Министерство образования и науки России

Рыбинский государственный авиационный технический университет

имени П.А. Соловьева

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

ДИСЦИПЛИНА:

Объектно-ориентированное программирование

ОТЧЁТ   
ПО ЛАБАРАТОРНОЙ РАБОТЕ № 1

Выполнил Ухов Д.Д.

(фамилия, имя, отчество)

студент гр. ЗИС-23

Преподаватель Пруктишина В.А.

(фамилия, имя, отчество)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

г. Рыбинск

2024 г.

**Содержание**

[**1.** **Постановка задачи** 3](#_Toc187686175)

[**2.** **Спецификация** 4](#_Toc187686176)

[**3.** **Теоретические сведения** 7](#_Toc187686177)

[**4.** **Организация данных** 9](#_Toc187686178)

[**5.** **Текст программы** 13](#_Toc187686180)

1. **Постановка задачи**

На языке программирования Python разработать графический редактор Требования:

* должны использоваться визуальные компоненты, с кнопками и формами ввода;
* должны соблюдаться основные принципы ООП, элементы точка и линии должны быть классами, рисование — методы;
* код выкладывать на GitHub.

1. **Спецификация**

Данная программа представляет собой графический редактор, созданный с использованием библиотеки «tkinter» для языка программирования Python. Основное назначение программы — предоставление пользователю возможности рисовать графические элементы (точки, линии, круги, квадраты и треугольники), а также произвольно рисовать с помощью мыши.

**Функциональные возможности:**

1. **Выбор режима (Mode):**
   * Пользователь может выбрать несколько режимов рисования, такие как:
     1. **Рисование линий (Line):**
        + Линии создаются путем установления начальной и конечной точки, и соединяет их между собой;
        + Для установки первой точки необходимо нажать ЛКМ, провести необходимую линию, и в момент отпускания ЛКМ проставляется вторая точка;
        + Цвет задаётся через панель инструментов.
     2. **Рисование овалов (****Oval):**
        + Пользователь может создавать круги произвольного размера, задавая начальную и конечную точку;
        + Цвет задаётся через панель инструментов.
     3. **Рисование прямоугольников (Rectangle):**
        + Прямоугольники создаются путем задания начальной и конечной точки, которая определяет размеры фигуры;
        + Цвет квадрата настраиваются через панель инструментов.
     4. **Карандаш (****Pencil):**
        + Данный функционал позволяет пользователю рисовать произвольные линии, перемещая мышь с зажатой левой кнопкой мыши;
        + Цвет задаётся в панели инструментов.
2. **Выбор цвета (Color):**

* Пользователь может выбрать цвет для всех инструментов с помощью диалогового окна цветового выбора.

1. **Выбор цвета фона (Background Color):**

* Пользователь может выбрать цвет фона с помощью диалогового окна цветового выбора.

1. **Функция отмены действия (Undo):**

* Каждое нарисованное на холсте действие сохраняется в истории, и по нажатию кнопки «Undo» последнее добавленное действие удаляется с интерфейса.

**Основные интерфейсные элементы:**

* Фон (Холст) –рабочая область, где пользователь может рисовать;
* Кнопка **Mode:**
  + - Кнопка **«Line»** для построения прямых линий.
    - Кнопка **«Oval»** для рисования овалов.
    - Кнопка **«Rectangle»** для создания прямоугольников.
    - Кнопка **«Pencil»** для свободного рисования.
* Кнопка **Color** – открывает меню выбора цвета для рисования;
* Кнопка **Background Color -** открывает меню выбора цвета для фона;
* Кнопка Undo – Отменяет последнее действие согласно истории действий.

**Технические особенности:**

* Каждый графический элемент (точка, линия, круг, квадрат, треугольник) реализован в виде отдельного класса (Point, Line, Circle, Square, Triangle), что обеспечивает модульность и упрощает расширение программы;
* Для рисования используется объектный подход: каждый инструмент рисования (например, точка или линия) создается и обрабатывается методами соответствующего класса;
* История действий (список всех созданных элементов) сохраняется в памяти, что позволяет пользователю шаг за шагом отменять выполненные действия.

**Ограничения:**

* Программа предназначена для базовых операций и не поддерживает сохранение или экспорт нарисованного в виде файла;
* Функционал рассчитан на интуитивное использование и не включает сложные инструменты редактирования.

1. **Теоретические сведения**
   1. **Принцип управления программой**

**Запуск программы:**

1. **Исполняемый файл**

Для запуска программы необходимо найти файл с именем draww**.exe** и дважды кликнуть по этому файлу, чтобы открыть программу.

1. **Окно программы**

После запуска программы откроется главное окно, содержащее:

* + Рабочую область (холст), на которой будут рисоваться элементы;
  + Панель инструментов, позволяющую выбирать цвет, цвет фона(холста) и тип рисуемых элементов.

**Работа с графическим интерфейсом**

1. **Инструмент «Line»**
   * Выберите инструмент «Line» в меню «Mode»;
   * Зажмите левую кнопку мыши в месте начала линии, а затем отпустите ЛКМ в месте её окончания. Программа соединит эти точки прямой линией.
2. **Инструмент «Oval»**
   * Выберите инструмент «Oval» в меню «Mode»;
   * Укажите начальную точку круга зажатием ЛКМ, а затем конечную точку отпусканием ЛКМ;
   * После этого на холсте появится круг заданного размера, цвета и толщины.
3. **Инструмент «Rectangle»**
   * Выберите инструмент «Rectangle» в меню «Mode»;
   * Нажмите ЛКМ в месте, где должен находиться один из углов квадрата, а затем потяните курсор и укажите противоположный угол путём отпускания ЛКМ.
4. **Инструмент «Pencil»**
   * Выберите кнопку «Pencil» в меню «Mode»;
   * Зажмите левую кнопку мыши на холсте и начните двигать курсор. Произвольная линия будет рисоваться вдоль движения вашего курсора;
5. **Изменение цвета**
   * Нажмите кнопку «Choose color» на панели инструментов;
   * В появившемся диалоговом окне выберите желаемый цвет и подтвердите выбор;
   * После этого все рисуемые элементы будут создаваться в выбранном цвете.
6. **Изменение цвета фона**
   * Нажмите кнопку «Choose background color» на панели инструментов;
   * В появившемся диалоговом окне выберите желаемый цвет и подтвердите выбор;
   * После этого все рисуемые элементы будут создаваться в выбранном цвете.
7. **Функция «Undo»**
   * Если вы хотите отменить последнее действие, нажмите кнопку «Undo»;
   * Последнее нарисованное действие будет удалено;
   * Вы можете использовать кнопку «Undo» многократно для последовательного удаления предыдущих действий.

#### Завершение работы

После завершения работы просто закройте окно программы. Программа не сохраняет результаты работы.

1. **Организация данных**

Структура программы представляет собой графический редактор, который использует библиотеку tkinter для создания пользовательского интерфейса. В редакторе можно рисовать различные геометрические фигуры (прямоугольники, овалы, линии), рисовать карандашом (свободное рисование), изменять цвет фигур и фона, а также отменять последнее действие.

Программа организована в виде классов с чётко выделенными ролями для каждого компонента. Давайте рассмотрим структуру более подробно.

**Основные компоненты программы:**

1. **Классы для фигур (Shape, Rectangle, Oval, Line)**:

* Это абстрактные и конкретные классы, которые отвечают за создание и рисование различных фигур.
* Shape (ABC) — это абстрактный базовый класс для всех фигур. Он определяет интерфейс для рисования (draw), а также хранит общие атрибуты для всех фигур (например, координаты и цвет).
* Rectangle, Oval, Line — это классы, которые наследуют Shape и реализуют метод draw, создавая конкретные геометрические фигуры (прямоугольники, овалы, линии) на холсте с помощью соответствующих методов create\_rectangle, create\_oval, create\_line из tkinter.

1. **Класс Pencil**:

* Pencil — это класс, который управляет рисованием с помощью карандаша (свободное рисование). Он отслеживает движение мыши и рисует линии между предыдущей и текущей позицией.
* Для каждой линии, нарисованной карандашом, сохраняется идентификатор объекта, который позволяет впоследствии удалить эти линии (например, при отмене действия).

1. **Класс GraphicEditor**:

* Это основной класс, который управляет логикой работы редактора.
* В нем создается графический интерфейс, включая холст для рисования и панель инструментов с различными опциями (выбор фигуры, цвет и фона, режим рисования).
  + **Атрибуты**:
  + canvas: холст для рисования.
  + shape: строковая переменная, которая определяет текущий выбранный тип фигуры (прямоугольник, овал, линия, карандаш).
  + color: цвет, выбранный для рисуемых объектов.
  + bg\_color: цвет фона канвы.
  + start\_x, start\_y: начальные координаты для рисования фигуры.
  + pencil: объект карандаша для рисования.
  + drawing\_mode: текущий режим рисования (фигура или карандаш).
  + objects: стек для хранения всех нарисованных объектов (для реализации функции отмены).
  + **Методы**:
  + create\_toolbox: создаёт панель инструментов для выбора формы, цвета и других опций.
  + switch\_mode: переключает режим рисования (фигуры или карандаш).
  + choose\_color: открывает диалоговое окно для выбора цвета.
  + choose\_bg\_color: открывает диалоговое окно для выбора цвета фона.
  + bind\_events: связывает события с обработчиками (например, щелчок мышью для рисования).
  + on\_mouse\_down: обрабатывает событие нажатия кнопки мыши (запоминает начальные координаты).
  + on\_mouse\_up: обрабатывает событие отпускания кнопки мыши (рисует фигуру или завершает рисование карандашом).
  + on\_mouse\_motion: обрабатывает событие перемещения мыши, рисуя линии в режиме карандаша.
  + undo: отменяет последнее действие, удаляя последний нарисованный объект или линии карандаша.
  + Структура классов:
  + Shape (ABC)
  + Абстрактный базовый класс для всех фигур.
  + Метод draw: абстрактный метод, который должны реализовать все дочерние классы.
  + Метод remove: удаляет объект с холста.
  + Rectangle, Oval, Line (наследуют Shape)
  + Каждый из этих классов реализует метод draw, рисуя соответствующую фигуру на холсте.
  + Pencil
  + Рисует линии на холсте, отслеживая движения мыши.
  + Содержит список всех нарисованных линий, что позволяет удалять их при отмене действия.
  + GraphicEditor
  + Это главный класс программы, который управляет логикой интерфейса и рисования.
  + Он обрабатывает события мыши, переключает режимы рисования и отменяет последние действия.
  + Принцип работы:
  + Интерфейс:
  + Когда программа запускается, создаётся окно с холстом для рисования и панелью инструментов, на которой можно выбрать тип фигуры (прямоугольник, овал, линия, карандаш), а также цвет объектов и фона.
  + Рисование:
  + В зависимости от выбранного режима рисования (форма или карандаш) при клике и движении мыши на холсте:
  + В режиме "Shape" рисуются фигуры (прямоугольники, овалы, линии).
  + В режиме "Pencil" создаются линии, следуя за движением мыши.

1. **Отмена последнего действия**:

* При нажатии кнопки "Undo" программа удаляет последний нарисованный объект (или линии, нарисованные карандашом). Для этого используется список objects, который хранит все нарисованные объекты, а также метод remove, который удаляет объект с холста.

1. **Текст программы**

import tkinter as tk  
from tkinter import colorchooser  
from abc import ABC, abstractmethod  
  
# Базовый класс для фигуры  
class Shape(ABC):  
 def \_\_init\_\_(self, canvas, x1, y1, x2, y2, \*\*kwargs):  
 self.canvas = canvas  
 self.x1, self.y1 = x1, y1  
 self.x2, self.y2 = x2, y2  
 self.options = kwargs  
 self.id = None # Идентификатор объекта на холсте  
  
 @abstractmethod  
 def draw(self):  
 pass  
  
 def remove(self):  
 *"""Удаляет фигуру с холста."""* if self.id:  
 self.canvas.delete(self.id)  
  
# Класс для прямоугольника  
class Rectangle(Shape):  
 def draw(self):  
 self.id = self.canvas.create\_rectangle(self.x1, self.y1, self.x2, self.y2, \*\*self.options)  
  
# Класс для овала  
class Oval(Shape):  
 def draw(self):  
 self.id = self.canvas.create\_oval(self.x1, self.y1, self.x2, self.y2, \*\*self.options)  
  
# Класс для линии  
class Line(Shape):  
 def draw(self):  
 self.id = self.canvas.create\_line(self.x1, self.y1, self.x2, self.y2, \*\*self.options)  
  
# Класс для рисования карандашом  
class Pencil:  
 def \_\_init\_\_(self, canvas, color):  
 self.canvas = canvas  
 self.color = color  
 self.previous\_x = None  
 self.previous\_y = None  
 self.lines = [] # Список всех линий, нарисованных карандашом  
  
 def draw(self, x, y):  
 if self.previous\_x and self.previous\_y:  
 line\_id = self.canvas.create\_line(self.previous\_x, self.previous\_y, x, y, fill=self.color)  
 self.lines.append(line\_id) # Сохраняем идентификатор линии  
 self.previous\_x = x  
 self.previous\_y = y  
  
 def reset(self):  
 self.previous\_x = None  
 self.previous\_y = None  
  
 def remove(self):  
 *"""Удаляет все линии, нарисованные карандашом."""* for line in self.lines:  
 self.canvas.delete(line)  
 self.lines.clear()  
  
# Класс для главного приложения  
class GraphicEditor:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Graphic Editor")  
  
 self.canvas = tk.Canvas(root, width=800, height=600, bg="white")  
 self.canvas.pack()  
  
 self.shape = tk.StringVar(value="Rectangle")  
 self.color = "blue"  
 self.bg\_color = "white"  
 self.start\_x = self.start\_y = None  
 self.pencil = None # Инициализируем карандаш  
 self.drawing\_mode = "Shape" # Режим рисования: по умолчанию "Shape"  
 self.objects = [] # Список всех нарисованных объектов для undo  
  
 self.create\_toolbox()  
 self.bind\_events()  
  
 def create\_toolbox(self):  
 toolbox = tk.Frame(self.root)  
 toolbox.pack(side=tk.TOP, fill=tk.X)  
  
 # Переключатель между режимами  
 mode\_label = tk.Label(toolbox, text="Mode:")  
 mode\_label.pack(side=tk.LEFT)  
  
 mode\_menu = tk.OptionMenu(toolbox, self.shape, "Rectangle", "Oval", "Line", "Pencil", command=self.switch\_mode)  
 mode\_menu.pack(side=tk.LEFT)  
  
 tk.Label(toolbox, text="Color:").pack(side=tk.LEFT)  
 color\_button = tk.Button(toolbox, text="Choose Color", command=self.choose\_color)  
 color\_button.pack(side=tk.LEFT)  
  
 # Добавляем кнопку для изменения цвета фона  
 tk.Label(toolbox, text="Background Color:").pack(side=tk.LEFT)  
 bg\_color\_button = tk.Button(toolbox, text="Choose Background Color", command=self.choose\_bg\_color)  
 bg\_color\_button.pack(side=tk.LEFT)  
  
 # Добавляем кнопку для отмены последнего действия  
 undo\_button = tk.Button(toolbox, text="Undo", command=self.undo)  
 undo\_button.pack(side=tk.LEFT)  
  
 def switch\_mode(self, selected\_mode):  
 *"""Переключает режим рисования (фигуры или карандаш)."""* if selected\_mode == "Pencil":  
 self.drawing\_mode = "Pencil"  
 else:  
 self.drawing\_mode = "Shape"  
  
 def choose\_color(self):  
 color\_code = colorchooser.askcolor(title="Choose Color")[1]  
 if color\_code:  
 self.color = color\_code  
  
 def choose\_bg\_color(self):  
 color\_code = colorchooser.askcolor(title="Choose Background Color")[1]  
 if color\_code:  
 self.bg\_color = color\_code  
 self.canvas.config(bg=self.bg\_color)  
  
 def bind\_events(self):  
 self.canvas.bind("<Button-1>", self.on\_mouse\_down)  
 self.canvas.bind("<ButtonRelease-1>", self.on\_mouse\_up)  
 self.canvas.bind("<B1-Motion>", self.on\_mouse\_motion)  
  
 def on\_mouse\_down(self, event):  
 self.start\_x, self.start\_y = event.x, event.y  
  
 # Если выбран режим карандаша, создаем объект карандаша  
 if self.drawing\_mode == "Pencil":  
 self.pencil = Pencil(self.canvas, self.color)  
  
 def on\_mouse\_up(self, event):  
 if self.drawing\_mode == "Shape":  
 end\_x, end\_y = event.x, event.y  
 shape\_class = {  
 "Rectangle": Rectangle,  
 "Oval": Oval,  
 "Line": Line,  
 }[self.shape.get()]  
  
 shape = shape\_class(self.canvas, self.start\_x, self.start\_y, end\_x, end\_y, fill=self.color)  
 shape.draw()  
 self.objects.append(shape) # Добавляем объект в список для отмены  
  
 # Если это был режим карандаша, сбрасываем карандаш  
 if self.pencil:  
 self.pencil.reset()  
  
 def on\_mouse\_motion(self, event):  
 if self.drawing\_mode == "Pencil" and self.pencil:  
 self.pencil.draw(event.x, event.y)  
  
 def undo(self):  
 *"""Отменяет последнее действие."""* if self.objects:  
 last\_object = self.objects.pop() # Извлекаем последний объект из списка  
 last\_object.remove() # Удаляем его с холста  
  
 # Если карандаш рисовал линии, удаляем их  
 if self.pencil:  
 self.pencil.remove()  
 self.pencil = None  
  
# Запуск приложения  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = GraphicEditor(root)  
 root.mainloop()