Destructuring Nested Structs and Enums

Until now, all our examples have been matching structs or enums that were one level deep. Matching can work on nested items too!

For example, we can refactor the code in Listing 18-15 to support RGB and HSV colors in the ChangeColor message, as shown in Listing 18-16.

enum Color {

Rgb(i32, i32, i32),

Hsv(i32, i32, i32)

}

enum Message {

Quit,

Move { x: i32, y: i32 },

Write(String),

ChangeColor(Color),

}

fn main() {

let msg = Message::ChangeColor(Color::Hsv(0, 160, 255));

match msg {

Message::ChangeColor(Color::Rgb(r, g, b)) => {

println!(

"Change the color to red {}, green {}, and blue {}",

r,

g,

b

)

},

Message::ChangeColor(Color::Hsv(h, s, v)) => {

println!(

"Change the color to hue {}, saturation {}, and value {}",

h,

s,

v

)

}

\_ => ()

}

}

Listing 18-16: Matching on nested enums

The pattern of the first arm in the match expression matches a Message::ChangeColor enum variant that contains a Color::Rgb variant; then the pattern binds to the three inner i32 values. The pattern of the second arm also matches a Message::ChangeColor enum variant, but the inner enum matches the Color::Hsv variant instead. We can specify these complex conditions in one match expression, even though two enums are involved.