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 Мацуев К.И.

(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](#_Toc184225809)

[2. Спецификация 4](#_Toc184225810)

[3. Теоретические сведения 7](#_Toc184225811)

[4. Организация данных 10](#_Toc184225812)

[5. Текст программы 14](#_Toc184225813)

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. **Рисование точек:**
   * Пользователь может выбрать инструмент «Точка» и размещать точки произвольной толщины и цвета на холсте;
   * Точки создаются в месте нажатия мыши на холсте;
   * Толщина точки и цвет задаются через панель инструментов.
2. **Рисование линий:**
   * 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;
   * Толщина линии и цвет задаются через панель инструментов.
3. **Рисование кругов:**
   * Пользователь может создавать круги произвольного размера, задавая начальную и конечную точку;
   * Толщина линии круга и цвет задаются через панель инструментов.
4. **Рисование квадратов:**
   * Квадраты создаются путем задания начальной и конечной точки, которая определяет размеры фигуры;
   * Цвет и толщина линии квадрата настраиваются через панель инструментов.
5. **Рисование треугольников:**
   * Треугольники создаются путем задания начальной и конечной точки, которая определяет размеры фигуры;
   * Фигура автоматически выравнивается для создания равнобедренного треугольника;
   * Толщина линий и цвет настраиваются в панели инструментов.
6. **Свободное рисование:**
   * Данный функционал позволяет пользователю рисовать произвольные линии, перемещая мышь с зажатой левой кнопкой мыши;
   * Цвет и толщина линии задаются в панели инструментов.
7. **Выбор цвета:**

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

1. **Задание толщины линий:**
   * Пользователь может установить толщину линий, которая будет применяться ко всем инструментам;
   * По умолчанию начальная толщина установлена на 3.
2. **Функция отмены действия:**

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

**Основные интерфейсные элементы:**

* **Холст** –рабочая область, где пользователь может рисовать;
* **Панель инструментов:**
  1. Кнопка **«Точка»** для размещения точек.
  2. Кнопка **«Линия»** для построения прямых.
  3. Кнопка **«Круг»** для рисования кругов.
  4. Кнопка **«Квадрат»** для создания квадратов.
  5. Кнопка **«Треугольник»** для построения треугольников.
  6. Кнопка **«Рисование»** для свободного рисования.
  7. Кнопка **«Цвет»** для выбора цвета элементов.
  8. Поле ввода толщины с начальным значением 3 для задания толщины линий.
  9. Кнопка **«Отменить»** для удаления последнего действия.

**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 **graphic\_editor.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, толщину линий и тип рисуемых элементов.

**Работа с графическим интерфейсом**

1. **Инструмент «Точка»**
   * Выберите инструмент «Точка», нажав на соответствующую кнопку на панели инструментов;
   * Кликните левой кнопкой мыши в любом месте холста, чтобы создать точку заданного цвета и толщины;
   * Чтобы нарисовать следующую точку, просто повторите клик в другом месте холста.
2. **Инструмент «Линия»**
   * Выберите инструмент «Линия»;
   * Зажмите левую кнопку мыши в месте начала линии, а затем отпустите ЛКМ в месте её окончания. Программа соединит эти точки прямой линией.
3. **Инструмент «Круг»**
   * Выберите инструмент «Круг»;
   * Укажите начальную точку круга зажатием ЛКМ, а затем конечную точку отпусканием ЛКМ;
   * После этого на холсте появится круг заданного размера, цвета и толщины.
4. **Инструмент «Квадрат»**
   * Выберите инструмент «Квадрат» ;
   * Нажмите ЛКМ в месте, где должен находиться один из углов квадрата, а затем потяните курсор и укажите противоположный угол путём отпускания ЛКМ.
5. **Инструмент «Треугольник»**
   * Выберите инструмент «Треугольник» ;
   * Укажите первую точку зажатием ЛКМ, а затем вторую точку отпусканием ЛКМ на холсте, которая задаст основание и высоту треугольника;
   * Треугольник автоматически выравнивается как равнобедренный.
6. **Инструмент «Рисование»**
   * Выберите кнопку «Рисование» ;
   * Зажмите левую кнопку мыши на холсте и начните двигать курсор. Произвольная линия будет рисоваться вдоль движения вашего курсора;
   * Толщина и цвет линии задаются параметрами, указанными в панели инструментов.
7. **Изменение цвета**
   * Нажмите кнопку «Цвет» на панели инструментов;
   * В появившемся диалоговом окне выберите желаемый цвет и подтвердите выбор;
   * После этого все рисуемые элементы будут создаваться в выбранном цвете.
8. **Изменение толщины линии**
   * В панели инструментов находится поле «Толщина»;
   * Введите желаемое значение толщины линии (по умолчания: 3) ;
   * Указанная толщина будет применяться ко всем последующим элементам.
9. **Функция «Отменить»**
   * Если вы допустили ошибку, нажмите кнопку «Отменить»;
   * Последнее нарисованное действие будет удалено;
   * Вы можете использовать кнопку «Отменить» многократно для последовательного удаления предыдущих действий.

#### Завершение работы

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

1. **Организация данных**

Программа построена на принципах объектно-ориентированного программирования (ООП), что делает её структуру гибкой и легко расширяемой. Код состоит из классов и методов, которые определяют логику работы графического редактора, его интерфейс и функционал.

**Структура программы**

Программа разделена на два основных компонента:

1. **Классы рисуемых объектов**

Представляют графические элементы (точка, линия, круг, квадрат, треугольник). Каждый объект содержит свойства (цвет, толщину, координаты) и логику для создания заданной фигуры.

1. **Класс графического редактора**

* Управляет взаимодействием пользователя с интерфейсом;
* Реализует обработку событий (нажатие кнопок, рисование на холсте);
* Хранит историю действий для их отмены.

**Описание классов и методов**

1. **Абстрактный класс «Drawable»**

Это базовый класс для всех рисуемых объектов.

**Атрибуты:**

* + - canvas – ссылка на холст, где происходит отрисовка;
    - color – цвет объекта;
    - thickness – толщина линии или размера объекта.

**Методы:**

* draw()– абстрактный метод, который должен быть реализован в каждом подклассе для отрисовки конкретного элемента.

1. **Подклассы «Drawable»**

**Point** – класс для рисования точки:

* + - Сохраняет координаты x и y;
    - Реализует метод «draw()», который создаёт точку на холсте с помощью метода «canvas.create\_oval».

**Line** – класс для рисования линии:

* Сохраняет координаты начальной и конечной точек (x1, y1, x2, y2) ;
* Метод «draw()»» создаёт линию на холсте с помощью метода «canvas.create\_line».

**Circle**, **Square**, **Triangle** – классы для рисования круга, квадрата и треугольника, соответственно.

* + - Все эти классы используют координаты начальной и конечной точек для определения размеров фигуры;
    - Каждый класс реализует метод draw() для соответствующего элемента (create\_oval, create\_rectangle, create\_polygon).

1. **Класс «»DrawingEditor»**

Это основной класс программы, который управляет взаимодействием между пользователем, графическим интерфейсом и рисуемыми объектами.

**Атрибуты:**

* + - canvas – рабочая область для рисования;
    - current\_color – текущий цвет, выбранный пользователем;
    - current\_thickness – текущая толщина линии;
    - history – последовательный список, в котором хранятся все нарисованные объекты для возможности их отмены;
    - drawing\_mode – текущий режим рисования;
    - temp\_shape – временный объект, используемый для отрисовки интерактивных фигур.

**Методы:**

* + - create\_toolbar() – создаёт панель инструментов;
    - bind\_canvas\_events() – связывает события мыши с обработчиками (нажатие, перемещение, отпускание кнопки) ;
    - choose\_color() – вызывает диалоговое окно для выбора цвета;
    - get\_thickness() – возвращает текущую толщину линии (считанную из поля ввода).

**Режимы рисования:**

Например: set\_point\_mode(), set\_line\_mode(), set\_circle\_mode(), set\_square\_mode(), set\_triangle\_mode(), set\_draw\_mode() – устанавливают текущий режим рисования.

**Обработка событий:**

* on\_click(event) – вызывается при нажатии кнопки мыши. Реализует начальную логику для рисования объекта или начала произвольной линии;
* on\_drag(event) – вызывается при перемещении мыши с зажатой кнопкой. Используется для интерактивного рисования (произвольного или объектов с изменяемыми размерами);
* on\_release(event) – вызывается при отпускании кнопки мыши. Завершает рисование объекта;
* undo() – удаляет последний нарисованный элемент, используя сохранённые из истории.

**Логика работы программы**

1. **Запуск и интерфейс:**

При запуске программы создаётся главное окно с холстом и панелью инструментов. Панель инструментов позволяет задавать цвет, толщину и выбирать тип рисуемого элемента.

1. **Выбор инструмента:**

Пользователь выбирает инструмент (точка, линия, фигура, рисование) с панели инструментов. Программа переключается в соответствующий режим с помощью методов set\_\*\_mode().

1. **Рисование:**

При нажатии на холст (on\_click) создаётся объект выбранного класса (Point, Line, Circle и т.д.), и вызывается метод draw() для его отрисовки. В случае сложных фигур (линия, круг, квадрат) временный объект создаётся в «on\_click», обновляется в «on\_drag и фиксируется в «on\_release».

1. **Отмена действий:**

Все нарисованные элементы сохраняются в списке history по мере их создания. Кнопка «Отменить» удаляет последний объект из холста и списка history, что позволяет последовательно отменять действия пользователя.

1. **Цвет и толщина:**

Цвет задаётся через диалоговое окно, вызываемое кнопкой «Цвет».

Толщина линий и фигур считывается из текстового поля.

1. **Текст программы**

import tkinter as tk  
from tkinter.colorchooser import askcolor  
  
  
class Drawable:  
 *"""Абстрактный класс для рисуемых объектов"""* def \_\_init\_\_(self, canvas, color, thickness):  
 self.canvas = canvas  
 self.color = color  
 self.thickness = thickness  
  
 def draw(self):  
 raise NotImplementedError("Метод 'draw' должен быть реализован в подклассах.")  
  
  
class Point(Drawable):  
 def \_\_init\_\_(self, canvas, x, y, color, thickness):  
 super().\_\_init\_\_(canvas, color, thickness)  
 self.x = x  
 self.y = y  
 self.draw()  
  
 def draw(self):  
 self.item = self.canvas.create\_oval(  
 self.x - self.thickness // 2,  
 self.y - self.thickness // 2,  
 self.x + self.thickness // 2,  
 self.y + self.thickness // 2,  
 fill=self.color,  
 outline=self.color  
 )  
  
  
class Line(Drawable):  
 def \_\_init\_\_(self, canvas, x1, y1, x2, y2, color, thickness):  
 super().\_\_init\_\_(canvas, color, thickness)  
 self.x1 = x1  
 self.y1 = y1  
 self.x2 = x2  
 self.y2 = y2  
 self.draw()  
  
 def draw(self):  
 self.item = self.canvas.create\_line(  
 self.x1, self.y1, self.x2, self.y2,  
 fill=self.color, width=self.thickness  
 )  
  
  
class Circle(Drawable):  
 def \_\_init\_\_(self, canvas, x1, y1, x2, y2, color, thickness):  
 super().\_\_init\_\_(canvas, color, thickness)  
 self.x1 = x1  
 self.y1 = y1  
 self.x2 = x2  
 self.y2 = y2  
 self.draw()  
  
 def draw(self):  
 self.item = self.canvas.create\_oval(  
 self.x1, self.y1, self.x2, self.y2,  
 outline=self.color, width=self.thickness  
 )  
  
  
class Square(Drawable):  
 def \_\_init\_\_(self, canvas, x1, y1, x2, y2, color, thickness):  
 super().\_\_init\_\_(canvas, color, thickness)  
 self.x1 = x1  
 self.y1 = y1  
 self.x2 = x2  
 self.y2 = y2  
 self.draw()  
  
 def draw(self):  
 self.item = self.canvas.create\_rectangle(  
 self.x1, self.y1, self.x2, self.y2,  
 outline=self.color, width=self.thickness  
 )  
  
  
class Triangle(Drawable):  
 def \_\_init\_\_(self, canvas, x1, y1, x2, y2, color, thickness):  
 super().\_\_init\_\_(canvas, color, thickness)  
 self.x1 = x1  
 self.y1 = y1  
 self.x2 = x2  
 self.y2 = y2  
 self.draw()  
  
 def draw(self):  
 half\_side = abs(self.x2 - self.x1) / 2  
 height = abs(self.y2 - self.y1)  
 self.item = self.canvas.create\_polygon(  
 self.x1, self.y1,  
 self.x1 - half\_side, self.y1 + height,  
 self.x1 + half\_side, self.y1 + height,  
 outline=self.color, fill="", width=self.thickness  
 )  
  
  
class DrawingEditor:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Графический редактор")  
  
 self.current\_color = "#000000"  
 self.current\_thickness = 3 # Изменено начальное значение толщины на 3  
 self.history = []  
  
 self.canvas = tk.Canvas(root, bg="white", width=600, height=400)  
 self.canvas.pack()  
  
 self.create\_toolbar()  
 self.bind\_canvas\_events()  
 self.drawing\_mode = None  
 self.temp\_shape = None  
  
 def create\_toolbar(self):  
 toolbar = tk.Frame(self.root)  
 toolbar.pack()  
  
 tk.Button(toolbar, text="Цвет", command=self.choose\_color).pack(side=tk.LEFT)  
 tk.Label(toolbar, text="Толщина:").pack(side=tk.LEFT)  
 self.thickness\_entry = tk.Entry(toolbar, width=3)  
 self.thickness\_entry.insert(0, str(self.current\_thickness))  
 self.thickness\_entry.pack(side=tk.LEFT)  
  
 tk.Button(toolbar, text="Точка", command=self.set\_point\_mode).pack(side=tk.LEFT)  
 tk.Button(toolbar, text="Линия", command=self.set\_line\_mode).pack(side=tk.LEFT)  
 tk.Button(toolbar, text="Круг", command=self.set\_circle\_mode).pack(side=tk.LEFT)  
 tk.Button(toolbar, text="Квадрат", command=self.set\_square\_mode).pack(side=tk.LEFT)  
 tk.Button(toolbar, text="Треугольник", command=self.set\_triangle\_mode).pack(side=tk.LEFT)  
 tk.Button(toolbar, text="Рисование", command=self.set\_draw\_mode).pack(side=tk.LEFT)  
 tk.Button(toolbar, text="Отменить", command=self.undo).pack(side=tk.LEFT)  
  
 def bind\_canvas\_events(self):  
 self.canvas.bind("<Button-1>", self.on\_click)  
 self.canvas.bind("<B1-Motion>", self.on\_drag)  
 self.canvas.bind("<ButtonRelease-1>", self.on\_release)  
  
 def choose\_color(self):  
 color = askcolor(color=self.current\_color)[1]  
 if color:  
 self.current\_color = color  
  
 def get\_thickness(self):  
 try:  
 return int(self.thickness\_entry.get())  
 except ValueError:  
 return self.current\_thickness  
  
 def set\_point\_mode(self):  
 self.drawing\_mode = "point"  
  
 def set\_line\_mode(self):  
 self.drawing\_mode = "line"  
  
 def set\_circle\_mode(self):  
 self.drawing\_mode = "circle"  
  
 def set\_square\_mode(self):  
 self.drawing\_mode = "square"  
  
 def set\_triangle\_mode(self):  
 self.drawing\_mode = "triangle"  
  
 def set\_draw\_mode(self):  
 self.drawing\_mode = "draw"  
  
 def on\_click(self, event):  
 self.start\_x, self.start\_y = event.x, event.y  
 thickness = self.get\_thickness()  
  
 if self.drawing\_mode == "point":  
 point = Point(self.canvas, event.x, event.y, self.current\_color, thickness)  
 self.history.append(point.item)  
 elif self.drawing\_mode == "line":  
 self.temp\_shape = Line(self.canvas, event.x, event.y, event.x, event.y, self.current\_color, thickness)  
 elif self.drawing\_mode == "circle":  
 self.temp\_shape = Circle(self.canvas, event.x, event.y, event.x, event.y, self.current\_color, thickness)  
 elif self.drawing\_mode == "square":  
 self.temp\_shape = Square(self.canvas, event.x, event.y, event.x, event.y, self.current\_color, thickness)  
 elif self.drawing\_mode == "triangle":  
 self.temp\_shape = Triangle(self.canvas, event.x, event.y, event.x, event.y, self.current\_color, thickness)  
 elif self.drawing\_mode == "draw":  
 self.last\_x, self.last\_y = event.x, event.y  
  
 def on\_drag(self, event):  
 if self.drawing\_mode == "draw":  
 thickness = self.get\_thickness()  
 line = self.canvas.create\_line(  
 self.last\_x, self.last\_y, event.x, event.y,  
 fill=self.current\_color, width=thickness  
 )  
 self.history.append(line)  
 self.last\_x, self.last\_y = event.x, event.y  
 elif self.drawing\_mode in {"line", "circle", "square", "triangle"}:  
 if self.temp\_shape:  
 self.canvas.delete(self.temp\_shape.item)  
 if self.drawing\_mode == "line":  
 self.temp\_shape = Line(self.canvas, self.start\_x, self.start\_y, event.x, event.y, self.current\_color,  
 self.get\_thickness())  
 elif self.drawing\_mode == "circle":  
 self.temp\_shape = Circle(self.canvas, self.start\_x, self.start\_y, event.x, event.y, self.current\_color,  
 self.get\_thickness())  
 elif self.drawing\_mode == "square":  
 self.temp\_shape = Square(self.canvas, self.start\_x, self.start\_y, event.x, event.y, self.current\_color,  
 self.get\_thickness())  
 elif self.drawing\_mode == "triangle":  
 self.temp\_shape = Triangle(self.canvas, self.start\_x, self.start\_y, event.x, event.y,  
 self.current\_color, self.get\_thickness())  
  
 def on\_release(self, event):  
 if self.temp\_shape:  
 self.history.append(self.temp\_shape.item)  
 self.temp\_shape = None  
  
 def undo(self):  
 if self.history:  
 last\_item = self.history.pop()  
 self.canvas.delete(last\_item)  
  
  
root = tk.Tk()  
app = DrawingEditor(root)  
root.mainloop()