Design Patterns in Rust

Builder Pattern

Builders are neat because they don't require us to specify *everything* to build our struct, which means we can add Bob's red clippers in a minor release without breaking anything. They also prefix every field with its name, which makes the code more readable. For example

```
pub struct YakShaverBuilder {
    clipper_size: u32,
    gas_powered_clippers: bool,
    solar_powered_clippers: bool,
    color_to_dye_yak: String,
    clipper_color: String,
}

impl YakShaverBuilder {
    pub fn new() -> Self {
        Self {
            clipper_size: 3,
            gas_powered_clippers: false,
            solar_powered_clippers: true,
            color_to_dye_yak: String::from("brown"),
            clipper_color: String::from("black"),
        }
    }

    pub fn clipper_size(mut self, v: u32) -> Self {
        self.clipper_size = v;
        self
    }

    pub fn gas_powered_clippers(mut self, v: bool) -> Self {
        self.gas_powered_clippers = v;
        self
    }

    pub fn solar_powered_clippers(mut self, v: bool) -> Self {
        self.solar_powered_clippers = v;
        self
    }

    pub fn color_to_dye_yak(mut self, v: String) -> Self {
```

```
        self.color_to_dye_yak = v;
        self
    }

    pub fn clipper_color(mut self, v: String) -> Self {
        self.clipper_color = v;
        self
    }

    pub fn build(self) -> YakShaver {
        YakShaver {
            clipper_size: self.clipper_size,
            gas_powered_clippers: self.gas_powered_clippers,
            solar_powered_clippers:
self.solar_powered_clippers,
            color_to_dye_yak: self.color_to_dye_yak,
            clipper_color: self.clipper_color,
        }
    }
}

let yak_shaver = YakShaverBuilder::new()
    .clipper_size(4)
    .color_to_dye_yak(String::from("hot pink"))
    .clipper_color(String::from("red"))
    .build();
```