Разработка на языке программирования Rust.

Выполнил: Беликов Константин
Группа: ИУ5-36Б
Дата: 20.10.24г.

Описание задания:

Разработать программу, реализующую работу с классами.

1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Абстрактный класс «Геометрическая фигура» содержит виртуальный метод для вычисления площади фигуры.
3. Класс «Прямоугольник» наследуется от «Геометрическая фигура». Ширина и высота объявляются как свойства (property). Класс должен содержать конструктор по параметрам «ширина» и «высота».
4. Класс «Квадрат» наследуется от «Прямоугольник». Класс должен содержать конструктор по длине стороны.
5. Класс «Круг» наследуется от «Геометрическая фигура». Радиус объявляется как свойство (property). Класс должен содержать конструктор по параметру «радиус».
6. Для классов «Прямоугольник», «Квадрат», «Круг» переопределить виртуальный метод Object.ToString(), который возвращает в виде строки основные параметры фигуры и ее площадь.
7. Разработать интерфейс IPrint. Интерфейс содержит метод Print(), который не принимает параметров и возвращает void. Для классов «Прямоугольник», «Квадрат», «Круг» реализовать наследование от интерфейса IPrint. Переопределяемый метод Print() выводит на консоль информацию, возвращаемую переопределенным методом ToString().

Код программы:

```rust
use std::fmt;

// Трейт для вычисления площади фигуры
trait Shape {
    fn area(&self) -> f64;
}

// Трейт для печати информации о фигуре
trait IPrint {
    fn print(&self);
}

// Структура Прямоугольник
struct Rectangle {
    width: f64,
    height: f64,
}

impl Rectangle {
    fn new(width: f64, height: f64) -> Rectangle {
        Rectangle { width, height }
    }
}

// Реализация трейта Shape для прямоугольника
impl Shape for Rectangle {
    fn area(&self) -> f64 {
        self.width * self.height
    }
}

// Реализация трейта ToString для прямоугольника
impl fmt::Display for Rectangle {
    fn fmt(&self, f: &mut fmt::Formatter) -> fmt::Result {
        write!(f, "Rectangle: width = {}, height = {}, area = {}", self.width,
self.height, self.area())
    }
}

// Реализация трейта IPrint для прямоугольника
impl IPrint for Rectangle {
    fn print(&self) {
        println!("{}", self);
    }
}

// Структура Квадрат
struct Square {
    side: f64,
```

```rust
}

impl Square {
    fn new(side: f64) -> Square {
        Square { side }
    }
}

// Реализация трейта Shape для квадрата
impl Shape for Square {
    fn area(&self) -> f64 {
        self.side * self.side
    }
}

// Реализация трейта ToString для квадрата
impl fmt::Display for Square {
    fn fmt(&self, f: &mut fmt::Formatter) -> fmt::Result {
        write!(f, "Square: side = {}, area = {}", self.side, self.area())
    }
}

// Реализация трейта IPrint для квадрата
impl IPrint for Square {
    fn print(&self) {
        println!("{}", self);
    }
}

// Структура Круг
struct Circle {
    radius: f64,
}

impl Circle {
    fn new(radius: f64) -> Circle {
        Circle { radius }
    }
}

// Реализация трейта Shape для круга
impl Shape for Circle {
    fn area(&self) -> f64 {
        std::f64::consts::PI * self.radius * self.radius
    }
}

// Реализация трейта ToString для круга
impl fmt::Display for Circle {
    fn fmt(&self, f: &mut fmt::Formatter) -> fmt::Result {
        write!(
```

```
            f,
            "Circle: radius = {}, area = {}",
            self.radius,
            self.area()
        )
    }
}

// Реализация трейта IPrint для круга
impl IPrint for Circle {
    fn print(&self) {
        println!("{}", self);
    }
}

/// Реализация супертрейта для объединения всех трёх трейтов
trait Figure: Shape + IPrint + fmt::Display {}
impl Figure for Rectangle {}
impl Figure for Square {}
impl Figure for Circle {}

/// Функция, применяющая все методы трейта Figure
fn print_figures(figures: &[&dyn Figure]) {
    for f in figures.iter() {
        println!("{}", f.to_string());
        f.print();

        println!("Area: {}", f.area());
        println!();
    }
}


fn main() {
    let rect = Rectangle::new(3.0, 4.0);
    let square = Square::new(5.0);
    let circle = Circle::new(2.0);

    print_figures(&[&rect, &square, &circle]);
}
```

Снимки экрана: