Classes & Traits

Looses consensuses and choices

Overall shape

```
class ClassName?  /*super stuff?*/ {
    // super stuff?
    // class-private instance var decls
    // prototype properties. IVs in scope somehow
    // class properties. IVs in scope only as class-private
    new (arguments list) {
        // instance attribute control?
        // constructor chaining
        // IV initialization
        // instance properties
    }
}
```

AWBish?

```
class ClassName {
    <superclass: SuperClassName>
    private i, j,
    method foo() { return @i + @j; },
    class method bar(allegedInst) { return allegedInst@i; },
    new (x, y) {
        <frozen>
        super new(y, x+y),
        @i: y*x, @j: x%y,
        xx: x*x
    }
}
```

Initial loose Google convergence

```
class ClassName : SuperClassName {

    private {i, j},
    foo() { return i + j; },
    class bar(allegedInst) { return private(allegedInst).i; },
    new (x, y) {
        // instance attribute control?
        super(y, x+y);
        let i = y*x, j = x%y;
        this.xx = x*x;
    }
}
```

Module-style instance publicness

```
class ClassName : SuperClassName {

    private i, j,
    foo() { return i + j; },
    class bar(allegedInst) { return private(allegedInst).i; },
    new (x, y) {
        // instance attribute control?
        super(y, x+y);
        let i = y*x, j = x%y;
        public xx = x*x;
    }
}
```

Traitish

```
abstract class ClassName {
    private i, j,
    foo() { return private(this).i + private(this).j; },
    class bar(allegedInst) { return private(allegedInst).i; },
    new (x, y) {
        // instance attribute control?
        extends Super1(y, x+y);
        extends Super2(y&x) with zip as zap;
        let i = y*x, j = x%y;
        public xx = x*x;
        required yy;
    }
}
```