LiteBrite

Help Session: Sunday, September 27nd, 6:00pm, Salomon 001

Early Handin: Tuesday, September 29nd, 11:59pm
Regular Handin: Thursday, October 1st, 11:59pm
Late Handin: Saturday, October 3rd 10:00pm

To run demo: cs015_runDemo LiteBrite
To install: cs015_install LiteBrite
To handin: cs015_handin LiteBrite

Silly Premise

Gabriella and Troy continue their acting careers by auditioning for Romeo and Juliet: A Laser Light Experience on Broadway. Unfortunately, they get beat out by Ryan and Sharpay for the leads—but in an exciting turn of events, the playwrights decide to write in two even larger parts for tbolt and gamonte. But oh no! Sharpay has sabotaged the show by destroying the interface used to control the laser light features! They need to create a brand new one and they need YOU to do it! They need something simple and straightforward to use... Something where you could just press a button and lights would appear... Something that harks back to our childhoods... What type of interface fulfills all these requirements you ask? LiteBrite, that’s what!

You may remember LiteBrite as a kid: that game with the different colored pegs that was a lot of fun until your little sister ate all the pegs? Well, now you get to make one! Help make Gabriella and Troy’s first Broadway performance a memorable one. Give them a light show to rival the Twinkle Town Spring Musicál. Break a leg!

Andy said, “Let there be LiteBrite,” and there was LiteBrite.

JavaDocs 1:3
New Concepts Covered

- Parameters
- Mutator Methods ("set" methods)
- Accessor Methods ("get" methods)
- Using return
- Handling user input (mouse clicks)
- Using built-in classes (javafx.scene.paint.Color)

Assignment Specifications

Create your own computerized LiteBrite. When the user clicks on the grid, colored pegs should be added at the proper location. There should be a palette with at least two color choices that are selected using LiteButtons. The palette should have a current color specified by whichever LiteButton was clicked last. When a peg is added to the grid, it should be the palette’s current color.

We are providing you with partially written "stencil" Grid and Palette classes to model the grid and palette, as well as completely written cs015.prj.LiteBriteSupport.Lite and cs015.prj.LiteBriteSupport.LiteButton support classes, which will model the light pegs and the palette color buttons, respectively. Everything you need to know about how to use these two classes is described in the Support Classes section below. Additionally, the javafx.scene.paint.Color class will provide you with the colors needed for your Lites and LiteButtons. Refer to the “Working With Objects” lecture to see how to use a javafx.scene.paint.Color.

Your job is to fill in the rest of the provided Grid and Palette classes, along with any class(es) you create, to build your program.

Design Header

In the comment at the top of your App class (called the header comment), write a brief description of your design. Note any differences or changes you made to the suggested design. If there are any bugs that you know of in your program, write a note in the header comment explaining what they are and how you might fix them (if you have any ideas). This makes it easier for a TA to grade, and it also shows the TA that you know what is wrong with your program. (Side note: not listing the bugs and hoping the TA won't notice them won't help your

---

1 Note: a "skeleton" and “stencil” classes mean the same thing - we’ll be using the term “stencil” from now on.
grade. The TA will think that you didn't test your program to see if it worked.) If you create a class of your own, make sure to write header comments. Also, write some in-line comments to explain the methods you write.

Remember to refer to the CS15 Style Guide for tips on commenting, especially under the “Internal Documentation” section. Note that style will be factored into your grade!

**Helpful Hints:**

1. Your first job is to decide what object(s) you are going to need in this program.
   a. Picking out the nouns from the program specification should help you get started. Also, be sure to look at the demo of the program and try to describe the objects that you see.
   b. When designing the object(s) you want to create, look over the list of predefined objects (see the Support Classes section), and decide how and where you want to use each one. You must also decide what to put in your App class.

2. Next, define the purpose for the empty methods in the Grid and Palette class stencils. Think about where in your code you will need to create instances of each of the object classes. Then, think about where in your code you will need to alter the properties of any of the object instances you have created.

3. Finally, when you are confident about your design, log in, run `cs015_install LiteBrite`, `cd` into the LiteBrite directory, and start writing your program. Begin by getting an empty frame to appear. Then add small parts to your program, making sure that they work as you expect them to. This idea of writing your program incrementally will be very important as your programs get larger and harder to debug, so getting into a good habit now will save you a great deal of time in the future.

4. The following `javafx.scene.paint.Color` constants and more (see a full list [here](#)) are available for your use (you'll see examples of these being used in the “Working With Objects” lecture): BLACK, BLUE, RED, CHARTREUSE, PERU, MEDIUMAQUAMARINE, LEMONCHIFFON, BLANCHEDALMOND, OLIVEDRAB, PAPAYAWHIP

Before you start programming, look over the slides from the first four lectures. Very special (not-so) secret tip: **review the Parameters lecture before beginning this assignment.** If you don't understand something that is covered in that lecture, see a TA during TA hours.
Running your code

To run the program you must first `cd (change directory)` into `course/cs015/LiteBrite` directory, then compile as you normally would by running `javac *.java` in your shell. Run your program by typing `java LiteBrite.App`.

Handing in your code

In order to hand in your code, run the handin script at the top of this handout. The script will list all of the files you are about to hand in, `App.java`, `Grid.java`, `Palette.java`, and any other classes you create and will prompt you to confirm. Once you’ve confirmed, you will receive an email stating that the handin was successful.

**Note:** The email is your receipt or proof that you’ve handed in the assignment successfully, so please do not delete it! *If you did not receive an email, we did not receive your handin.* You can run this script as many times as you would like; however, once you run the script, all past handins for this assignment are overridden. This means that if you run the script after the deadline, the project will be marked *late*, even if you handed in an earlier version on time. We will not accept emailed submissions or handins after the late deadline.

Remember that the TAs are here to help you with the assignment, the programming environment, or any concepts that you are not clear about. TA hours are posted on the website, and you can find them on hours in the Fishbowl (271) on the second floor of the CIT.

Stencil Classes

Below is a listing of the stencil classes provided for this assignment.

**Name:**

`App`

**Purpose:**

This class models an application. When you write your program you should fill in this class so that it contains your top-level object. When you install the LiteBrite project, this class will already be in your LiteBrite directory, and you will only need to add code to it to run your program.
Methods:

App()
Constructs the application.

Name:

Grid

Purpose:
This class models a grid that can detect when a mouse has been clicked on top of it. It passes a cs015.prj.LiteBriteSupport.GridPosition as a parameter to the insertLite method.

Methods:

Grid(Palette p)
Constructs the grid with a reference to the instance of class Palette indicated by the parameter p. Note: Your top-level class should contain this and the Palette, but it is not written for you; you need to write this class yourself.

void insertLite(cs015.prj.LiteBriteSupport.GridPosition p)
This method is called automatically when the mouse is clicked inside the grid. You do not ever need to call insertLite(). If you want your grid to respond to a mouse click, you need to fill in this method.

Name:

Palette

Purpose:
This class models a palette that can have cs015.prj.LiteBriteSupport.LiteButtons added to it by instantiating them in the constructor. You should add at least two, but no more than ten LiteButtons to the Palette.

Methods:

Palette()
Constructs an empty palette.

void setColor(javafx.scene.paint.Color newColor)
This method is called automatically when a LiteButton is clicked. You never need to
**call** setColor(), just be sure to select a default color. However, if you want your Palette to respond to mouse clicks, you need to fill in this method.

---

**Support Code Classes**

This is a listing of the support code classes. We provide you with a description of their constructors and the methods you can call on them. See the Support Code handout online for more information about what support code is and how it interacts with your stencil code.

**Name:**

cs015.prj.LiteBriteSupport.LiteButton

**Purpose:**

This class models a color button on the Palette which controls the color of the light pegs.

**Methods:**

cs015.prj.LiteBriteSupport.LiteButton(Palette p,
javafx.scene.paint.Color buttonColor)

Constructs a light button with a reference to your Palette.

**Name:**

cs015.prj.LiteBriteSupport.Lite

**Purpose:**

This class models a light peg that will add itself to the Grid when constructed. Don't forget to set its color and position!

**Methods:**

cs015.prj.LiteBriteSupport.Lite()

Constructs the light peg.
void setPosition(cs015.prj.LiteBriteSupport.GridPosition p)
This method moves the Lite to the position p on the Grid.

void setColor(javafx.scene.paint.Color newColor)
This method sets the color of the light peg to be newColor. You're correctly using the Palette when new Lites appear with the color of the last color clicked. Make sure to run the demo (by typing cs015_runDemo LiteBrite in a shell) to get an idea of what this means.