Summary 11 – Constructors

- What is this?

When an object is constructed, using the ***new*** keyword:

    new Blah(…)

the corresponding constructor of the Blah class runs.

For example,

    new Eye()

calls the first constructor in the example to the right, and

    new Eye(Color.red, Color.black)

calls the second constructor in the example to the right.

A constructor's name is the name of the class. Constructors don't return a value and don't specify a return type.

A constructor should initialize the fields and do any other initialization required for the object.

The expression this(blah, blah, blah) calls the constructor in the same class that matches the arguments blah, blah, blah. For example, the first constructor in the example to the right calls the seconds constructor. You can use this(…) only as the *first* statement in a constructor.

Java has two dirty little secrets regarding constructors:

- If a class definition has no constructor, then Java supplies one that has no parameters and an empty body.
- If a constructor does not contain a this(…) or super(…) expression as its first statement, then Java inserts super() as the first statement of the constructor.

- Example

```
public class Eye extends JPanel
                 implements MouseMotionListener {
    private static final int DEFAULT_RADIUS = 25;
    protected EyeBall eyeBall;
    protected Color eyeColor;
    protected int eyeRadius;

    public Eye() {
        this(JavaEyes.DEFAULT_EYE_COLOR,
             JavaEyes.DEFAULT_EYEBALL_COLOR);
    }

    public Eye(Color eyeColor, Color eyeBallColor) {
        this.eyeColor = eyeColor;
        this.eyeBall = new EyeBall(eyeBallColor);

        this.eyeRadius = Eye.DEFAULT_RADIUS;
        this.setPreferredSize(new Dimension(
                2 * this.eyeRadius,
                2 * this.eyeRadius));
    }
    ...
}
```

- For further study:
  - *Big Java*, Section 2.6 *Constructing Objects*, for how a constructor is called
  - *Big Java*, Section 3.2 *Specifying the Public Interface of a Class*, for how a constructor is defined
  - This summary's *author:* David Mutchler
  - See also super() in the Summary on *Inheritance*.