_label = ttk.Label(self.\_\_root, textvariable=self.\_\_time, font=("Arial", 16))

self.\_\_timer = Timer(self.\_\_root, self.\_\_update\_time, self.\_\_end, time)

self.\_\_start\_button = ttk.Button(self.\_\_root, text="Режим теста", command=self.\_\_start)

self.\_\_start\_button.pack(pady=10)

def \_\_update\_time(self):

self.\_\_time.set(f"Осталось времени: {self.\_\_timer.time}")

def \_\_start(self):

if not messagebox.askyesno("Режим теста", "Вы точно хотите войти в режим теста?"):

return

self.\_\_timer\_label.pack()

self.\_\_start\_button["state"] = "disabled"

self.\_\_on\_start()

self.\_\_timer.start()

self.\_\_update\_time()

def \_\_end(self):

messagebox.showinfo("Режим теста", "Тест завершён!")

self.\_\_start\_button["state"] = "normal"

self.\_\_timer\_label.pack\_forget()

self.\_\_on\_end()

**translator.py**

import tkinter as tk

from tkinter import ttk, messagebox

from typing import Protocol

class ITranslator(Protocol):

def translate(self, text) -> str: pass

@property

def src(self) -> str: pass

@src.setter

def src(self, value): pass

@property

def dst(self) -> str: pass

@dst.setter

def dst(self, value): pass

class TranslatorGUI:

def \_\_init\_\_(self, root, translator: ITranslator):

self.\_\_root = root

self.\_\_translator = translator

ttk.Label(self.\_\_root, text="Переводчик", font=("Arial", 24, "bold"), anchor="center").pack(pady=20)

ttk.Label(self.\_\_root, text="Введите текст для перевода:", font=("Arial", 14)).pack(pady=10)

self.input\_text = tk.Text(self.\_\_root, font=("Arial", 14), height=5, wrap=tk.WORD)

self.input\_text.pack(pady=5)

ttk.Label(self.\_\_root, text="Результат перевода:", font=("Arial", 14)).pack(pady=10)

self.output\_text = tk.Text(self.\_\_root, font=("Arial", 14), height=5, wrap=tk.WORD, state=tk.DISABLED)

self.output\_text.pack(pady=5)

self.translate\_button = ttk.Button(self.\_\_root, text="Перевести", command=self.\_\_translate\_text)

self.translate\_button.pack(pady=10)

def \_\_display\_text(self, text):

self.output\_text.config(state=tk.NORMAL)

self.output\_text.delete("1.0", tk.END)

self.output\_text.insert(tk.END, text)

self.output\_text.config(state=tk.DISABLED)

def \_\_translate\_text(self):

"""Перевод текста из текстового поля"""

input\_text = self.input\_text.get("1.0", tk.END).strip()

if not input\_text:

messagebox.showerror("Ошибка", "Введите текст для перевода!")

return

result = self.\_\_translator.translate(input\_text)

if not result:

messagebox.showerror("Ошибка", "Не удалось получить перевод")

return

self.\_\_display\_text(result)

**flashcards.py**

class Flashcard:

"""Карточка слова/предложения с переводом и счётчиками (не)правильных ответов."""

def \_\_init\_\_(self, word, translation):

self.\_\_word = word

self.\_\_translation = translation

self.\_\_correct = 0

self.\_\_wrong = 0

@property

def word(self):

return self.\_\_word

@property

def translation(self):

return self.\_\_translation

@property

def correct(self):

return self.\_\_correct

def inc\_correct(self):

self.\_\_correct += 1

@property

def wrong(self):

return self.\_\_wrong

def inc\_wrong(self):

self.\_\_wrong += 1

@property

def wc\_ratio(self):

return (self.wrong + 1) / (self.correct + 1)

def reset(self):

self.\_\_correct = 0

self.\_\_wrong = 0

class WeightedFlashcardList:

def \_\_init\_\_(self, rng, \*cards, \*\*words):

self.\_\_rng = rng

flashcards = {}

if words:

flashcards.update(\*\*{word: Flashcard(word, translation) for word, translation in words.items()})

if cards:

flashcards.update(\*\*{card.word: card for card in cards})

self.\_\_flashcards = flashcards

self.\_\_current = None

self.\_\_total\_correct = 0

self.\_\_total\_wrong = 0

self.\_\_total\_ratio\_cache = None

for card in self.\_\_flashcards.values():

self.\_\_total\_correct += card.correct

self.\_\_total\_wrong += card.wrong

self.\_\_progress = []

@property

def \_\_total\_wc\_ratio(self):

if not self.\_\_total\_ratio\_cache:

self.\_\_total\_ratio\_cache = sum([card.wc\_ratio for card in self.\_\_flashcards.values()])

return self.\_\_total\_ratio\_cache

def \_\_calculate\_probability(self, card):

return card.wc\_ratio / self.\_\_total\_wc\_ratio

def next(self):

probability\_dist = [self.\_\_calculate\_probability(card) for card in self.\_\_flashcards.values()]

self.\_\_current = self.\_\_rng.choice(list(self.\_\_flashcards.keys()), p=probability\_dist)

@property

def current\_word(self):

return self.\_\_current

@property

def current\_translation(self):

if not self.current\_word:

return None

return self.\_\_flashcards[self.current\_word].translation

def correct\_answer(self):

self.\_\_flashcards[self.current\_word].inc\_correct()

self.\_\_total\_correct += 1

self.\_\_total\_ratio\_cache = None

def wrong\_answer(self):

self.\_\_flashcards[self.current\_word].inc\_wrong()

self.\_\_total\_wrong += 1

self.\_\_total\_ratio\_cache = None

@property

def total\_correct(self):

return self.\_\_total\_correct

@property

def total\_wrong(self):

return self.\_\_total\_wrong

def reset\_answers(self):

for card in self.\_\_flashcards.values():

card.reset()

def clear\_word(self):

self.\_\_current = None

def save\_progress(self):

self.\_\_progress.append({"correct": self.total\_correct, "wrong": self.total\_wrong})

@property

def progress(self):

return self.\_\_progress

def try\_add\_flashcard(self, flashcard):

if self.\_\_flashcards.get(flashcard.word):

return False

self.\_\_flashcards[flashcard.word] = flashcard

return True

**main.py**

import tkinter as tk

from tkinter import ttk

from numpy.random import default\_rng

from googletrans import Translator as GoogleTranslator

from gui.flashcards import FlashcardGUI

from gui.translator import TranslatorGUI

from gui.testmode import TestModeGUI

from flashcards import WeightedFlashcardList

from translator import Translator

def main():

root = tk.Tk()

root.title("Угадалка от нас")

root.geometry("1200x600")

root.configure(bg="#2e2e2e")

flashcard\_source = WeightedFlashcardList(

default\_rng(),

\*\*{

"cat": "кот",

"dog": "собака",

"apple": "яблоко",

"car": "машина",

"house": "дом",

"sun": "солнце",

"moon": "луна",

"tree": "дерево",

"sky": "небо",

"water": "вода"

}

)

flashcards\_frame = tk.Frame(root, bg="#2e2e2e")

flashcard\_gui = FlashcardGUI(flashcards\_frame, flashcard\_source)

flashcards\_frame.pack(fill=tk.BOTH, expand=True)

translator\_window = tk.Toplevel(root, bg="#2e2e2e")

translator\_window.protocol("WM\_DELETE\_WINDOW", translator\_window.withdraw)

translator\_window.withdraw()

TranslatorGUI(translator\_window, Translator(GoogleTranslator(), "en", "ru")) # noqa

open\_translator = ttk.Button(root, text="Открыть переводчик", command=translator\_window.deiconify)

open\_translator.pack(expand=True)

def start\_test():

open\_translator["state"] = "disabled"

translator\_window.withdraw()

flashcard\_gui.show\_word()

flashcard\_gui.reset\_answers()

def end\_test():

open\_translator["state"] = "normal"

flashcard\_gui.clear\_word()

flashcard\_gui.save\_progress()

TestModeGUI(

root,

start\_test,

end\_test

)

root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**translator.py**

class Translator:

def \_\_init\_\_(self, translator, source: str, destination):

self.\_\_translator = translator

self.\_\_src: str = source

self.\_\_dst = destination

@property

def src(self) -> str:

return self.\_\_src

@src.setter

def src(self, value):

self.\_\_src = value

@property

def dst(self):

return self.\_\_dst

@dst.setter

def dst(self, value):

self.\_\_dst = value

def translate(self, text) -> str:

translation = self.\_\_translator.translate(text, src=self.src, dest=self.dst)

if translation: return translation.text

return ""

Диаграмма прецедентов:

