References vs. Values

To better grasp the idea of reference types, let's look at the other kind of type: *value types*. While reference-type variables refer to a place in memory, value-type variables hold the actual data.

`int` is a value type, so the variable `num` holds the value `6`:

```
int num = 6;
```

Reference types, on the other hand, refer to a location in memory. Every class is a reference type, so the variable `diss` refers to a location in memory that has the `Dissertation` object:

```
Dissertation diss = new Dissertation(50);
```

Every "primitive" data type is a value type, including:

- `int`
- `double`
- `bool`
- `char`

Revisiting our metaphor: a reference is like directions to a house, which "points" to a house. It isn't the actual house. A value type is the house itself!

You might have noticed that `string` is missing here. It works a bit differently, so it will be covered in a later lesson.

☑Instructions

The diagram to the right represents a computer's memory:

An object is stored in the first memory block

- `diss1` hold a reference to the first memory block
- `diss2` also holds a reference to the first memory block
- `num` refers to a value in the fourth memory block

Notice that the object takes up more memory than either reference (size is represented by the width of the slot), and that changing the object would affect both `diss1` and `diss2`.