```
class MyClass
{
    int myInt = 0;

    string myString = "Something";
}

class Program
{
    static void Main(string[] args)
    {
       MyClass m = new MyClass();
    }
}
```

`m` is allocated on the heap, and that includes `myInt`.
The situations where primitive types (and structs) are allocated on the stack is during method invocation, which allocates room for local variables on the stack (because it's faster). For example:

```
class MyClass
{
    int myInt = 0;

    string myString = "Something";

    void Foo(int x, int y) {
       int rv = x + y + myInt;
       myInt = 2^rv;
    }
}
```

`rv`, `x`, `y` will all be on the stack. `myInt` is somewhere on the heap (and must be access via the `this` pointer).

Primitive Types

Because a primitive type aliases a regular type, every primitive type has members. For example, **Integer** has the members declared in **System.Int32**. Literals can be treated as instances of their corresponding types.