software studio

abstract data types

Daniel Jackson

review: why abstract types?

code organization

- abstract types raise level of client code
- extend repertoire of types beyond the built-ins

rep invariants

- can ensure that invariants are maintained locally

plugins

- can replace ADT with different implementation
- eg, can stub network interface

rep independence

- can change rep without changing client
- choice of rep is an encapsulated "design secret"

hiding the rep using closures

in conventional language

- hide the rep using namespace qualifiers (eg, *private*)

in functional language

- can hide the rep using closures

an abstract type

```
var Color = function (r, g, b) {
    var color = {};
    var rgb = [r, g, b];
    color.red = function () {return r;}
    color.green = function () {return g;}
    color.blue = function () {return b;}
    color.toCSS = function () {
        return "rgb(" + rgb.join(",") + ")";
    }
    return color;
}
var show = function (c) {
  document.body.style.backgroundColor = c.toCSS();
  };
```

```
> blue = Color(0,0,255);
Object
> show(blue)
undefined
```

how is rep hidden?

using *this*

```
var Color = function (r, g, b) {
    var rgb = [r, g, b];
    this.red = function () {return r;}
    this.green = function () {return g;}
    this.blue = function () {return b;}
    this.toCSS = function () {
        return "rgb(" + rgb.join(",") + ")";
    }
}
```

```
> blue = new Color(0,0,255);
Color
> show(blue)
undefined
```

prototype method

```
Color.prototype.distance = function (c) {
    var sq = function (x) {return x * x;};
    return Math.sqrt(
        sq(c.red() - this.red()) +
        sq(c.green() - this.green()) +
        sq(c.blue() - this.blue()));
}
```

```
> red = new Color (255,0,0);
green = new Color (0,255,0);
Color
> red.distance(green);
360.62445840513925
```

a mutable type

```
var ColorChart = function () {
    // mapping from color names to color objects
    var name_to_color = {};

    // adds mapping from name to color
    this.add = function (name, color) {
        name_to_color[name] = color;
    };

    // returns undefined if no match
    this.lookup = function (name) {
        return name_to_color[name];
    };

    // returns name of color closest to argument
    this.findBestMatch = function (color) {
        var MAX = 500; // larger than any RGB distance
        var shortest_distance = MAX; var best_match;
        for(var name in name_to_color) {
            if(name_to_color.hasOwnProperty(name)) {
                var c = name_to_color[name];
                distance = c.distance(color);
                if (distance < shortest_distance) {
                    shortest_distance = distance;
                    best_match = name;
                };
            };
        };
        return best_match;
    };
}
```

finding lego color matches

```
var lego_colors = [
["White", [242, 243, 242]],
["Grey", [161, 165, 162]],
...]
```

```
var lego_color_chart = new ColorChart();
lego_colors.each(function (nc) {
    var name = nc[0]; var color = nc[1];
    lego_color_chart.add(nc[0], new Color(color[0], color[1], color[2]));
});
```

```
> var c = new Color(100,50,150);
Color
> document.body.style.backgroundColor = c.toCSS();
"rgb(100,50,150)"
> var n = lego_color_chart.findBestMatch(c);
undefined
> n
"Bright violet"
> var c2 = lego_color_chart.lookup(n);
> document.body.style.backgroundColor = c2.toCSS();
"rgb(107,50,123)"
```

rep invariant for Color

```
var Color = function (r, g, b) {
    var rgb = [r, g, b];
    var inRange = function (x) {return x >= 0 && x <= 255;}
    this.checkRep = function () {
       return inRange(r) && inRange(g) && inRange(b);}
...
}
```

```
> red = new Color (255,0,0);
Color
> red.checkRep()
true
> red = new Color (256,0,0);
Color
> red.checkRep()
false
```

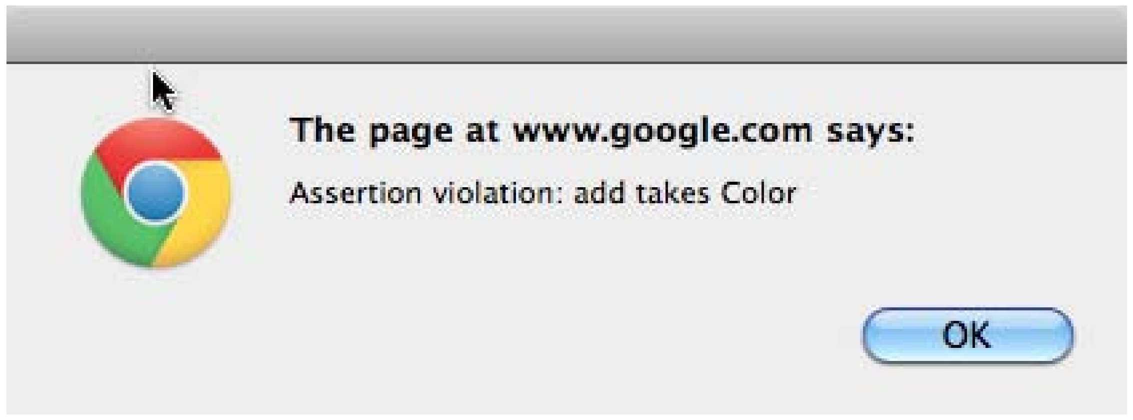
asserting argument type

```
var assert = function (msg, pred) {
    if (!pred) alert ("Assertion violation: " + msg);
}

var ColorChart = function () {
    var name_to_color = {};

    // adds mapping from name to color
    this.add = function (name, color) {
        assert ("add takes Color", color instanceof Color)
        name_to_color[name] = color;
    };
   ...
}
```

```
> legoColorChart.add ("black", [0,0,0]);
undefined
```


The page at www.google.com says:

Assertion violation: add takes Color