Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №31
дисциплины «Основы программной инженерии»

Выполнил:
Плугатырев Владислав Алексеевич
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Проверил Воронкин Роман Александрович
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Основы работы с Tkinter

Цель работы: приобретение навыков построения графического интерфейса пользователя GUI с помощью пакета Tkinter языка программирования Python версии 3.x.

Ход работы.

1. Создание репозитория.

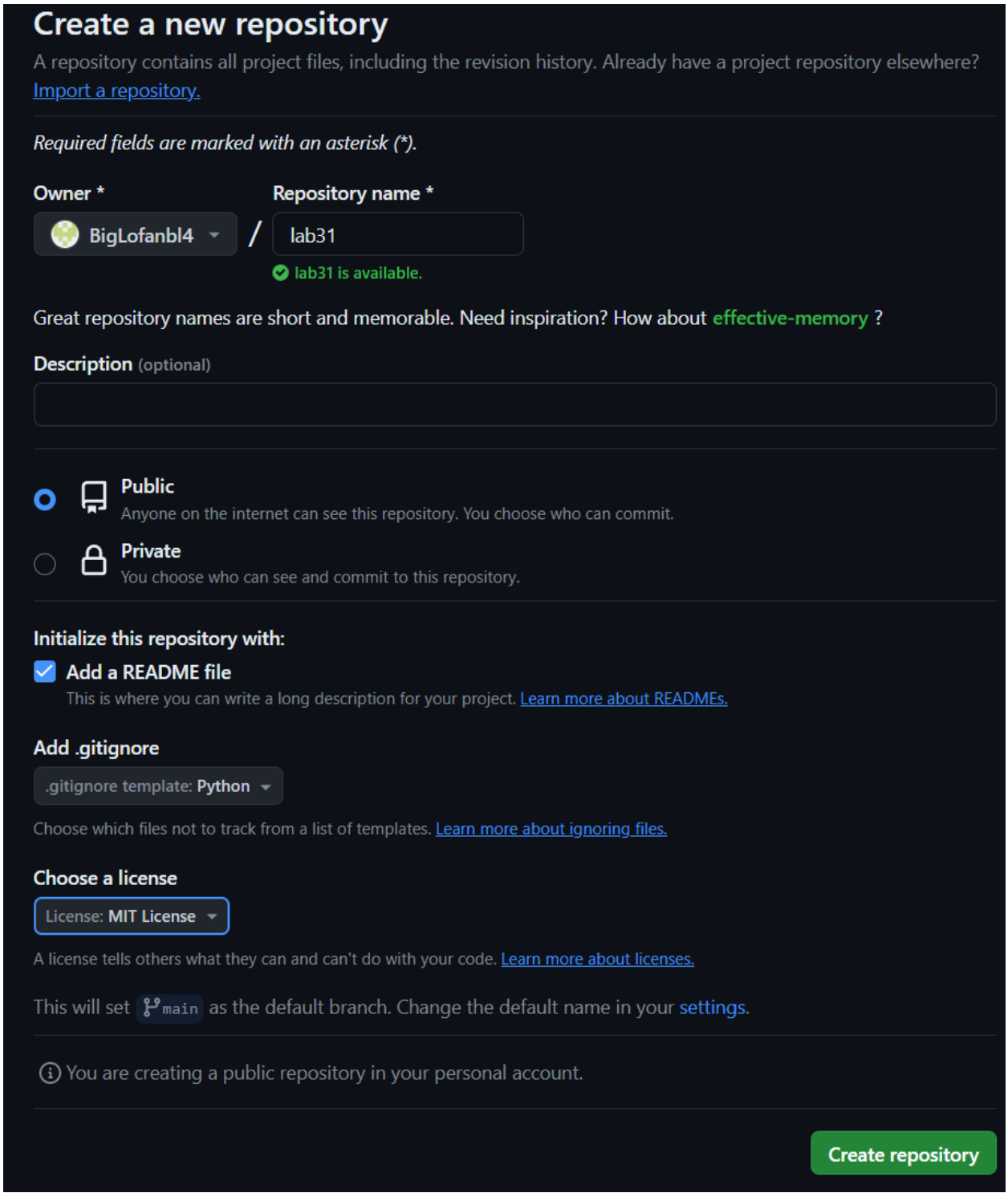

Рисунок 1.1 – Создание репозитория

2. Выполнение первого задания: напишите программу, состоящую из двух списков Listbox . В первом будет, например, перечень товаров, заданный программно. Второй изначально пуст, пусть это будет перечень покупок. При клике на одну кнопку товар должен переходить из одного списка в другой. При клике на вторую кнопку – возвращаться (человек передумал покупать). Предусмотрите возможность множественного выбора элементов списка и их перемещения.

```python
# !/usr/bin/env python3
# -*- coding: utf-8 -*-

import tkinter as tk

def add_to_cart():
    selected_items = list_box_products.curselection()
    for i in reversed(selected_items):
        cart_list.insert(tk.END, list_box_products.get(i))
        list_box_products.delete(i)

def remove_from_cart():
    selected_items = cart_list.curselection()
    for i in reversed(selected_items):
        list_box_products.insert(tk.END, cart_list.get(i))
        cart_list.delete(i)


if __name__ == "__main__":
    root = tk.Tk()
    root.title("Список покупок")

    list_box_products = tk.Listbox(root, selectmode='extended', height=10, exportselection=0)
    list_box_products.pack(side=tk.LEFT, fill=tk.Y, padx=(10,0), pady=10)

    products = ["Хлеб", "Молоко", "Яйца", "Колбаса", "Сыр", "Чай", "Сахар", "Соль"]
    for product in products:
        list_box_products.insert(tk.END, product)

    cart_list = tk.Listbox(root, selectmode='extended', height=10, exportselection=0)
    cart_list.pack(side=tk.RIGHT, fill=tk.Y, padx=(0,10), pady=10)

    add_button = tk.Button(root, text="Добавить в корзину ->", command=add_to_cart)
    add_button.pack(pady=5, padx=10)
    remove_button = tk.Button(root, text="<- Удалить из корзины", command=remove_from_cart)
    remove_button.pack(pady=5, padx=10)

    root.mainloop()
```

Рисунок 2.1 – Код программы

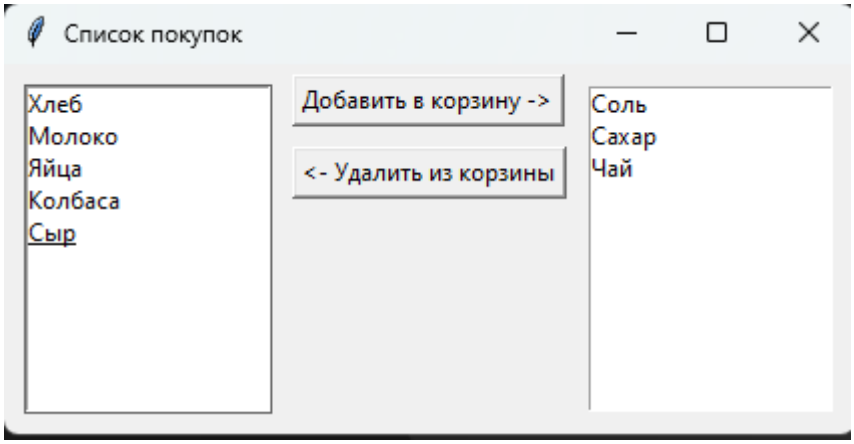

Рисунок 2.2 - Результат

3. Выполнение второго задания: напишите программу по следующему описанию. Нажатие Enter в однострочном текстовом поле приводит к перемещению текста из него в список (экземпляр Listbox). При двойном клике по элементу-строке списка, она должна копироваться в текстовое поле.

```python
# !/usr/bin/env python3
# -*- coding: utf-8 -

import tkinter as tk

def move_to_list(event):
    text = entry.get()
    if text:
        listbox.insert(tk.END, text)
        entry.delete(0, tk.END)

def copy_to_entry(event):
    selected_indices = listbox.curselection()
    if selected_indices:
        selected_text = listbox.get(selected_indices[0])
        entry.delete(0, tk.END)
        entry.insert(0, selected_text)


if __name__ == "__main__":
    root = tk.Tk()
    root.title("Entry to Listbox")

    entry = tk.Entry(root)
    entry.bind('<Return>', move_to_list)
    entry.pack(pady=10)

    listbox = tk.Listbox(root)
    listbox.bind('<Double-Button-1>', copy_to_entry)
    listbox.pack(pady=10)

    root.mainloop()
```

Рисунок 3.1 – Код программы

Рисунок 3.2 - Результат

4. Выполнение третьего задания: напишите программу по описанию. Размеры многострочного текстового поля определяются значениями, введенными в однострочные текстовые поля. Изменение размера происходит при нажатии мышью на кнопку, а также при нажатии клавиши Enter. Цвет фона экземпляра Text светлосерый (lightgrey), когда поле не в фокусе, и белый, когда имеет фокус.

```python
import tkinter as tk


def resize_text_field():
    try:
        new_width = int(width_entry.get())
        new_height = int(height_entry.get())
        text_field.config(width=new_width, height=new_height)
    except ValueError:
        print("Пожалуйста, введите числовые значения для ширины и высоты.")


def on_focus_in(event):
    text_field.config(bg="white")

def on_focus_out(event):
    text_field.config(bg="lightgrey")


if __name__ == "__main__":
    root = tk.Tk()
    root.title("Ресайзер текстового поля")

    width_entry = tk.Entry(root)
    width_entry.pack(pady=(10, 0))
    height_entry = tk.Entry(root)
    height_entry.pack(pady=10)

    resize_button = tk.Button(
        root, text="Изменить размер", command=resize_text_field
    )
    resize_button.pack(pady=10)

    root.bind("<Return>", lambda event: resize_text_field())

    text_field = tk.Text(root, width=20, height=5, bg="lightgrey")
    text_field.pack(pady=10)
    text_field.bind("<FocusIn>", on_focus_in)
    text_field.bind("<FocusOut>", on_focus_out)

    root.mainloop()
```

Рисунок 4.1 – Код программы

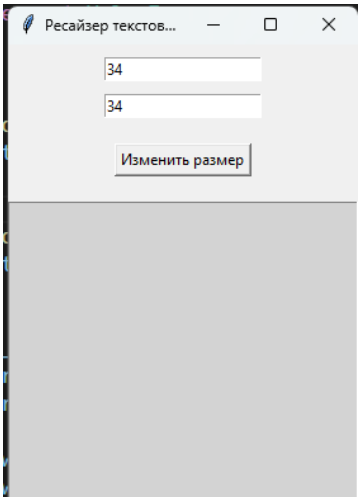

Рисунок 4.2 – Результат

5. Выполнение четвертого задания: создать изображение на холсте.

```python
import tkinter as tk

if __name__ == "__main__":
    root = tk.Tk()
    root.title("Анапа 2007")

    canvas = tk.Canvas(root, width=400, height=400)
    canvas.pack()

    center_x = canvas.winfo_reqwidth() / 2
    center_y = canvas.winfo_reqheight() / 2

    # Рисуем дом
    house_width = 100
    house_height = 120
    canvas.create_rectangle(
        center_x - house_width / 2,
        center_y - house_height / 2,
        center_x + house_width / 2,
        center_y + house_height / 2,
        fill="lightblue",
    )

    # Рисуем крышу дома
    canvas.create_polygon(
        center_x,
        center_y - house_height / 2,
        center_x - house_width / 2,
        center_y - house_height / 2,
        center_x,
        center_y - house_height / 2 - 40,
        center_x + house_width / 2,
        center_y - house_height / 2,
        fill="darkgrey",
    )

    # Рисуем солнце в правом верхнем углу
    sun_radius = 30
    sun_center_x = 350
    sun_center_y = 50
    canvas.create_oval(
        sun_center_x - sun_radius,
        sun_center_y - sun_radius,
        sun_center_x + sun_radius,
        sun_center_y + sun_radius,
        fill="orange",
    )

    # Рисуем траву в нижней части холста
    grass_color = "green"
    grass_height = 20
    line_width = 3
    for i in range(0, canvas.winfo_reqwidth(), 20):
        x = i
        y_start = canvas.winfo_reqheight()
        y_end = y_start - grass_height
        x_end = x + grass_height / 2  # Добавляем уклон вправо к верху
        canvas.create_line(
            x, y_start, x_end, y_end, fill=grass_color, width=line_width
        )

    root.mainloop()
```

Рисунок 5.1 – Код программы

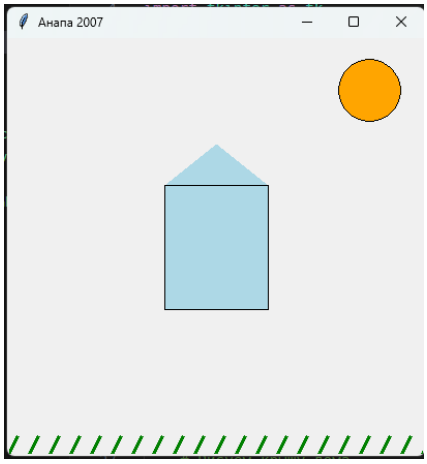

Рисунок 5.2 – Результат

6. Выполнение пятого задания: изучите приведенную программу и самостоятельно запрограммируйте постепенное движение фигуры в ту точку холста, где пользователь кликает левой кнопкой мыши.

```python
# !/usr/bin/env python3
# -*- coding: utf-8 -

from tkinter import *

def motion(event):
    current_coords = c.coords(ball)
    center_x = (current_coords[0] + current_coords[2]) / 2
    center_y = (current_coords[1] + current_coords[3]) / 2

    delta_x = (event.x - center_x) / 50
    delta_y = (event.y - center_y) / 50

    move_ball(delta_x, delta_y, event.x, event.y)


def move_ball(delta_x, delta_y, target_x, target_y):
    c.move(ball, delta_x, delta_y)

    current_coords = c.coords(ball)
    center_x = (current_coords[0] + current_coords[2]) / 2
    center_y = (current_coords[1] + current_coords[3]) / 2

    # Проверяем, достиг ли круг конечной точки
    if abs(center_x - target_x) > abs(delta_x) or abs(
        center_y - target_y
    ) > abs(delta_y):
        # Если не достиг, продолжаем движение
        root.after(10, lambda: move_ball(delta_x, delta_y, target_x, target_y))


if __name__ == "__main__":
    root = Tk()
    c = Canvas(root, width=300, height=200, bg="white")
    c.pack()
    ball = c.create_oval(0, 100, 40, 140, fill="green")

    c.bind("<Button-1>", motion)

    root.mainloop()
```

(variable) current_coords: list[float]

Рисунок 6.1 – Код программы

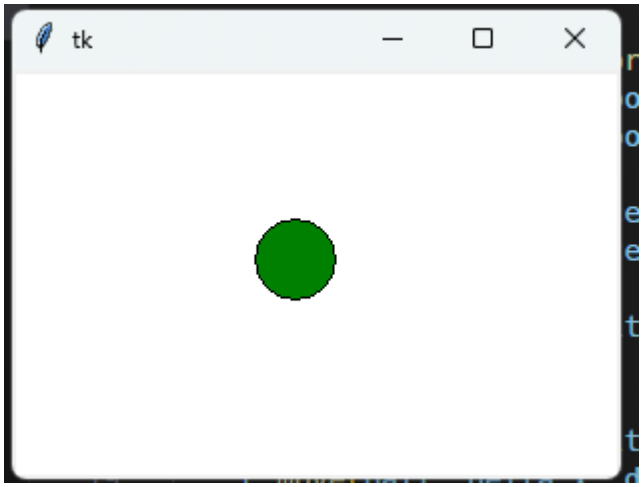

Рисунок 6.2 - Результат

Ответы на контрольные вопросы

1. Назначение виджета ListBox:

Виджет «ListBox» в Tkinter используется для отображения списка элементов, из которых пользователи могут выбирать один или несколько. Этот виджет подходит для отображения простого списка, например, файлов или других данных.

2. Связывание события или действия с виджетом Tkinter:

События можно связать с виджетом Tkinter с помощью метода «bind». Вы связываете тип события (например, нажатие клавиши или клик мыши) с функцией-обработчиком, которая вызывается при наступлении этого события.

Пример связывания события:

3. Типы событий в Tkinter:

- Клавиатурные события: например, нажатие клавиш («<KeyPress>«, «<KeyRelease>«, «<Return>«, «<Escape>« и т.д.).

- Мышиные события: клик мыши («<Button-1>«, «<Button-2>«, «<Button-3>«), движение мыши («<Motion>«), вход и выход курсора мыши в область виджета («<Enter>«, «<Leave>«).

- События фокуса: получение и потеря фокуса виджетом («<FocusIn>«, «<FocusOut>«).

4. Обработка событий в Tkinter:

Обработка событий осуществляется через функции-обработчики (callback functions), которые вызываются автоматически при возникновении событий. Эти функции имеют параметр, который представляет объект события («event»), содержащий информацию о нем, такую как координаты мыши или нажатая клавиша.

5. Обработка событий мыши в Tkinter:

События мыши обрабатываются путем присвоения обработчиков определенным событиям мыши.

6. Отображение графических примитивов в Tkinter:

Графические примитивы в Tkinter отображаются с помощью холста («Canvas»), который предоставляет методы для рисования примитивов, такие как линии, прямоугольники, эллипсы и полигоны.

7. Основные методы для отображения графических примитивов в Tkinter:

- «create_line()»: рисует линию.
- «create_oval()»: рисует эллипс или круг.
- «create_rectangle()»: рисует прямоугольник.
- «create_polygon()»: рисует полигон.
- «create_text()»: отображает текст.
- «create_arc()»: рисует дугу.
- «create_image()»: размещает изображение.

8. Обращение к ранее созданным фигурам на холсте:

При создании графического примитива на холсте «Canvas», метод возвращает идентификатор, который можно использовать для дальнейшей работы с этим примитивом. Используя идентификатор, можно изменять свойства фигуры, перемещать или удалять её с помощью методов «itemconfig», «move», «delete» и так далее.

9. Назначение тэгов в Tkinter:

Тэги в Tkinter используются для группировки и управления одним или несколькими элементами. Например, в виджете «Text», тэги позволяют применять стилевые изменения (например, изменение шрифта или цвета) ко всему тексту с данным тэгом. В холсте «Canvas», тэги позволяют управлять группами объектов, например, перемещать или изменять их сразу целиком.