Kinds

Kinds are the types of types, and because they are so transcendent and almighty they are represented by stars.

```
Int   ::  *
Char  ::  *
Bool  ::  *
```

Your normal data types like `Int` or `Bool` or `Char` all have kind `*`. You can check this now on GHCi by asking the compiler using the `:k` command what the kinds of these data types are. Something that has kind star can be thought of as a fully specified entity. `Int` is an integer, we know how to deal with them, we know where they live, we know their family, you know we have fully stalked `Int` so we can class them as kind `*`. Likewise if I had a `Maybe Int` I know everything about `Maybe` and I know everything about `Int`.

Things get more interesting when I don't specify who the occupant of the `Maybe` is. I can no longer say it has kind star since it could be a `Maybe Char` or a `Maybe Bool` or even a `Maybe Monkey` who knows?! So what the compiler says is:

```
Maybe :: * -> *
```

Basically "Ohhh this `Maybe` can be combined with something of kind `*` to create a stary thing!". Tuples are even more exciting since they can contain lots of different kinds of stary things, it is like they are a collector of pretty stary things.

```
 (,) :: * -> * -> *
(,,) :: * -> * -> * -> *
```

If you think of a tuple as a cabinet with a row of drawers, then each draw contains something of kind star and the final star written in the kind represents the tuple as a whole.