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 Лебедев А.А.

(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) а также произвольно рисовать с помощью мыши.

**Функциональные возможности:**

1. **Рисование точек:**
   * Пользователь может выбрать инструмент «Нарисовать точку» и размещать точки произвольно на холсте;
   * Точки создаются в месте нажатия мыши на холсте;
2. **Рисование линий:**
   * Линии создаются автоматически по уже нарисованным точкам;
   * При нажатии кнопки «Нарисовать линию» все точки автоматически соединяются в том порядке, в каком были поставлены, а также соединяя последнюю и первую точку (1 -> 2 -> … -> 1);
3. **Рисование кругов:**
   * Пользователь может создавать круги произвольного размера в центре экрана;
   * Радиус круга указывается в отдельной форме ввода.
4. **Рисование квадратов:**
   * Пользователь может создавать квадрата произвольного размера в центре экрана;
   * Размер стороны квадрата указывается в отдельной форме ввода.
5. **Свободное рисование:**
   * Данный функционал позволяет пользователю рисовать произвольные линии, перемещая мышь с зажатой левой кнопкой мыши;
   * Для активации режима свободного рисования требуется нажать кнопку «Свободное рисование».
6. **Функция очистки холста:**

* В случае, если пользователю требуется очистить холст, это делается по нажатию кнопки «Очистить окно»

**Основные интерфейсные элементы:**

* **Холст** –рабочая область, где пользователь может рисовать;
* **Панель инструментов:**
  1. Кнопка **«Нарисовать точку»** для размещения точек.
  2. Кнопка **«Нарисовать линию»** для построения прямых.
  3. Кнопка **«Построить круг»** для рисования кругов.
  4. Кнопка **«Построить квадрат»** для создания квадратов.
  5. Кнопка **«Очистить окно»** для удаления всех действий на холсте.

**Технические особенности:**

* Каждый графический элемент (точка, линия, круг, квадрат, треугольник) реализован в виде отдельного класса (Point, Line, Circle, Square), 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 подключить проект в IDE “PyCharm” и в нем нажать F10

1. **Окно программы**

После запуска программы откроется главное окно, содержащее:

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

**Работа с графическим интерфейсом**

1. **Инструмент «****Нарисовать точку»**
   * Выберите инструмент «Нарисовать точку», 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 «****Нарисовать линию»**
   * Выберите инструмент «Нарисовать линию»;
   * При выборе инструмента все нарисованные точки на холсте автоматически соединятся линиями в том порядке, в каком они были поставлены, плюс соединяется первая и последняя точки
3. **Инструмент «Построить круг»**
   * Выберите инструмент «Построить круг»;
   * Укажите радиус круга;
   * После этого в центре холста отрисуется круг указанного размера.
4. **Инструмент «Построить квадрат»**
   * Выберите инструмент «Построить квадрат» ;
   * Укажите длину стороны квадрата.
   * После этого в центре холста отрисуется квадрат указанного размера.
5. **Инструмент «Свободное рисование»**
   * Выберите кнопку «Свободное рисование» ;
   * Зажмите левую кнопку мыши на холсте и начните двигать курсор. Произвольная линия будет рисоваться вдоль движения вашего курсора;
6. **Функция «Очистить окно»**
   * Если рабочая область слишком переполнена, нажмите кнопку «Очистить окно»;
   * Окно будет полностью очищено;
   * Процесс необратим.

#### 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);

**Описание классов и методов**

1. **Абстрактный класс «Shape»**

Это базовый класс для всех рисуемых объектов.

**Атрибуты:**

* + - canvas – ссылка на холст, где происходит отрисовка;

**Методы:**

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

1. **Подклассы «Shape»**

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

**Circle**, **Square**– 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).

1. **Класс «»GrapgicalEditor»**

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

**Атрибуты:**

* + - canvas – рабочая область для рисования;
    - drawing\_mode – текущий режим рисования;
    - shapes[] – массив фигур на экране
    - points[] – массив точек на экране

**Методы:**

* + - \_init\_ () – создаёт панель инструментов;
    - bind\_canvas\_events() – связывает события мыши с обработчиками (нажатие, перемещение, отпускание кнопки) ;
    - clear\_canvas() – очистка экрана

**Режимы рисования:**

Например: enable\_point\_drawing(), enable\_free\_drawing(), add\_point(), connect\_points(), free\_draw(), draw\_circle(), draw\_square() – 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;

**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 методо.

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 списках shapes и points по мере их создания. Кнопка «Очистить окно» удаляет все объекты с холста и списков.

1. **Текст программы**
2. import tkinter as tk  
   from tkinter import Canvas, simpledialog  
     
   class Shape:  
    *"""Базовый класс для всех фигур."""* def draw(self, canvas):  
    raise NotImplementedError("Метод draw() должен быть реализован в подклассе.")  
     
   class Point(Shape):  
    def \_\_init\_\_(self, x, y):  
    self.x = x  
    self.y = y  
     
    def draw(self, canvas):  
    canvas.create\_oval(self.x - 2, self.y - 2, self.x + 2, self.y + 2, fill="black")  
     
   class Line(Shape):  
    def \_\_init\_\_(self, x1, y1, x2, y2):  
    self.x1 = x1  
    self.y1 = y1  
    self.x2 = x2  
    self.y2 = y2  
     
    def draw(self, canvas):  
    canvas.create\_line(self.x1, self.y1, self.x2, self.y2, fill="black")  
     
   class Circle(Shape):  
    def \_\_init\_\_(self, x, y, radius):  
    self.x = x  
    self.y = y  
    self.radius = radius  
     
    def draw(self, canvas):  
    canvas.create\_oval(self.x - self.radius, self.y - self.radius,  
    self.x + self.radius, self.y + self.radius, outline="black")  
     
   class Square(Shape):  
    def \_\_init\_\_(self, x, y, side):  
    self.x = x  
    self.y = y  
    self.side = side  
     
    def draw(self, canvas):  
    half\_side = self.side / 2  
    canvas.create\_rectangle(self.x - half\_side, self.y - half\_side,  
    self.x + half\_side, self.y + half\_side, outline="black")  
     
   class GraphicalEditor:  
    def \_\_init\_\_(self, root):  
    self.root = root  
    self.root.title("Графический редактор")  
     
    self.canvas = Canvas(self.root, width=800, height=600, bg="white")  
    self.canvas.pack()  
     
    self.btn\_draw\_point = tk.Button(self.root, text="Нарисовать точку", command=self.enable\_point\_drawing)  
    self.btn\_draw\_point.pack(side=tk.LEFT)  
     
    self.btn\_connect\_points = tk.Button(self.root, text="Соединить линиями", command=self.connect\_points)  
    self.btn\_connect\_points.pack(side=tk.LEFT)  
     
    self.btn\_draw\_circle = tk.Button(self.root, text="Построить круг", command=self.draw\_circle)  
    self.btn\_draw\_circle.pack(side=tk.LEFT)  
     
    self.btn\_draw\_square = tk.Button(self.root, text="Построить квадрат", command=self.draw\_square)  
    self.btn\_draw\_square.pack(side=tk.LEFT)  
     
    self.btn\_free\_draw = tk.Button(self.root, text="Свободное рисование", command=self.enable\_free\_drawing)  
    self.btn\_free\_draw.pack(side=tk.LEFT)  
     
    self.btn\_clear = tk.Button(self.root, text="Очистить окно", command=self.clear\_canvas)  
    self.btn\_clear.pack(side=tk.LEFT)  
     
    self.shapes = []  
    self.points = []  
     
    self.drawing\_points = False  
    self.free\_drawing = False  
     
    self.canvas.bind("<Button-1>", self.add\_point)  
    self.canvas.bind("<B1-Motion>", self.free\_draw)  
     
    def enable\_point\_drawing(self):  
    self.drawing\_points = True  
    self.free\_drawing = False  
     
    def enable\_free\_drawing(self):  
    self.free\_drawing = True  
    self.drawing\_points = False  
     
    def add\_point(self, event):  
    if self.drawing\_points:  
    point = Point(event.x, event.y)  
    point.draw(self.canvas)  
    self.points.append(point)  
     
    def connect\_points(self):  
    if len(self.points) < 2:  
    return  
    for i in range(len(self.points) - 1):  
    line = Line(self.points[i].x, self.points[i].y,  
    self.points[i + 1].x, self.points[i + 1].y)  
    line.draw(self.canvas)  
    self.shapes.append(line)  
    # Соединяем последнюю точку с первой  
    line = Line(self.points[-1].x, self.points[-1].y,  
    self.points[0].x, self.points[0].y)  
    line.draw(self.canvas)  
    self.shapes.append(line)  
    self.drawing\_points = False  
     
    def free\_draw(self, event):  
    if self.free\_drawing:  
    x, y = event.x, event.y  
    point = Point(x, y)  
    if self.shapes and isinstance(self.shapes[-1], Line):  
    last\_line = self.shapes[-1]  
    line = Line(last\_line.x2, last\_line.y2, x, y)  
    else:  
    line = Line(x, y, x, y)  
    line.draw(self.canvas)  
    self.shapes.append(line)  
     
    def draw\_circle(self):  
    radius = simpledialog.askinteger("Параметры круга", "Введите радиус:")  
    if radius is not None:  
    circle = Circle(400, 300, radius)  
    circle.draw(self.canvas)  
    self.shapes.append(circle)  
     
    def draw\_square(self):  
    side = simpledialog.askinteger("Параметры квадрата", "Введите длину стороны:")  
    if side is not None:  
    square = Square(400, 300, side)  
    square.draw(self.canvas)  
    self.shapes.append(square)  
     
    def clear\_canvas(self):  
    self.canvas.delete("all")  
    self.shapes.clear()  
    self.points.clear()  
     
   if \_\_name\_\_ == "\_\_main\_\_":  
    root = tk.Tk()  
    app = GraphicalEditor(root)  
    root.mainloop()