Classes as Unions

Slice does not offer a dedicated union construct because it is redundant. By deriving classes from a common base class, you can create the same effect as with a union:

Slice

```
interface ShapeShifter {
    Shape translate(Shape s, long xDistance, long yDistance);
};
```

The parameter `s` of the `translate` operation can be viewed as a union of two members: a `Circle` and a `Rectangle`. The receiver of a `Shape` instance can use the type ID of the instance to decide whether it received a `Circle` or a `Rectangle`. Alternatively, if you want something more along the lines of a conventional discriminated union, you can use the following approach:

Slice

```
class UnionDiscriminator {
    int d;
};

class Member1 extends UnionDiscriminator {
    // d == 1
    string s;
    float f;
};

class Member2 extends UnionDiscriminator {
    // d == 2
    byte b;
    int i;
};
```

With this approach, the `UnionDiscriminator` class provides a discriminator value. The "members" of the union are the classes that are derived from `UnionDiscriminator`. For each derived class, the discriminator takes on a distinct value. The receiver of such a union uses the discriminator value in a `switch` statement to select the active union member.

See Also

- Type IDs