МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций Институт цифрового развития

ОТЧЁТ

по лабораторной работе

Дисциплина: «Объектно – ориентированное программирование»

Выполнил: студент 3 курса
группы ИВТ-б-о-21-1
Мальцев Николай Артемович

Ставрополь 2023

Обработка событий и рисование в Tkinter

Цель работы: приобретение навыков улучшения графического интерфейса пользователя GUI с помощью обработки событий и рисования, реализованных в пакете Tkinter языка программирования Python версии 3.x.

Ход работы:

Задание 1. Решите задачу с использованием Tkinter: 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.

Листинг программы:

```
import tkinter as tk

class ShoppingListApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Список покупок")

        # Создаем два списковых виджета
        self.available_items_listbox = tk.Listbox(root, selectmode=tk.MULTIPLE)
        self.available_items_listbox.pack(side=tk.LEFT, padx=10)

        self.shopping_listbox = tk.Listbox(root, selectmode=tk.MULTIPLE)
        self.shopping_listbox.pack(side=tk.RIGHT, padx=10)

        # Наполняем первый список товарами
        available_items = ["Хлеб", "Молоко", "Яйца", "Фрукты", "Овощи", "Мясо"]
        for item in available_items:
            self.available_items_listbox.insert(tk.END, item)

        # Создаем кнопки для перемещения товаров
        self.add_button = tk.Button(root, text="Добавить в список",
command=self.add_to_shopping_list)
        self.add_button.pack(pady=10)

        self.remove_button = tk.Button(root, text="Убрать из списка",
command=self.remove_from_shopping_list)
        self.remove_button.pack(pady=10)

    def add_to_shopping_list(self):
        selected_items = self.available_items_listbox.curselection()
        for index in selected_items[::-1]:  # Перебираем в обратном порядке,
чтобы удаление не нарушало индексы
```

```
            item = self.available_items_listbox.get(index)
            self.shopping_listbox.insert(tk.END, item)
            self.available_items_listbox.delete(index)

    def remove_from_shopping_list(self):
        selected_items = self.shopping_listbox.curselection()
        for index in selected_items[::-1]:
            item = self.shopping_listbox.get(index)
            self.available_items_listbox.insert(tk.END, item)
            self.shopping_listbox.delete(index)

if __name__ == "__main__":
    root = tk.Tk()
    app = ShoppingListApp(root)
    root.mainloop()
```

Результат работы программы:

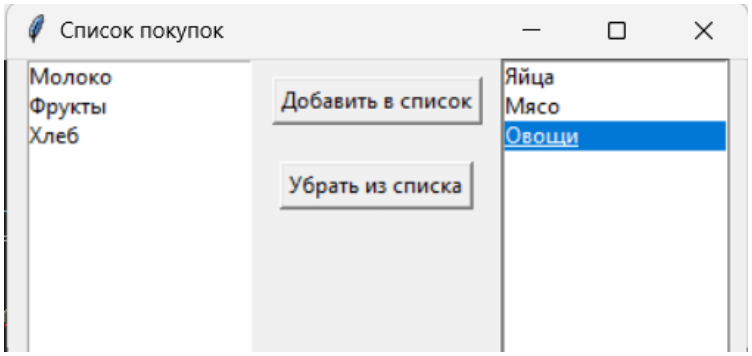

Рисунок 1. Пример работы программы к первому заданию

Задание 2. Решите задачу с использованием Tkinter: напишите программу по следующему описанию. Нажатие Enter в однострочном текстовом поле приводит к перемещению текста из него в список (экземпляр Listbox). При двойном клике (<Double-Button-1>) по элементу-строке списка, она должна копироваться в текстовое поле.

Листинг программы:

```
import tkinter as tk

class TextListApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Текст и список")

        # Создаем однострочное текстовое поле
        self.text_entry = tk.Entry(root)
        self.text_entry.pack(pady=10)
        self.text_entry.bind("<Return>", self.add_to_list)
```

```
        # Создаем список
        self.text_listbox = tk.Listbox(root)
        self.text_listbox.pack(expand=True, fill=tk.BOTH)
        self.text_listbox.bind("<Double-Button-1>", self.copy_to_text_entry)

    def add_to_list(self, event):
        text = self.text_entry.get()
        if text:
            self.text_listbox.insert(tk.END, text)
            self.text_entry.delete(0, tk.END)

    def copy_to_text_entry(self, event):
        selected_index = self.text_listbox.curselection()
        if selected_index:
            selected_text = self.text_listbox.get(selected_index)
            self.text_entry.delete(0, tk.END)
            self.text_entry.insert(0, selected_text)

if __name__ == "__main__":
    root = tk.Tk()
    app = TextListApp(root)
    root.mainloop()
```

Результат работы программы:

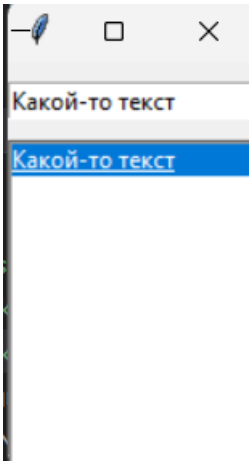

Рисунок 2. Пример работы программы ко второму заданию

Задание 3. Решите задачу с использованием Tkinter: 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. Событие получения фокуса обозначается как <FocusIn> , потери – как <FocusOut> . Для справки: фокус перемещается по виджетам при нажатии Tab,

Ctrl+Tab, Shift+Tab, а также при клике по ним мышью (к кнопкам последнее не относится).

Листинг программы:

```
import tkinter as tk

class ResizableTextApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Размеризуемый текст")

        # Переменные для хранения размеров текстового поля
        self.width_var = tk.StringVar()
        self.height_var = tk.StringVar()

        # Создаем однострочные текстовые поля для ввода размеров
        self.width_entry = tk.Entry(root, textvariable=self.width_var)
        self.width_entry.pack(side=tk.LEFT, padx=10)
        self.width_entry.bind("<Return>", self.resize_text)

        self.height_entry = tk.Entry(root, textvariable=self.height_var)
        self.height_entry.pack(side=tk.LEFT, padx=10)
        self.height_entry.bind("<Return>", self.resize_text)

        # Кнопка для изменения размера текстового поля
        self.resize_button = tk.Button(root, text="Изменить размер",
command=self.resize_text)
        self.resize_button.pack(side=tk.LEFT, padx=10)

        # Многострочное текстовое поле
        self.text_widget = tk.Text(root, wrap=tk.WORD, bg="lightgrey")
        self.text_widget.pack(expand=True, fill=tk.BOTH)
        self.text_widget.bind("<FocusIn>", self.on_focus_in)
        self.text_widget.bind("<FocusOut>", self.on_focus_out)

    def resize_text(self, event=None):
        try:
            width = int(self.width_var.get())
            height = int(self.height_var.get())
            self.text_widget.config(width=width, height=height)
        except ValueError:
            pass

    def on_focus_in(self, event):
        self.text_widget.config(bg="white")

    def on_focus_out(self, event):
        self.text_widget.config(bg="lightgrey")

if __name__ == "__main__":
    root = tk.Tk()
```

```
    app = ResizableTextApp(root)
    root.mainloop()
```

Результат работы программы:

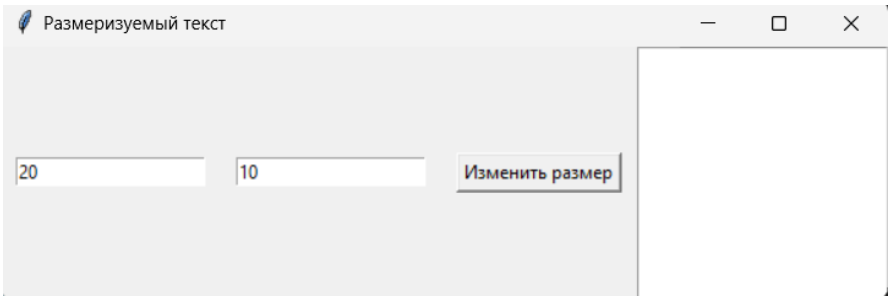

Рисунок 3. Пример работы программы к третьему заданию

Задание 4. Нарисуйте изображение, как на примере. Для отрисовки травы используйте цикл.

Листинг программы:

```
import tkinter as tk

class LandscapeApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Пейзаж")

        self.canvas = tk.Canvas(root, width=400, height=300, bg="white")
        self.canvas.pack()

        # Рисуем траву
        self.draw_grass()

        # Рисуем силуэт домика
        self.draw_house()

        # Рисуем солнце
        self.draw_sun()

    def draw_grass(self):
        grass_color = "green"

        # Рисуем траву в виде полос
        for y in range(200, 300, 10):
            self.canvas.create_rectangle(0, y, 400, y + 5, fill=grass_color)

    def draw_house(self):
        house_color = "blue"
        roof_color = "red"
```

```
        # Рисуем стены домика
        self.canvas.create_rectangle(100, 100, 300, 200, fill=house_color)

        # Рисуем крышу домика
        self.canvas.create_polygon(100, 100, 200, 50, 300, 100, fill=roof_color)

    def draw_sun(self):
        sun_color = "orange"
        sun_radius = 30
        sun_center = (370, 30)

        # Рисуем круг (солнце)
        self.canvas.create_oval(sun_center[0] - sun_radius, sun_center[1] -
sun_radius,
                                sun_center[0] + sun_radius, sun_center[1] +
sun_radius, fill=sun_color)

if __name__ == "__main__":
    root = tk.Tk()
    app = LandscapeApp(root)
    root.mainloop()
```

Результат работы программы:

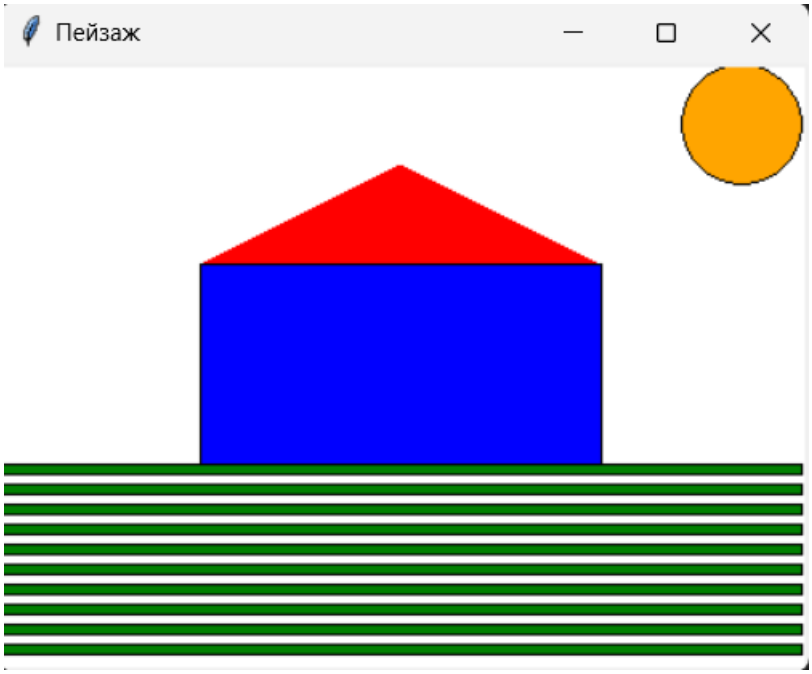

Рисунок 4. Пример работы программы к четвёртому заданию

Задание 5.

Листинг программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import tkinter as tk
```

```python
class TextListApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Текст и список")

        # Создаем однострочное текстовое поле
        self.text_entry = tk.Entry(root)
        self.text_entry.pack(pady=10)
        self.text_entry.bind("<Return>", self.add_to_list)

        # Создаем список
        self.text_listbox = tk.Listbox(root)
        self.text_listbox.pack(expand=True, fill=tk.BOTH)
        self.text_listbox.bind("<Double-Button-1>", self.copy_to_text_entry)

    def add_to_list(self, event):
        text = self.text_entry.get()
        if text:
            self.text_listbox.insert(tk.END, text)
            self.text_entry.delete(0, tk.END)

    def copy_to_text_entry(self, event):
        selected_index = self.text_listbox.curselection()
        if selected_index:
            selected_text = self.text_listbox.get(selected_index)
            self.text_entry.delete(0, tk.END)
            self.text_entry.insert(0, selected_text)

if __name__ == "__main__":
    root = tk.Tk()
    app = TextListApp(root)
    root.mainloop()
```

Результат работы программы:

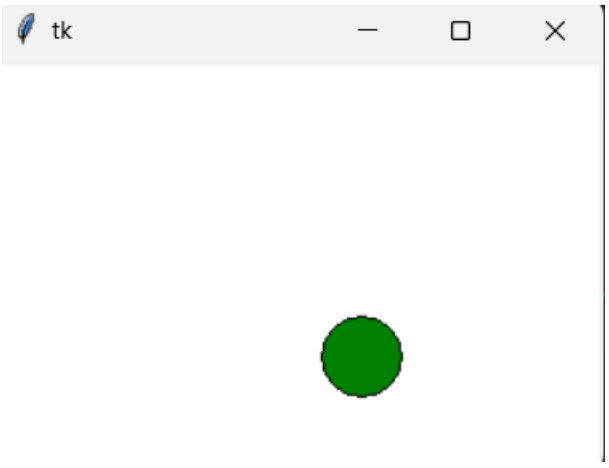

Рисунок 5. Пример работы программы к пятому примеру

Ответы на контрольные вопросы:

1. Каково назначение виджета ListBox?

Виджет ListBox в Tkinter предназначен для отображения списка элементов, из которых пользователь может выбирать один или несколько. Этот виджет предоставляет прокручиваемый список элементов.

2. Каким образом осуществляется связывание событие или действие с виджетом Tkinter?

Связывание событий в Tkinter осуществляется с использованием метода bind(). Этот метод позволяет привязывать функции-обработчики к определенным событиям, таким как нажатие кнопки, перемещение мыши и другие.

3. Какие существуют типы событий в Tkinter? Приведите примеры.

Существует множество типов событий в Tkinter. Некоторые из них включают:

Button-1: Левая кнопка мыши.

Button-2: Средняя кнопка мыши (если она есть).

Button-3: Правая кнопка мыши.

Motion: Движение мыши.

KeyPress: Нажатие клавиши на клавиатуре.

KeyRelease: Отпускание клавиши на клавиатуре.

4. Как обрабатываются события в Tkinter?

События обрабатываются путем привязки функций-обработчиков к определенным событиям с использованием метода bind() или через метод after() для периодического выполнения действий.

5. Как обрабатываются события мыши в Tkinter?

События мыши, такие как нажатие кнопок или движение мыши, обрабатываются путем привязки функций-обработчиков к соответствующим событиям с использованием метода bind().

6. Каким образом можно отображать графические примитивы в Tkinter?

Для отображения графических примитивов в Tkinter используется виджет Canvas. Этот виджет позволяет рисовать линии, прямоугольники, окружности и другие графические элементы.

7. Перечислите основные методы для отображения графических примитивов в Tkinter.

Некоторые основные методы для работы с графическими

примитивами на холсте (Canvas):

create_line(): Создает линию.

create_rectangle(): Создает прямоугольник.

create_oval(): Создает овал.

create_text(): Создает текст.

8. Каким образом можно обратиться к ранее созданным фигурам на холсте?

Каждая фигура на холсте имеет уникальный идентификатор, который возвращается методами создания фигур. Идентификаторы могут использоваться для обращения к ранее созданным фигурам.

9. Каково назначение тэгов в Tkinter

Тэги в Tkinter используются для группировки и идентификации набора объектов на холсте. Они позволяют применять действия к определенным группам объектов. Тэги могут быть присвоены при создании объекта на холсте или позднее с использованием метода addtag_withtag().