ООП в Rust

Нгуен Тхи Хань Хуен
Королева Дарья

Введение

Rust не является классическим ООП-языком.

Но он поддерживает все три ключевых принципа ООП.

Rust реализует ООП через

- структуры
- трейты

Инкапсуляция

- приватные поля структуры
- приватные функцие
- модули и ограниченный импорт

```rust
mod aaa {
    fn foo(inner: bbb::Inner) {
        // Есть доступ к публичному полю.
        let a = inner.public;

        // Ошибка компиляции: попытка обращения к приватному полю.
        let b = inner.private;

        // Ошибка компиляции: попытка использования приватной структуры.
        let c = bbb::Private {};
    }

    mod bbb {
        pub struct Inner {
            private: i32,
            pub public: i32,
        }

        struct Private {}
    }
}
```

Наследование

В классическом ООП наследование описывает отношение **«является»**.

Rust опирается на **композицию** и механизм трейтов.

```rust
trait Shape {
    // У любой фрормы можно посчитать площадь.
    fn area(&self) -> f32;
}

trait HasAngles: Shape {
    // У любой фигуры с углами можно посчитать количество углов.
    fn angles_count(&self) -> i32;
}

struct Rectangle {
    x: f32,
    y: f32,
}
// Прямоугольник является формой.
impl Shape for Rectangle {
    fn area(&self) -> f32 {
        self.x * self.y
    }
}
// Прямоугольник является фигурой с углами.
impl HasAngles for Rectangle {
    fn angles_count(&self) -> i32 {
        4
    }
}

struct Circle {
    r: f32,
}
// Круг является формой
impl Shape for Circle {
    fn area(&self) -> f32 {
        self.r.powi(2) * PI
    }
}
```

Полиморфизм

Rust поддерживает два вида:

- Статический полиморфизм - компилятор знает конкретный тип во время компиляции
- Динамический полиморфизм - компилятор определяет конкретный тип во время исполнения программы

Динамический полиморфизм

Trait object — это указатель на какой-то тип, который реализует заданный трейт.

Содержит указатель на данные и указатель на таблицу виртуальных методов (vtable).

```rust
pub struct Screen {
    // Вектор указателей на объекты, реализующие трейт Draw
    pub components: Vec<Box<dyn Draw>>,
}

impl Screen {
    pub fn run(&self) {
        for component in self.components.iter() {
            component.draw(); // Вызывается правильная реализация draw!
        }
    }
}

// Использование
fn main() {
    let button = Button { width: 50, height: 10, label: "OK".to_string() };
    let text_field = TextField { value: "Hello".to_string() };

    let screen = Screen {
        components: vec![
            Box::new(button),    // Упаковываем Button в Box<dyn Draw>
            Box::new(text_field), // Упаковываем TextField в Box<dyn Draw>
        ],
    };

    screen.run(); // Рисует и кнопку, и текстовое поле.
}
```

Статический полиморфизм

```rust
// Статический полиморфизм через generics
fn draw_shape<T: Draw>(shape: &T) {
    shape.draw();
}

// Или с помощью where clause для сложных случаев
fn process_shapes<T, U>(shape1: &T, shape2: &U)
where
    T: Draw + Clone,
    U: Draw + PartialEq,
{
    shape1.draw();
    shape2.draw();
}

// Использование
let circle = Circle { radius: 5.0 };
let square = Square { side: 10.0 };

draw_shape(&circle); // Компилятор генерирует draw_shape::<Circle>
draw_shape(&square); // Компилятор генерирует draw_shape::<Square>
```

Итоги

- Rust — это неклассический ООП-язык
- Инкапсуляция реализуется через pub и приватность полей структур
- Наследование заменено на композицию и, что важнее, на трейты
- Трейты — основной инструмент абстракции
- Полиморфизм бывает двух видов:
  - Динамический (dyn Trait)
  - Статический (Generics + Trait Bounds)

Литература

- Статья “Rust и ООП”
  - https://habr.comRust и ООП/ru/companies/otus/articles/661863/
- Возможности объектно-ориентированного программирования в Rust
  - https://doc.rust-lang.ru/book/ch17-00-oop.html