[0460] Structs

A Struct is user-defined data type (a "record") containing a collection of related data types ("fields") that can be used as if they were one thing.

For instance a Person struct might contain fields for name, age, address and phone number.

```
struct Person
{
    string name;
    int age;
    string address;
    string phone;
}
```

You can then declare a Person variable just like any other local variable and then use the dot operator to populate its fields:

```
Person peter;
peter.name = "Peter";
peter.age = 24;
peter.address = "42 Wallaby Way, Sydney";
peter.phone = "iPhone";
```

You can use a struct just like any other local variable. And, like all local variables, the struct and its data disappears when it goes out of scope.

As the book shows, you can also put methods inside of Structs. In my opinion, that is not a good idea. As we will see, the combination of data and code is what a Class is. As the book points out, Structs can live on either the stack or the heap while Class-based Objects live on the heap. Unless having object-like-things-that-aren't-objects on the stack gives your program some unusual benefit, I recommend sticking with Classes and Objects on the garbage collected heap for most things. Given that, I don't find much use for structs in my programming.