Day 20

Know Your Status

The Story

"IS THE SLEIGH PARKED, READY, OR FLYING? WHY DOESN'T ANYONE KNOW?"

"Uh... shouldn't you know? Didn't we use the type state pattern?" Bernard spoke up.

The Story

Santa's face twitched. "Of course, we used the type state pattern! But does anyone have a quick way to check the state right now? There are some sleighs in the air, it's hard to keep track on all of them."

The Story

Blitzen, lounging in a corner, snorted. "Sounds like y'all need to add a `status()` method."

Prancer's ears perked up. "Yeah! A single method we can call to figure out the state, no matter which one it's in."

The Story

Santa stroked his beard, muttering. "Fine. Add it. And make sure it's compile-time safe. If I get a wrong answer, someone's spending Christmas hand-delivering packets in binary."

Your Mission

It's time to give Santa what he wants: a single `status()` method that works in all states of the sleigh (`Empty`, `Ready`, and `Flying`), returning the current state as a string.

Here's the plan:

1. Define the `status()` method: It should be callable on the `Sleigh` regardless of its state.

Your Mission

2. Return a string indicating the sleigh's state:
   - `"Empty"` when the sleigh is empty.
   - `"Ready"` when the sleigh is loaded and ready to fly.
   - `"Flying"` when the sleigh is in the air.
3. The `Sleigh` struct should have a trait bound, not just a generic `<T>`.

Your Mission

Here's how Santa wants to the the API:

```rust
let sleigh = Sleigh::new();
assert_eq!(sleigh.status(), "Empty");

sleigh.load();
assert_eq!(sleigh.status(), "Ready");

sleigh.take_off();
assert_eq!(sleigh.status(), "Flying");
```

Hints

If you're unsure where to start, take a look at these tips:

- Use a trait to define shared behavior across all states. For example:

```rust
pub trait State {
    fn status() → &'static str;
}
```

Hints

- Implement the trait for each state. e.g.

```rust
impl State for Empty {
    fn status() → &'static str {
        "Empty"
    }
}
```

Hints

- Implement a method for all types that implement the `State` trait. e.g.

```rust
impl<T: State> Sleigh<T> {
    pub fn status(&self) → &'static str {
        T::status()
    }
}
```