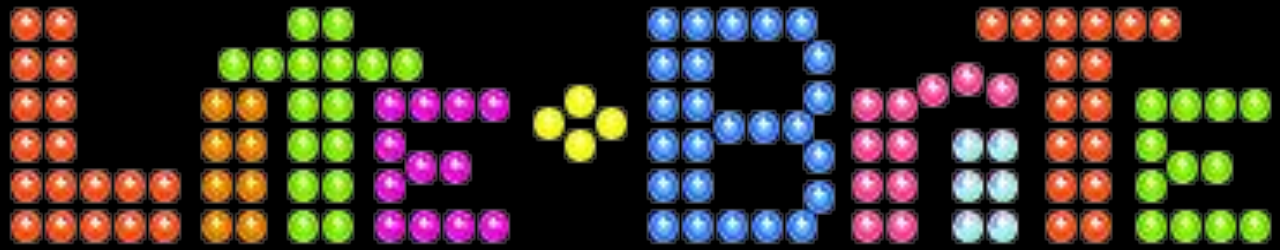
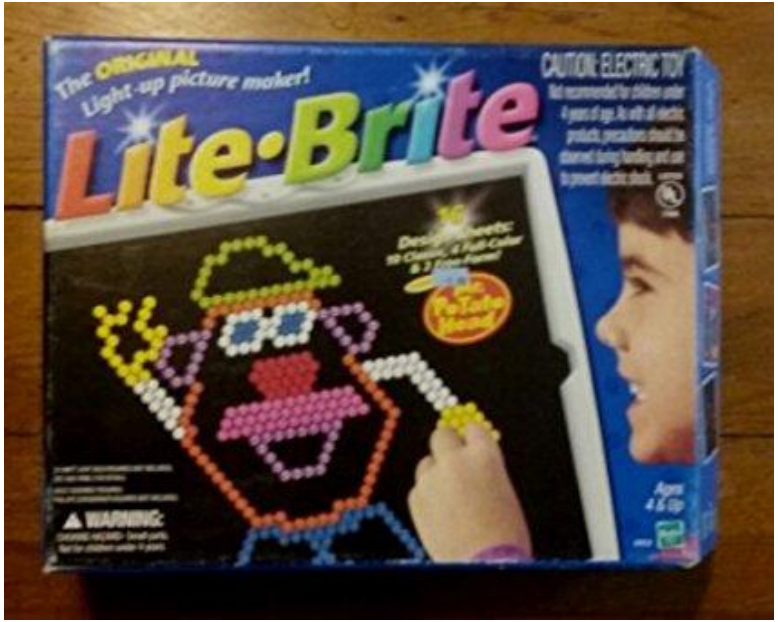


Help Slides

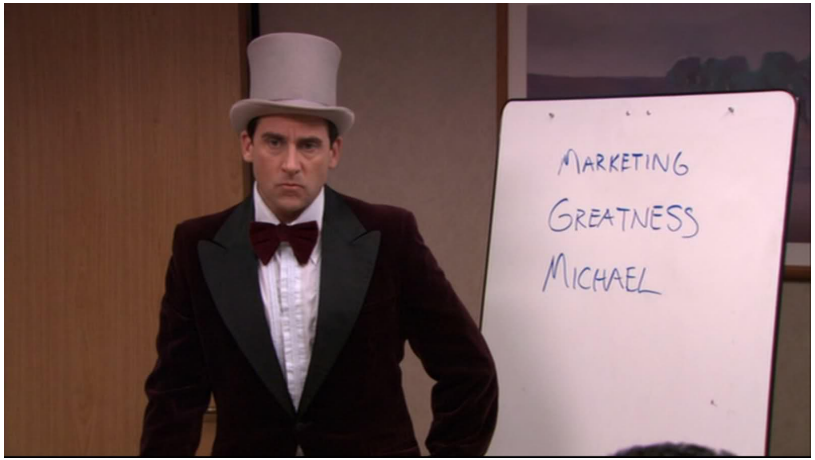
Overview

- Planning
- How To Get Started
- Support Code
- Design
- Parameters & Arguments
- Accessor & Mutator Methods
- Color



Planning your design

- In your last lab, we introduced you to writing your own class. Now, you are going to put that skill to use.
- Before you start, make sure you understand how to design your entire program!
- Identify the nouns and verbs in the “assignment specifications” to determine the objects (nouns) and methods (verbs) you will need in the program.



Getting Started (1/2)

Assignment Specification (The Specs)

Make LiteBrite!

- When the user clicks on the lite box, a colored lite peg should be inserted at that position.
- There should be a color palette with at least two color choices in the form of ColorButtons.
- The color palette should have a current color specified by whichever ColorButton was clicked last.
- When a lite peg is added to the lite box it should correspond to the color palette's current color.

List the nouns. Which ones become classes?

Getting Started (2/2)

Assignment Specification (The Specs)

Make LiteBrite!

- When the user clicks on the lite box, a colored lite peg should be inserted at that position.
- There should be a color palette with at least two color choices in the form of ColorButtons.
- The color palette should have a current color specified by whichever ColorButton was clicked last.
- When a lite peg is added to the lite box it should correspond to the color palette's current color.

List the nouns. Which ones become classes?

BEHIND THE LITE-BRITE

Stencil Code (1/2)

We provide you with partially written (“stencil”) `App`, `LiteBox` and `ColorPalette` classes as well as completely written support classes `cs015.prj.LiteBriteSupport.LitePeg` and `cs015.prj.LiteBriteSupport.ColorButton` for the pegs and color buttons, respectively.

Find information about the support classes on the Javadocs

Stencil code methods are described in the “Stencil Code” section of your assignment handout. For this program, you will have to:

- Fill in the `LiteBox` and `ColorPalette` classes
- Call on the support code (you never have to edit it, and you won’t be able to view it)
- Create a simple class containing `LiteBox` and `ColorPalette`

Stencil Code (2/2)

As with any CS15 class, you will need an `App` class to get started!

- This class is partially written for you, but you will have to fill in the rest!
- **In the `App` class, you should not do anything except instantiate your top-level object in your program!**

A (not so) Great Design

Problem: How do you make the pegs the same color as the current palette color?

One Possible Design:

- Have the `LiteBox` know about the `ColorPalette`'s current color when the `LiteBox` is created
- The `LiteBox` can store this color in the instance variable and use it to set the peg colors

Caveats:

- How will the `LiteBox` know when the user selects a new color on the `ColorPalette`?

A Better Design (1/2)

Associate the `LiteBox` with the `ColorPalette`

- Do this by passing the `ColorPalette` as a parameter to the `LiteBox's` constructor, as outlined in the stencil code.

`setColor` method - mutator

- Make sure the current color is "set" in the `setColor` method already outlined by the stencil code.
- For you inquisitive folk: when the user clicks one of the `ColorButtons`, this method magically gets invoked by our support code.

`getColor` method - accessor

- Make a method in the `ColorPalette` class to "get" its current `javafx.scene.paint.Color`
- The `LiteBox` can "get" the current color from the `ColorPalette` by calling this accessor method.

A Better Design (2/2)

Why is this better?

What happens when the `ColorPalette` color changes?

What would happen if the LiteBox knew directly about the color? Think about how this affects encapsulation.

What is a Top-Level Object?

- You have multiple other classes that need to be instantiated - but should you do that in App?

- Instantiate instances of those classes inside a top-level object, so one class contains all other component classes.
  - By convention, we call this class the name of the project, e.g. `LiteBrite.java`

- Your top-level object for LiteBrite shouldn't need anything other than a constructor. Why?

- Now just instantiate an instance of your top-level object in App!

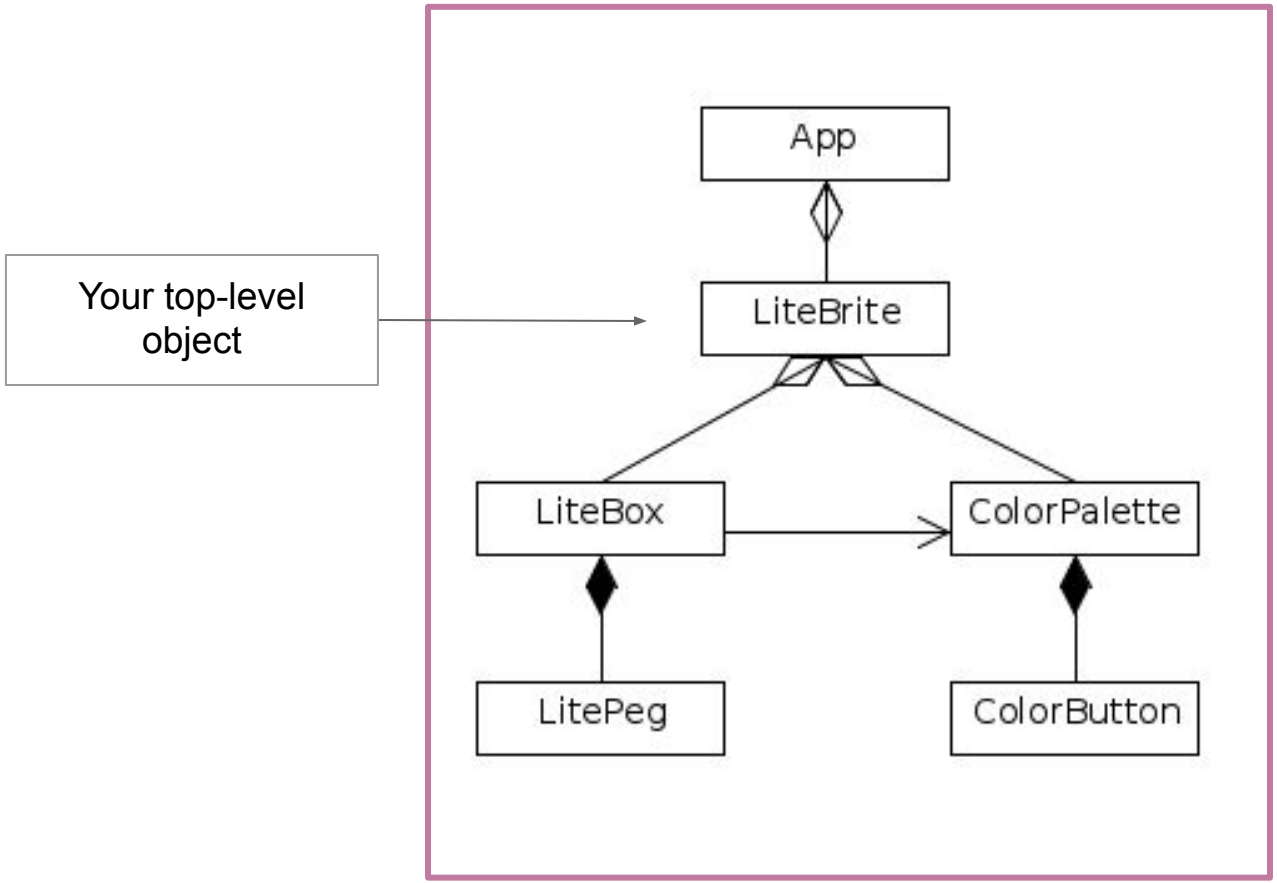
So What Should the Design Look Like?

We can represent the design with a diagram!

Containment Diagram

- Shows containment and associations
- **Containment**: what classes contain instances of other classes
- **Associations**: what classes a given class knows about (has a reference to)

Containment Diagram



Arguments vs Parameters (1/3)

What is a parameter?

- A placeholder variable when you are declaring a method
- Part of a definition of a method
- Also called a formal parameter!
- Example: x in $f(x) = 3x^2 + 5x + 2$

Arguments vs Parameters (2/3)

What is an argument?

- A value of specific type that is passed in when you are actually calling a method
- Can be found in constructors or inside other methods
- Also called an actual parameter!
- Example: 5 in y = f(5)

Arguments vs Parameters (3/3)

Method Declaration

```
public void add(int a, int b) {
    //arithmetic elided
}
```

Method Invocation (from another class)

```
_calculator.add(4,2);
```

On your mark, “GET” “SET”, go!

Accessor(get) and Mutator(set) methods

Used to “get” (access) and “set” (mutate or change) variables.

Accessors and Mutators (1/2)

```
public class CDPlayer {
  private CD _currentCD;
  public CDPlayer(CD myCD) {
     _currentCD = myCD;
  }
/**
  *  This is a mutator!
  */
  public void setCD(CD newCD) {
     _currentCD = newCD;
  }

/**
  *  This is an accessor!
  */
  public CD getCD() {
     return _currentCD;
  }
}
```

Accessors and Mutators (2/2)

```
public class Car {
    private CDPlayer _myPlayer;
    private CD _whatsPlaying;

    public Car() {
        CD jazzCD = new CD();
        CD classicalCD = new CD();
        _myPlayer = new CDPlayer(jazzCD);
        _myPlayer.setCD(classicalCD);
        this.seeWhatsPlaying();
    }

    public void seeWhatsPlaying() {
        _whatsPlaying = _myPlayer.getCD();
    }
}
```

What's the value of `_whatsPlaying` when a Car is instantiated?

Quick note on COLOR

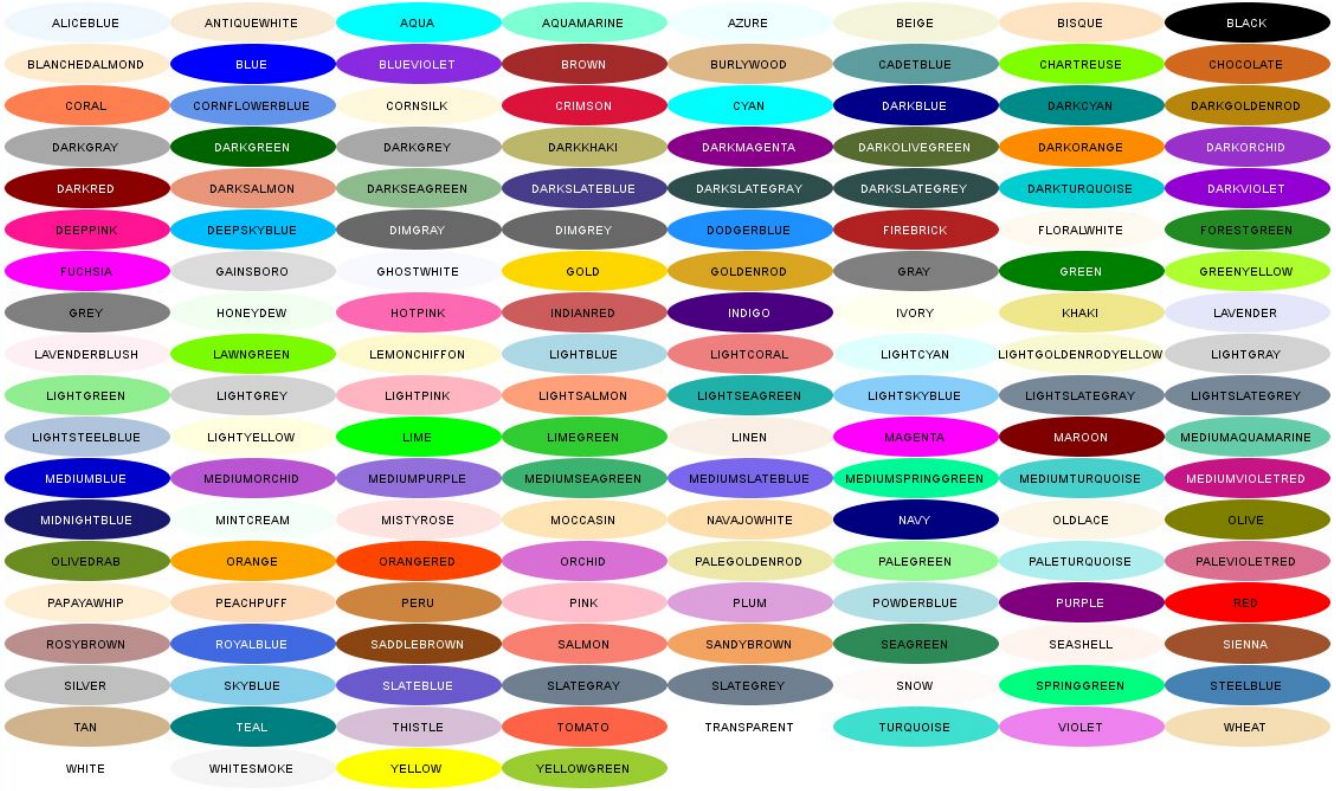
```
import javafx.scene.paint.Color
```

To specify the color aquamarine in Java, you can use

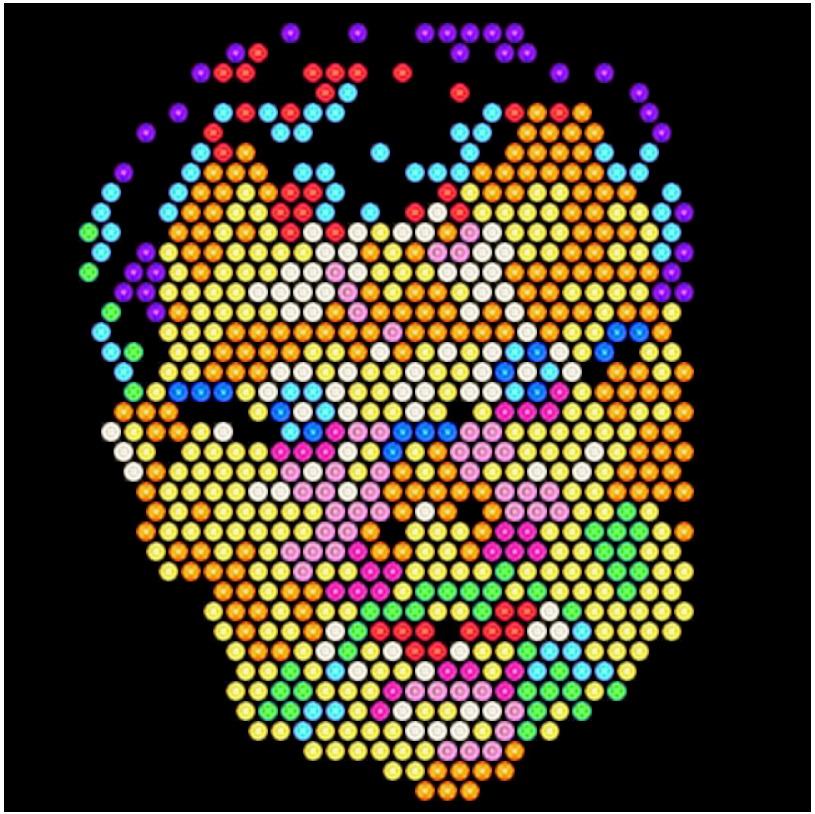
```
javafx.scene.paint.
Color.AQUAMARINE
```

or, after importing javafx.scene.paint.Color, you can specify the color aquamarine simply by

```
Color.AQUAMARINE
```



Let There Be



LiteBrite!