Another Way of Looking at Objects

- In C, you have typedef statement which allows you to give meaningful names to built-in types or to "compound" types.

```
typedef Money double;


typedef struct {
    int balance;
    char* name;
} Account;
```

- We can think of objects as structs. We create a new type every time we define a new class. And classes are just chunks of data (like structs)… but with behavior. In other words, we package data along with the code that operates on it.
- However, objects can inherit too…more later.

Creating an Object

- Assume we have a class called **`Money`**
- Then one could "instantiate" an object of type **`Money`** by calling

```
new Money();
```

- But an instantiation by itself does not associate the newly constructed object with a name!
- The assignment operator (=) is usually used together with **`new`** to create a new object and give it a name.

```
mon = new Money();
```

- The right hand side of above statement creates object and assignment operator "binds" **`mon`** to that object.
- One can think of **`mon`** as a kind of pointer to an object of type **`Money`** that you don't have to de-reference.

Creating an Object (cont.)

```
mon1 = new Money();
mon2 = new Money();
```