apprentissage rust

m billaud

31-03-2021

Table des matières

1 Définir et utiliser une classe

1. on définit une **structure** avec des **champs**
2. on **implémente** des **méthodes**

1.1 Définition de la structure

```
struct Position {
   row : usize,
   col : usize
}
```

pour l'utiliser l'utiliser.

```
fn main() {
    let p = Position{col : 3, row : 4};
    println!("row = {}, col = {}", p.row, p.col);
}
```

Remarques :

- le “constructeur” a des paramètres nommés ;
- affectation d'une struct à l'autre ;
- déstructuration :

```
let Position {row :r, col :c} = p;
    println!("r = {}, c = {}", r, c);
```

1.2 Implémentation de méthode

```
impl Position {
    fn get_row(&self ) -> usize
```

```
    {
        self.row
    }
}
```

- tournure idiomatique de Rust : terminer par une expression (`return` implicite) sans point-virgule.

1.3 Mutateurs

Pour une méthode qui modifie l'objet : `& mut self`

```
  fn set_row(& mut self, row :usize)
    {
        self.row = row;
    }
```

Les objets à qui on l'applique doivent être mutables

```
let mut p = Position{col : 3, row : 4};
```

1.4 "Constructeur"

Le constructeur est une convention en Rust.

```
impl Position {
    // ...
    fn new(r :usize, c :usize) -> Self {
        Self{ row :r, col :c}
    }
}
```

- C'est une méthode statique (le premier paramètre n'est pas `self`),
- `Self` désigne le type du bloc `impl` en cours de définition. La variable `self` est de type `Self`.
- `& mut self` est une abréviation pour `self : &mut Self`.

Appel du constructeur : `let q = Position ::new(22, 33) ;`