Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ

По лабораторной работе №8

Дисциплины «Объектно-ориентированное программирование»

Выполнил:

Пустяков Андрей Сергеевич

3 курс, группа ИВТ-б-о-22-1,

09.03.01 «Информатика и вычислительная техника (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____

(подпись)

Руководитель практики:

Воронкин Р. А., доцент департамента цифровых и робототехнических систем и электроники института перспективной инженерии

_____

(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Обработка событий и рисование в Tkinter.

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

Ход работы:

Выполнение заданий:

Задание 7.

Необходимо написать 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.

Код программы задания 7:

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

# Необходимо написать 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.

import tkinter as tk
from tkinter import Listbox, Button, MULTIPLE


class ProductTransferApp:
    """
    Приложение для управления перечнем товаров.
    Позволяет перемещать товары из списка товаров в корзину и обратно.
    """

    def __init__(self, root):
        """
        Инициализирует главное окно приложения и виджеты.

        :param root: Главное окно tkinter.
        """

        self.root = root
        self.root.title("Перечень товаров")

        self.create_widgets()
        self.populate_product_list()
```

```
    def create_widgets(self):
        """
        Создает виджеты приложения: списки и кнопки управления.
        """

        # Список товаров
        self.product_list = Listbox(self.root, selectmode=MULTIPLE, width=30,
height=15)
        self.product_list.grid(row=0, column=0, padx=10, pady=10)

        # Кнопки управления
        self.add_button = Button(self.root, text=">>>",
command=self.add_to_cart)
        self.add_button.grid(row=0, column=1, padx=5, pady=5)

        self.remove_button = Button(self.root, text="<<<",
command=self.remove_from_cart)
        self.remove_button.grid(row=1, column=1, padx=5, pady=5)

        # Список покупок
        self.cart_list = Listbox(self.root, selectmode=MULTIPLE, width=30,
height=15)
        self.cart_list.grid(row=0, column=2, padx=10, pady=10)

    def populate_product_list(self):
        """
        Заполняет список товаров предустановленными значениями.
        """

        products = [
            "Гвозди",
            "Шурупы",
            "Краска",
            "Кисти",
            "Молоток",
            "Отвёртка",
            "Цемент",
            "Линейка",
            "Пила",
            "Перчатки"
        ]
        for product in products:
            self.product_list.insert(tk.END, product)

    def add_to_cart(self):
        """
        Перемещает выделенные товары из списка товаров в корзину.
        """

        selected_items = self.product_list.curselection()
        for index in reversed(selected_items):  # Перебор в обратном порядке
для корректного удаления
            item = self.product_list.get(index)
            self.cart_list.insert(tk.END, item)
            self.product_list.delete(index)

    def remove_from_cart(self):
        """
        Перемещает выделенные товары из корзины обратно в список товаров.
        """

        selected_items = self.cart_list.curselection()
        for index in reversed(selected_items):  # Перебор в обратном порядке
для корректного удаления
```

```
            item = self.cart_list.get(index)
            self.product_list.insert(tk.END, item)
            self.cart_list.delete(index)


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

Результаты работы программы задания 7, выбор товаров и перемещение их из одного поля и обратно при необходимости (рис. 1).

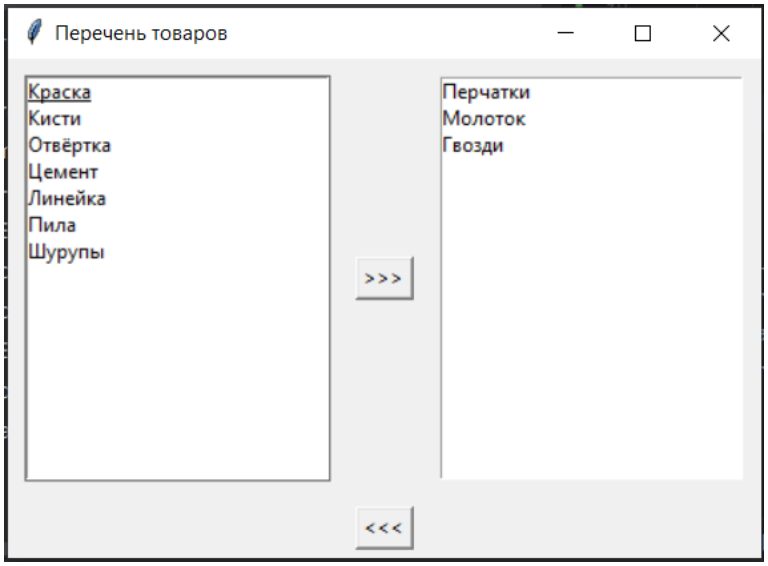

Рисунок 1 – Результаты работы программы задания 7

Для проверки перемещения продуктов из одного перечня в другой, проверку на ожидаемые элементы в списке был создан unit-тест для программы задания 7. Результаты тестирования задания 7 (рис. 2).

```
tests/test_task_1.py::TestProductTransferApp::test_add_to_cart PASSED
tests/test_task_1.py::TestProductTransferApp::test_initial_product_list PASSED
tests/test_task_1.py::TestProductTransferApp::test_remove_from_cart PASSED

============================================================================== 3 passed in 0.56s
```

Рисунок 2 – Результаты тестирования программы задания 7

Задание 8.

Необходимо написать 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.

Код программы по описанию задания 8:

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

# Необходимо написать 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.

import tkinter as tk


class TextInputListApp:
    """
    Класс приложения для работы с текстовым полем и списком.
    """

    def __init__(self, root):
        """
        Инициализация главного окна и создание виджетов.

        :param root: Главное окно Tkinter.
        """

        self.root = root
        self.root.title("Текстовое поле и список")

        self.initialize_widgets()

    def initialize_widgets(self):
        """
        Создает виджеты: текстовое поле и список.
        """

        # Текстовое поле для ввода
        self.entry_field = tk.Entry(self.root, width=40)
        self.entry_field.grid(row=0, column=0, padx=10, pady=10)

        # Привязываем нажатие Enter к методу add_text_to_list
        self.entry_field.bind("<Return>", self.add_text_to_list)

        # Список для отображения текста
        self.list_box = tk.Listbox(self.root, width=40, height=15)
        self.list_box.grid(row=1, column=0, padx=10, pady=10)

        # Привязываем двойной клик к методу copy_text_to_entry
        self.list_box.bind("<Double-Button-1>", self.copy_text_to_entry)

    def add_text_to_list(self, event=None):
        """
        Добавляет текст из текстового поля в список.

        :param event: Событие, связанное с нажатием клавиши Enter.
        """

        text = self.entry_field.get().strip()
        if text:  # Проверяем, что текст не пустой
```

```
            self.list_box.insert(tk.END, text)
            self.entry_field.delete(0, tk.END)  # Очищаем текстовое поле

    def copy_text_to_entry(self, event=None):
        """
        Копирует выделенный элемент из списка в текстовое поле.

        :param event: Событие, связанное с двойным кликом.
        """

        selected_indices = self.list_box.curselection()
        if selected_indices:  # Проверяем, что есть выделенный элемент
            text = self.list_box.get(selected_indices[0])
            self.entry_field.delete(0, tk.END)
            self.entry_field.insert(0, text)


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

Результаты работы программы задания 8, добавленный текст в список и копирование этого же элемента обратно в текстовое поле (рис. 3).

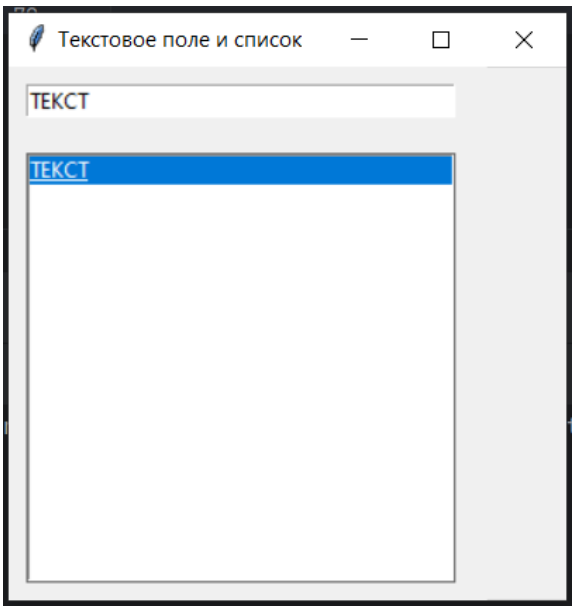

Рисунок 3 – Результаты работы программы задания 8

Для тестирования работоспособности программы был создан unit-тест для программы задания 8. Результаты тестирования программы задания 8 (рис. 4).

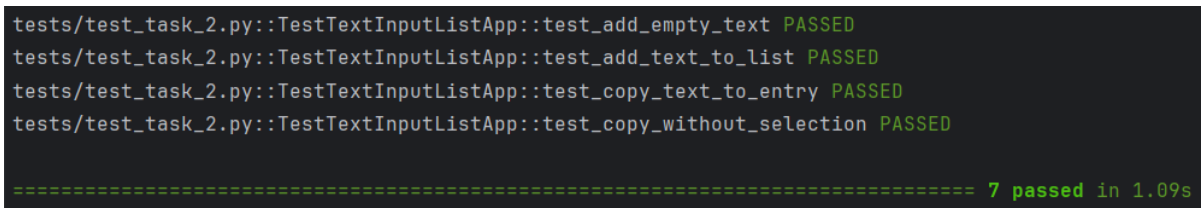

Рисунок 4 – Результаты тестирования программы задания 8

Задание 9.

Необходимо написать программу по следующему 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 светло-серый (lightgrey), когда поле не в фокусе, и белый, когда имеет фокус.

Код программы задания 9 по приведенному описанию:

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

"""
Необходимо написать программу по следующему 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.
"""

import tkinter as tk


class DynamicTextFieldApp:
    """
    Класс приложения для управления размерами и фокусом многострочного
текстового поля.
    """

    def __init__(self, root):
        """
        Инициализирует главное окно и создает виджеты приложения.

        :param root: Главное окно Tkinter.
        """

        self.root = root
        self.root.title("Управление текстовым полем")

        self.initialize_widgets()

    def initialize_widgets(self):
        """
        Инициализирует и размещает виджеты в главном окне приложения.
        """
```

```
        # Поле для ввода ширины текстового поля
        self.width_input = tk.Entry(self.root, width=5)
        self.width_input.insert(0, "30")  # Устанавливаем значение по
умолчанию
        self.width_input.grid(row=0, column=0, padx=5, pady=5)

        # Поле для ввода высоты текстового поля
        self.height_input = tk.Entry(self.root, width=5)
        self.height_input.insert(0, "10")  # Устанавливаем значение по
умолчанию
        self.height_input.grid(row=1, column=0, padx=5, pady=5)

        # Кнопка для изменения размеров текстового поля
        self.update_button = tk.Button(self.root, text="Изменить",
command=self.update_text_field)
        self.update_button.grid(row=0, column=1, rowspan=2, padx=5, pady=5)

        # Многострочное текстовое поле
        self.text_field = tk.Text(self.root, width=30, height=10,
bg="lightgrey")
        self.text_field.grid(row=2, column=0, columnspan=2, padx=5, pady=5)

        # Привязка событий изменения фокуса
        self.text_field.bind("<FocusIn>", self.set_focus_background)
        self.text_field.bind("<FocusOut>", self.remove_focus_background)

        # Привязка клавиши Enter для изменения размеров
        self.width_input.bind("<Return>", self.update_text_field)
        self.height_input.bind("<Return>", self.update_text_field)

    def update_text_field(self, event=None):
        """
        Изменяет размеры текстового поля на основе введенных данных.

        :param event: Событие, связанное с нажатием клавиши Enter.
        """

        try:
            new_width = int(self.width_input.get())
            new_height = int(self.height_input.get())
            self.text_field.config(width=new_width, height=new_height)
        except ValueError:
            # Игнорируем ошибки, если введены некорректные данные
            pass

    def set_focus_background(self, event):
        """
        Устанавливает белый цвет фона для текстового поля при получении
фокуса.

        :param event: Событие получения фокуса.
        """

        self.text_field.config(bg="white")

    def remove_focus_background(self, event):
        """
        Устанавливает светло-серый цвет фона для текстового поля при потере
фокуса.

        :param event: Событие потери фокуса.
        """
```

```
        self.text_field.config(bg="lightgrey")


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

Результаты работы программы, регулировка длины и ширины текстового поля, изменение цвета текстового поля в зависимости от фокуса (рис. 5).

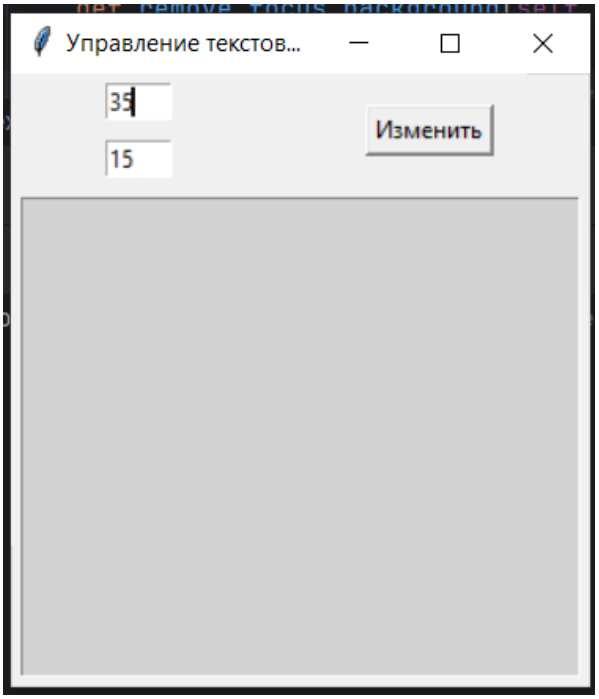

Рисунок 5 – Результаты работы программы задания 9

Для тестирования работоспособности программы задания 9 был создан unit-тест. Результаты тестирования программы задания 9 (рис. 6).

```
tests/test_task_3.py::TestDynamicTextFieldApp::test_initial_sizes PASSED
tests/test_task_3.py::TestDynamicTextFieldApp::test_invalid_input_handling PASSED
tests/test_task_3.py::TestDynamicTextFieldApp::test_update_text_field PASSED

============================================================================== 10 passed in 1.95s
```

Рисунок 6 – Успешные результаты тестирования задания 9

Задание 10.

Необходимо написать программу, которая нарисует на холсте следующее изображение (рис. 7) (для рисования травы необходимо использовать цикл).

Рисунок 7 – Примерный холст с рисунком

Код программы задания 10, позволяющий рисовать подобный рисунок:

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

"""
Необходимо написать программу, которая рисует дом,
 солнце и траву циклом на холсте.
"""

import tkinter as tk


def create_landscape(canvas, canvas_width, canvas_height):
    """
    Рисует на холсте небо, дом, солнце и траву.

    :param canvas: Холст.
    :param canvas_width: Ширина холста.
    :param canvas_height: Высота холста.
    """

    # Небо
    canvas.create_rectangle(0, 0, canvas_width, canvas_height, fill="white",
outline="")

    # Крыша
    canvas.create_polygon(
        canvas_width // 2 - 70, canvas_height // 2,  # Левая точка крыши
        canvas_width // 2 + 70, canvas_height // 2,  # Правая точка крыши
        canvas_width // 2, canvas_height // 2 - 100,  # Верхняя точка крыши
        fill="lightblue", outline=""
    )

    # Каркас
    canvas.create_rectangle(
        canvas_width // 2 - 50, canvas_height // 2,
        canvas_width // 2 + 50, canvas_height // 2 + 100,
        fill="lightblue", outline=""
    )
```

```
    # Солнце
    canvas.create_oval(
        canvas_width - 100, 50,
        canvas_width - 50, 100,
        fill="orange", outline=""
    )

    # Трава
    for position_x in range(0, canvas_width, 15):
        canvas.create_arc(
            position_x, canvas_height - 80, position_x + 30, canvas_height -
20,
            start=120, extent=60,
            style=tk.ARC, outline="green", width=2
        )


if __name__ == "__main__":
    """
    Основная функция программы.
    Настраивает окно и вызывает функцию для рисования.
    """

    # Настройка окна
    root = tk.Tk()
    root.title("Сцена с домом и травой")

    # Размеры холста
    canvas_width, canvas_height = 400, 300
    canvas = tk.Canvas(root, width=canvas_width, height=canvas_height)
    canvas.pack()

    # Рисуем сцену
    create_landscape(canvas, canvas_width, canvas_height)

    # Запуск основного цикла Tkinter
    root.mainloop()
```

Результаты работы программы, отрисованный дом, небо, солнце и трава на холсте (рис. 8).

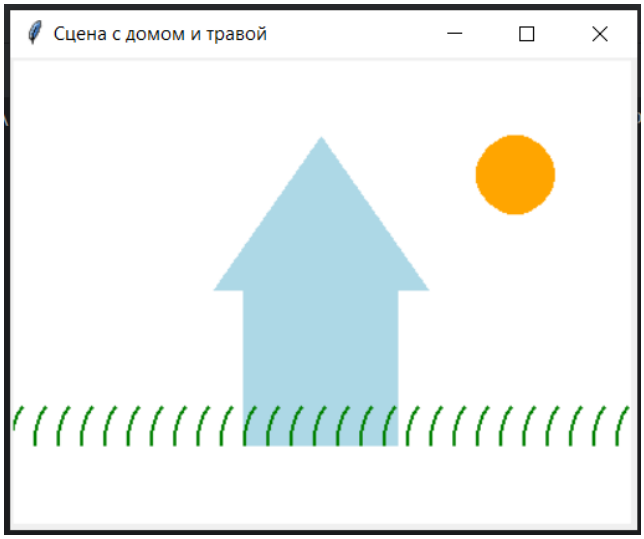

Рисунок 8 – Результаты работы программы задания 10

Задание 11.

Необходимо по примеру кода создать программу анимации перемещения круга, который движется плавно в ту сторону, в которую пользователь кликает левой кнопкой мыши.

Код программы задания 11:

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

"""
Необходимо по примеру кода создать программу анимации перемещения
круга, который движется плавно в ту сторону, в которую пользователь
кликает левой кнопкой мыши.
"""

from tkinter import Tk, Canvas


class BallMover:
    """
    Класс, управляющий перемещением круга на холсте.
    """

    def __init__(self, canvas, ball):
        """
        Инициализирует объект BallMover.

        :param canvas: Экземпляр холста.
        :param ball: Круг.
        """

        self.canvas = canvas
        self.ball = ball
        self.target_x = None  # Целевая координата X
        self.target_y = None  # Целевая координата Y

    def move_to_target(self):
        """
        Перемещает круг в сторону установленной цели.
        Если цель не достигнута, продолжает двигаться с задержкой.
        """

        if self.target_x is not None and self.target_y is not None:
            # Получаем текущие координаты круга
            x1, y1, x2, y2 = self.canvas.coords(self.ball)
            ball_center_x = (x1 + x2) / 2
            ball_center_y = (y1 + y2) / 2

            # Вычисляем смещение
            dx = 1 if ball_center_x < self.target_x else -1 if ball_center_x
> self.target_x else 0
            dy = 1 if ball_center_y < self.target_y else -1 if ball_center_y
> self.target_y else 0

            # Двигаем круг, если он не достиг цели
            if dx != 0 or dy != 0:
                self.canvas.move(self.ball, dx, dy)
                self.canvas.after(10, self.move_to_target)
```

```
    def set_target(self, event):
        """
        Устанавливает новую цель для движения круга.

        :param event: Событие мыши, содержащее координаты клика.
        """

        self.target_x = event.x
        self.target_y = event.y
        self.move_to_target()


def main():
    """
    Главная функция программы. Создаёт окно с холстом и запускает анимацию.
    """

    root = Tk()
    root.title("Ball Movement")

    # Создаём холст
    canvas = Canvas(root, width=300, height=200, bg="white")
    canvas.pack()

    # Создаём круг
    ball = canvas.create_oval(100, 40, 140, 80, fill="green")

    # Создаём объект BallMover
    ball_mover = BallMover(canvas, ball)

    # Привязываем обработчик клика мыши
    canvas.bind("<Button-1>", ball_mover.set_target)

    root.mainloop()


if __name__ == "__main__":
    main()
```

Результаты работы программы задания 11, перемещение круга по холсту в зависимости от той координаты, в которой произошел щелчок мышью (рис. 9).

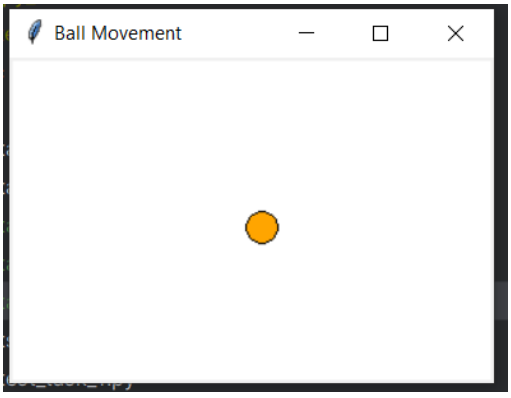

Рисунок 9 – Результаты работы программы задания 11

Ссылка на репозиторий данной лабораторной работы:

https://github.com/AndreyPust/Object-Oriented_Programming_laboratory_work_8.git

Ответы на контрольные вопросы:

1. Каково назначение виджета ListBox?

ListBox в Tkinter предназначен для отображения списка элементов, из которого пользователь может выбирать один или несколько элементов. Этот виджет часто используется для выбора из заранее заданного набора значений.

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

Связывание событий с виджетом осуществляется с помощью метода bind. Синтаксис: widget.bind(event, handler). Event — строка, обозначающая событие (например, нажатие клавиши или щелчок мыши). Handler — функция, которая будет вызвана при наступлении события.

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

Основные типы событий:

– События мыши:

<Button-1> — левый клик мыши.

<Button-2> — средний клик мыши.

<Button-3> — правый клик мыши.

<Double-Button-1> — двойной левый клик.

– События клавиатуры:

<Key> — любое нажатие клавиши.

<KeyPress-a> — нажатие клавиши "a".

<Return> — нажатие клавиши Enter.

– Системные события:

<Configure> — изменение размера окна.

<Destroy> — уничтожение виджета.

4. Как обрабатываются события в Tkinter?

Обработка событий в Tkinter организована через циклический механизм событий. Событие добавляется в очередь, и привязанный обработчик выполняется при возникновении события. Для обработки событий:

– создается функция-обработчик (callback);

– функция связывается с конкретным виджетом и событием через bind или встроенные методы (например, command для кнопки);

– когда событие происходит, вызов функции передается в mainloop.

5. Как обрабатываются события мыши в Tkinter?

События мыши обрабатываются через привязку (например, <Button-1> для левого клика).

6. Каким образом можно отображать графические примитивы в Tkinter?

Графические примитивы отображаются с использованием виджета Canvas. На холсте можно рисовать линии, круги, прямоугольники и другие фигуры. Методы для создания примитивов:

– create_line — для линий.

– create_oval — для эллипсов и окружностей.

– create_rectangle — для прямоугольников.

– create_polygon — для произвольных многоугольников.

– create_text — для текста.

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

– линия: create_line(x1, y1, x2, y2, ...);

– прямоугольник: create_rectangle(x1, y1, x2, y2, ...);

– овал/круг: create_oval(x1, y1, x2, y2, ...);

– многоугольник: create_polygon(coordinates, ...);

– текст: create_text(x, y, text=...);

– дуга: create_arc(x1, y1, x2, y2, ...).

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

Каждая фигура, созданная на холсте, получает уникальный идентификатор. Этот идентификатор можно использовать для обращения к фигуре с помощью методов:

– coords(item_id) — получить или изменить координаты фигуры.

– itemconfig(item_id, options) — изменить свойства фигуры (например, цвет).

– delete(item_id) — удалить фигуру.

9. Каково назначение тэгов в Tkinter?

Тэги (tags) используются для группировки и управления несколькими объектами на холсте одновременно. Фигуре можно присвоить один или несколько тэгов, чтобы обращаться к ним как к единой группе.

Вывод: в ходе выполнения лабораторной работы были приобретены навыки улучшения графического интерфейса пользователя GUI с помощью обработки событий и рисования, реализованных в пакете Tkinter языка программирования Python версии 3.x.