Московский Государственный Технический
Университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчёт по лабораторной работе №5
«Объектно-ориентированные возможности языка Python (исполнение на Rust)»

Выполнил:
студент группы ИУ5-25б
Бикматов Д. А.

Проверил:
преподаватель
Гапанюк Ю. В.

Москва, 2024 г.

Постановка задачи:

Выполнить следующее задание на ЯП Rust.

1. Необходимо создать виртуальное окружение и установить в него хотя бы один внешний пакет с использованием pip.
2. Необходимо разработать программу, реализующую работу с классами. Программа должна быть разработана в виде консольного приложения на языке Python 3.
3. Все файлы проекта (кроме основного файла main.py) должны располагаться в пакете lab_python_oop.
4. Каждый из нижеперечисленных классов должен располагаться в отдельном файле пакета lab_python_oop.
5. Абстрактный класс «Геометрическая фигура» содержит абстрактный метод для вычисления площади фигуры.
6. Класс «Цвет фигуры» содержит свойство для описания цвета геометрической фигуры.
7. Класс «Прямоугольник» наследуется от класса «Геометрическая фигура». Класс должен содержать конструктор по параметрам «ширина», «высота» и «цвет». В конструкторе создается объект класса «Цвет фигуры» для хранения цвета. Класс должен переопределять метод, вычисляющий площадь фигуры.
8. Класс «Круг» создается аналогично классу «Прямоугольник», задается параметр «радиус». Для вычисления площади используется константа math.pi из модуля math.
9. Класс «Квадрат» наследуется от класса «Прямоугольник». Класс должен содержать конструктор по длине стороны. Для классов «Прямоугольник», «Квадрат», «Круг»:
    - Определите метод "repr", который возвращает в виде строки основные параметры фигуры, ее цвет и площадь. Используйте метод format
    - Название фигуры («Прямоугольник», «Квадрат», «Круг») должно задаваться в виде поля данных класса и возвращаться методом класса.
10. В корневом каталоге проекта создайте файл main.py для тестирования Ваших классов. Создайте следующие объекты и выведите о них информацию в консоль (N - номер Вашего варианта по списку группы):
    - Прямоугольник синего цвета шириной N и высотой N.
    - Круг зеленого цвета радиусом N.
    - Квадрат красного цвета со стороной N.
    - Также вызовите один из методов внешнего пакета, установленного с использованием pip.

Текст программы:

figure.rs

```
pub trait Figure{
    fn repr(&self) -> String;
    fn get_area(&self) -> f64;
}
```

color.rs

```
pub struct Color{
    color: String
}

impl Color{
    pub fn new(str: String) -> Color{
        Color {color: str}
    }
    pub fn set_color(&mut self, str: String){
        self.color = str;
    }
    pub fn get_color(&self) -> String{
        self.color.clone()
    }
}
```

rect.rs

```
use crate::color::Color;
use std::fmt;
use crate::figure::Figure;

pub struct Rect{
    figure_type: String,
    pub width: f64,
    height: f64,
    pub color: Color
}

impl Rect{
    pub fn new(h: f64, w: f64, col: &Color) -> Rect{
        let c: Color = Color::new(col.get_color());
        Rect {figure_type: "rectangle".to_string(), width: w, height: h, color: c}
    }
    pub fn get_type(&self) -> String{
        self.figure_type.clone()
```

```
    }
}

impl Figure for Rect{
    fn get_area(&self) -> f64{
        self.width*self.height
    }
    fn repr(&self) -> String{
        format!("Тип: {}, Ширина: {}, Высота: {}, Цвет: {}, Площадь: {}",
self.get_type(), self.width, self.height, self.color.get_color(), self.get_area())
    }
}

impl std::fmt::Display for Rect{
    fn fmt(&self, f: &mut fmt::Formatter) -> fmt::Result {
        write!(f, "{}", self.repr())
    }
}
```

square.rs

```
use std::fmt;
use crate::figure::Figure;
use crate::color::Color;
use crate::rect::Rect;

pub struct Square{
    figure_type: String,
    rect: Rect
}

impl Square{
    pub fn new(s: f64, col: &Color) -> Square{
        let c: Color = Color::new(col.get_color());
        let r: Rect = Rect::new(s, s, &c);
        Square {figure_type: "square".to_string(), rect: r}
    }
    pub fn get_type(&self) -> String{
        self.figure_type.clone()
    }
}

impl Figure for Square{
    fn get_area(&self) -> f64{
        self.rect.get_area()
```

```
    }
    fn repr(&self) -> String{
        format!("Тип: {}, Сторона: {}, Цвет: {}, Площадь: {}", self.get_type(),
self.rect.width, self.rect.color.get_color(), self.get_area())
    }
}

impl std::fmt::Display for Square{
    fn fmt(&self, f: &mut fmt::Formatter) -> fmt::Result {
        write!(f, "{}", self.repr())
    }
}
```

circle.rs

```
use crate::color::Color;
use std::fmt;
use crate::figure::Figure;

pub struct Circle{
    figure_type: String,
    radius: f64,
    color: Color
}

impl Circle{
    pub fn new(rad: f64, col: &Color) -> Circle{
        let c: Color = Color::new(col.get_color());
        Circle {figure_type: "circle".to_string(), radius: rad, color: c}
    }
    pub fn get_type(&self) -> String{
        self.figure_type.clone()
    }
}

impl Figure for Circle{
    fn get_area(&self) -> f64{
        std::f64::consts::PI*self.radius.powi(2)
    }
    fn repr(&self) -> String{
        format!("Тип: {}, Радиус: {}, Цвет: {}, Площадь: {}", self.get_type(),
self.radius, self.color.get_color(), self.get_area())
    }
}
```

```
impl std::fmt::Display for Circle{
    fn fmt(&self, f: &mut fmt::Formatter) -> fmt::Result {
        write!(f, "{}", self.repr())
    }
}
```

lib.rs

```
pub mod rect;
pub mod square;
pub mod circle;
pub mod color;
pub mod figure;
```

cargo.toml

```
[package]
name = "figures"
version = "0.1.0"
edition = "2021"

[lib]
name = "figures"
path = "src/lib.rs"

[dependencies]
```

main.rs

```
use figures::color::Color;
use figures::square::Square;
use figures::rect::Rect;
use figures::circle::Circle;

fn main() {
    let mut col = Color::new("синий".to_string());
    let rect = Rect::new(3.0, 3.0, &col);
    col.set_color("зелёный".to_string());
    let circle = Circle::new(3.0, &col);
    col.set_color("красный".to_string());
    let square = Square::new(3.0, &col);
    println!("{}", rect);
    println!("{}", circle);
    println!("{}", square);
}
```

Пример выполнения:

Входные данные	Выходные данные
-