User-defined value types aka structs

Value types in boo are like value types in C# and include int, double, and other basic types. User-defined value types can be created in boo by defining a **struct** or inheriting from **System.ValueType**.

Value types are constructed on the stack rather than the heap as objects are. This makes them faster to create and dispose of, and they don't need to be garbage collected. User-defined value types can have methods and fields, just like classes. Value types cannot inherit from other types (exception - all value types inherit from System.ValueType, which itself inherits from **object**), and other types cannot inherit from value types. Value types can implement interfaces.

Here's an example showing the implementation of a **Point** value type by inheriting from System.ValueType.

```
import System

class Point(ValueType):
    public X as int
    public Y as int

p1 = Point(X: 200, Y: 300)
p2 = p1 # value type semantics means this creates a copy

p1.X = 250
assert 200 == p2.X # copy still unchanged
```

When **p1** is assigned to **p2**, the contents of **p1** are physically copied to **p2**. If **Point** were a class (classes are reference types), the assignment would simply copy a pointer to **p1** into **p2** and the assertion would fail.

Here's the equivalent definition for Point implemented using **struct**:

```
import System

struct Point:
    X as int
    Y as int
```

Note that by default, fields are public in a struct.