



```
rust 4
```

- package:
- crate
- modules use
- path

package crate

```
cargo new project crate root src/main.rs src/lib.rs (library) package
```

module

```
/Cargo.toml
/src/
/src/main.rs
/src/ffmpeg_encoder/mod.rs
/src/ffmpeg_encoder/book.rs
/src/ffmpeg_encoder/people.rs
```

```
crate root src/main.rs src/lib.rs
```

```
# src/main.rs

// 
use crate::ffmpeg_encoder::book::Book;
use crate::ffmpeg_encoder::people::Employ;
use crate::ffmpeg_encoder::people::Sex;
use crate::ffmpeg_encoder::Shop;

// foo module
pub mod ffmpeg_encoder;
```

```
fn main() {

    println!("{}", Shop::to_string());

    println!("{}", Book::new().to_string());

    let em = Employ::new(String::from("[]"), 33, Sex::Woman);
    println!("{}", em.to_string());
}
```

```
# [ ]rust[ ]
# src/ffmpeg_encoder/mod.rs
# src/ffmpeg_encoder.rs

pub mod book;
pub mod people;

pub struct Shop {}

impl Shop {
    pub fn to_string() -> String {
        return String::from("Shop to_string");
    }
}
```

```
# src/ffmpeg_encoder/book.rs

pub struct Book;

impl Book {
    pub(crate) fn new() -> Book {
        Book {}
    }

    pub(crate) fn to_string(&self) -> String {
```

```
        return String::from("Book to_string");
    }
}
```

```
# 000000
# src/ffmpeg_encoder/people.rs

#[derive(Debug)]
pub struct Employ {
    name: String,
    age: u32,
    sex: Sex,
}

impl Employ {
    pub fn new(name: String, age: u32, sex: Sex) -> Employ {
        return Employ {
            name: name,
            age: age,
            sex: sex,
        };
    }
    // 0000self0000
    pub fn to_string(&self) -> String {
        return self.name.to_string();
    }
}

#[derive(Debug)]
pub enum Sex {
    Main = 0,
    Woman = 1,
}
```

000 #5

000 000 26 000 2023 10:33:16

000 000 27 000 2023 02:33:45