МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет»

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 4.8 «Обработка событий и рисование в Tkinter»

по дисциплине «Объектно-ориентированное программирование»

Выполнил студент группы ИВТ-б-о-21-1
Павленко М.С. « »_____20__г.
Подпись студента_____
Работа защищена « »_____20__г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

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:

1. Клонировал общедоступный репозиторий GitHub на свой локальный сервер.

2. Изучил теоретический материал и приступил к выполнению заданий.

```python
# Создаем функцию для добавления продуктов в список
def add_item():
    product = []
    select = list(lbox_first.curselection())
    select.reverse()
    for item in select:
        op = lbox_first.get(item)
        product.append(op)
    for val in product:
        lbox_second.insert(0, val)
    for k in select:
        lbox_first.delete(k)


# Создаем функцию для удаления из списка
def delete_item():
    product = []
    select = list(lbox_second.curselection())
    select.reverse()
    for item in select:
        op = lbox_second.get(item)
        product.append(op)
    for val in product:
        lbox_first.insert(0, val)
    for k in select:
        lbox_second.delete(k)
```

Рисунок 1 – Код программы первого задания

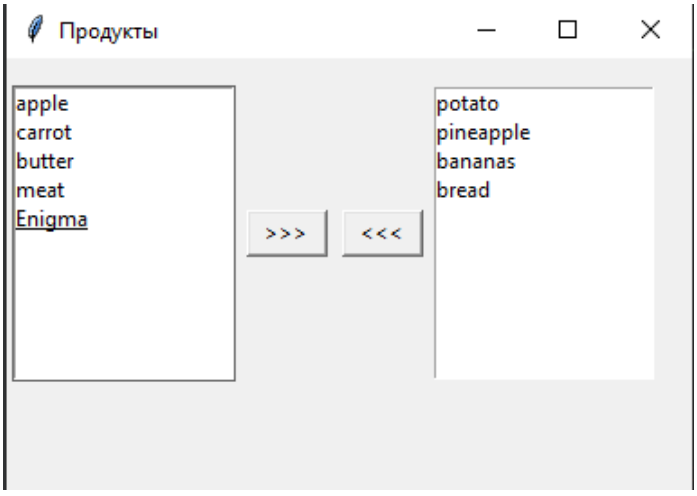

Рисунок 2 – Графический интерфейс программы

3. Выполнил второе задание.

```
if __name__ == "__main__":
    root = Tk()
    root.geometry("200x200+90+90")
    root.title("Task2")

    ent1 = Entry()
    lb1 = Listbox()

    ent1.pack()
    lb1.pack()
    ent1.bind('<Return>', lambda e:  lb1.insert(0, ent1.get()))
    lb1.bind('<Double-Button-1>', lambda e: ent1.insert(0, lb1.get(lb1.curselection())))

    root.mainloop()
```

Рисунок 3 – Код программы второго задания

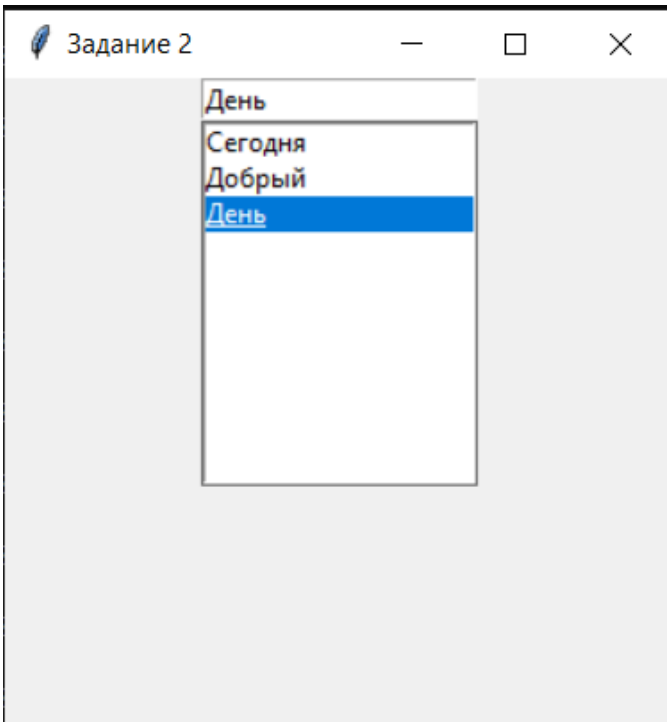

Рисунок 4 – Графический интерфейс программы

4. Выполнил третью задачу.

```python
def button_click(event):
    txt1['width'] = ent1.get()
    txt1['height'] = ent2.get()


def focus_change(event, color):
    txt1['bg'] = color


if __name__ == "__main__":
    root = Tk()
    root.geometry("500x300+300+300")
    root.title("Задание 3")

    f = Frame()
    f.pack()
    ent1 = Entry(f, width=3)
    ent2 = Entry(f, width=3)
    bt1 = Button(f, text='Изменить')
    txt1 = Text(width=25, height=25, bg='lightgrey')

    bt1.bind('<Button-1>', button_click)
    ent1.bind('<Return>', button_click)
    ent2.bind('<Return>', button_click)
    bt1.bind('<Return>', button_click)
    root.bind('<Return>', button_click)
```

Рисунок 5 – Код программы третьего задания

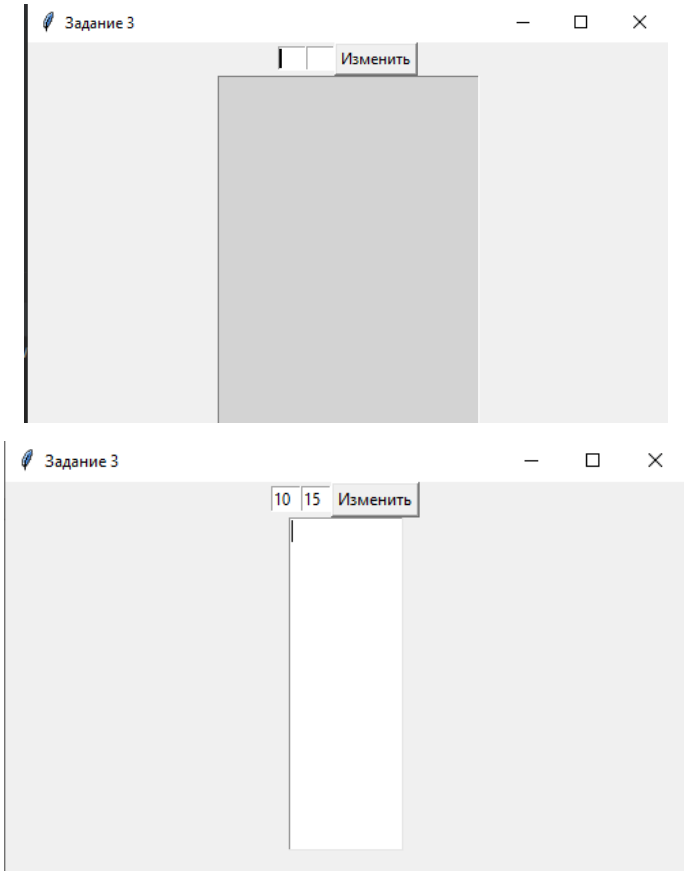

Рисунок 6 – Результат работы программы

5. Выполнил четвертую задачу.

```
if __name__ == '__main__':
    root = Tk()
    root.geometry("500x300+300+300")
    root.title("Task4")
    c = Canvas(root, width=200, height=200, bg='white')
    c.pack()
    c.create_rectangle(50, 90, 140, 175,
                       fill='#7fc7ff', outline='#7fc7ff')
    c.create_polygon(30, 90, 160, 90, 95, 40,
                     fill='#7fc7ff', outline='#7fc7ff')
    c.create_oval(150, 10, 190, 50,
                  fill='yellow', outline='yellow')
    x = -20
    while x < 200:
        c.create_arc(x, 170, x + 40, 250, outline='green',
                     style=ARC, start=200, extent=-80,
                     width=2)
        x += 10
    root.mainloop()
```

Рисунок 7 – Код программы

Выполнил пятую задачу.

```
def click(event):
    c.x = event.x + c.radius
    c.y = event.y + c.radius
    motion()


def motion():
    x = c.coords(c.ball)[2]
    y = c.coords(c.ball)[3]
    if c.x == x and c.y == y:
        return
    if c.x < x:
        c.move(c.ball, -1, 0)
    if c.x > x:
        c.move(c.ball, 1, 0)
    if c.y < y:
        c.move(c.ball, 0, -1)
    if c.y > y:
        c.move(c.ball, 0, 1)
    root.after(10, motion)

```

Рисунок 9 – Код программы

Контрольные вопросы:

1. Каково назначение виджета ListBox?

От класса Listbox создаются списки – виджеты, внутри которых в столбик перечисляются элементы. При этом можно выбирать один или множество элементов списка

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

Events - события. В tkinte с помощью метода bind() между собой связываются виджет, событие и действие. Например, виджет – кнопка, событие – клик по ней левой кнопкой мыши, действие – отправка сообщения. Другой пример: виджет – текстовое поле, событие – нажатие Enter, действие – получение текста из поля методом get() для последующей обработки программой.

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

Можно выделить три основных типа событий: производимые мышью, нажатиями клавиш на клавиатуре, а также события, возникающие в результате изменения виджетов. Нередко обрабатываются сочетания. Например, клик мышью с зажатой клавишей на клавиатуре.

4. Как обрабатываются события в Tkinter?

Все события обрабатываются методом класса print_event(), который выводит тип события и положение мыши в консоли. Можете поэкспериментировать, нажимая на зеленую рамку мышью и двигая ею, пока она будет выводить сообщения события.

5. Как обрабатываются события мыши в Tkinter?

– <Button-1> – клик левой кнопкой мыши;

– <Button-2> – клик средней кнопкой мыши;

– <Button-3> – клик правой кнопкой мыши;

– <Double-Button-1> – двойной клик левой кнопкой мыши;

– <Motion> – движение мыши.

6. Каким образом можно отображать графические примитивы в Tkinter?

В tkinter от класса Canvas создаются объекты-холсты, на которых можно "рисовать", размещая различные фигуры и объекты. Делается это с помощью вызовов соответствующих методов. При создании экземпляра Canvas необходимо указать его ширину и высоту. При размещении геометрических примитивов и других объектов указываются их координаты на холсте. Точкой отсчета является верхний левый угол.

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

Методом create_polygon рисуется произвольный многоугольник путем задания координат каждой его точки.

Метод create_oval создает эллипсы. При этом задаются координаты гипотетического прямоугольника, описывающего эллипс. Если нужно получить круг, то соответственно описываемый прямоугольник должен быть квадратом.

Более сложные для понимания фигуры получаются при использовании метода create_arc . В зависимости от значения опции style можно получить сектор (по умолчанию), сегмент (CHORD) или дугу (ARC). Также как в случае create_oval координаты задают прямоугольник, в который вписана окружность (или эллипс), из которой "вырезают" сектор, сегмент или дугу. Опции start присваивается градус начала фигуры, extent определяет угол поворота.

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

В Tkinter существует два способа "пометить" фигуры, размещенные на холсте, – это идентификаторы и теги. Первые всегда уникальны для каждого объекта. Два объекта не могут иметь одни и тот же идентификатор. Теги не уникальны. Группа объектов на холсте может иметь один и тот же тег. Это дает возможность менять свойства всей группы. Отдельно взятая фигура на Canvas может иметь как идентификатор, так и тег.

9. Каково назначение тэгов в Tkinter?

В отличие от идентификаторов, которые являются уникальными для каждого объекта, один и тот же тег может присваиваться разным объектам. Дальнейшее обращение к такому тегу позволит изменить все объекты, в которых он был указан. Метод tag_bind позволяет привязать событие (например, щелчок кнопкой мыши) к определенной фигуре на Canvas. Таким образом, можно реализовать обращение к различным областям холста с помощью одного и того же события.