Shared Ownership | Smart Pointers

Smart Pointers

- Allow multiple owners of data
- Reference counted – “***Rc***”
    - Data deleted only when last owner is dropped
- Atomic reference counted – “***Arc***”
    - Safe to use with multiple threads

```rust
use std::rc::Rc;

#[derive(Debug)]
struct Vehicle {
    vin: String,
}

#[derive(Debug)]
struct Door {
    vehicle: Rc<Vehicle>,
}
```

```rust
let car = Rc::new(Vehicle {
    vin: "123".to_owned(),
});

let left_door = Door {
    vehicle: Rc::clone(&car),
};
let right_door = Door {
    vehicle: Rc::clone(&car),
};

drop(car);
```

```rust
println!("vehicle = {:?}", left_door.vehicle);
```

```
vehicle = Vehicle { vin: "123" }
```

Recap

- ***Rc*** & ***Arc*** are used to share ownership
- Data is dropped once all owners are dropped
- ***Rc*** for single-threading
  - ***Rc::clone*** to make a new reference
- ***Arc*** for multi-threading
  - ***Arc::clone*** to make a new reference