LiteBrite

Help Slides released: Friday, September 23 at 2pm
Early Handin: Tuesday, September 27 at 11:59pm
Regular Handin: Thursday, September 29 at 11:59pm
Late Handin: Saturday, October 1 at 11:59pm

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

Silly Premise

After the Grinch’s change of heart, the next Christmas he decides to give all the children of Whoville a present! So he thinks and thinks to himself, and comes up with the perfect present: a LiteBrite. He buys one for every child in Whoville, and on the night of Christmas Eve he gets ready to deliver them to every sleeping child. However, on this very night, the Cat in the Hat decides to visit Whoville and he brings along Thing 1 and Thing 2. Together, the two Things cause mayhem and destruction wherever they go. They invade the Grinch’s cave and destroy all the LiteBrites! The Grinch needs your help to fix them all, so he can show the children of Whoville how much he cares. You may not remember LiteBrite from your childhood, but use your new computer science skills to give the kids of Whoville a present that will bring joy to all!

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

From the collaboration policy:

No collaboration is allowed on project-specific details. You may not discuss the implementation and debugging of code for projects with anyone except the course staff. You should not ever view or provide help with a project solution other than your own. You should not discuss program structure, method breakdown, or pseudocode with anyone except the TAs.

Importantly, though, you may absolutely discuss general (i.e., not assignment-specific) CS15 concepts with anyone, including other current students. The following falls into this category:

- Going over CS15 lecture slides, our (non-assignment) handouts, Javadocs, etc.
- Discussing object-oriented programming concepts, such as polymorphism
- General syntax questions. For example, “How do I declare an instance variable?”
- How to work remotely, and how to move and hand in files

Note that in each case, any examples used must be from the lectures or your own creativity – you may not discuss how even broad design concepts like association pertain to a specific assignment.

New Concepts Covered

- Parameters
- Containment and association
- 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 color palette with at least two color choices that are selected using ColorButtons. The color palette should have a current color specified by whichever ColorButton was clicked last. When a lite peg is added to the grid, it should be the color palette’s current color.

We are providing you with partially written stencil LiteBox and ColorPalette classes to model the grid and palette, as well as completely written cs015.prj.LiteBriteSupport.LitePeg and

1 Note: a “skeleton” and “stencil” mean the same thing - we’ll be using the term “stencil” from now on.
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 LitePegs and ColorButtonss. 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 LiteBox and ColorPalette 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 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 LiteBox and ColorPalette 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.

   a. You can open all of the files in the LiteBrite directory by running atom * from your ~/course/cs015/LiteBrite directory -- the * means “all files” in the current directory.

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 and the code that you wrote in Lab 1 (the Cupcake lab). 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, LiteBox.java, ColorPalette.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, though you will need to edit it to run your program.

**Methods:**

```java
start(Stage stage)
```

Starts the application. This is equivalent to a constructor for the App class, but is a special method used to start a graphical application. You can instantiate your top-level class here as if it were the App constructor, and *you do not need to use the Stage parameter*.

---

**Name:**

**LiteBox**

**Purpose:**

This class models a lite box that can detect when a mouse has been clicked on top of it. It passes a `cs015.prj.LiteBriteSupport.LitePosition` as a parameter to the `insertLitePeg` method.

**Methods:**

```java
LiteBox(ColorPalette palette)
```

Constructs the grid (the lite box) with a reference to the instance of class `ColorPalette` indicated by the parameter `palette`. Note: Your top-level class should contain this and the `ColorPalette`, but it is not written for you; *you need to write this class yourself*. 
void insertLitePeg(cs015.prj.LiteBriteSupport.LitePosition position)
This method is called automatically when the mouse is clicked inside the grid. You do not ever need to call insertLitePeg(). If you want your grid to respond to a mouse click, you need to fill in this method.

Name:
ColorPalette

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

Methods:
ColorPalette()
Constructs an empty palette.

void setColor(javafx.scene.paint.Color newColor)
This method is called automatically when a ColorButton is clicked. You never need to call ColorPalette ’s setColor() method, just be sure to select a default color. However, to make your ColorPalette 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.ColorButton

Purpose:
This class models a color button on the ColorPalette which controls the color of the light pegs.

Methods:
```java
```
Constructs a color button with a reference to your ColorPalette.

Name:
```java
cs015.prj.LiteBriteSupport.LitePeg
```

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

Methods:
```java
cs015.prj.LiteBriteSupport.LitePeg()
```
Constructs the light peg.

```java
void setPosition(cs015.prj.LiteBriteSupport.LitePosition p)
```
This method moves the LitePeg to the position p on the LiteBox.

```java
void setColor(javafx.scene.paint.Color newColor)
```
This method sets the color of the light peg to be newColor. You're correctly using the ColorPalette when new LitePegs 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.

Minimum Functionality Requirements

MF Policy Summary: In order to pass CS15, you will have to meet minimum functionality requirements for all projects. If you don't meet them the first time around, you may hand the project in again until you succeed, but you will keep your original grade. MF requirements are not the same as the requirements for full credit on the project. You should attempt the full requirements on every project to keep pace with the course material. An 'A' project would meet all of the requirements on the handout and have good design and code style.
To meet MF for LiteBrite:

- The Lite Box and Color Palette appear on screen.
- Lite Pegs appear in the proper location on mouse clicks.
- The color of each new Lite Peg matches the color in the Color Palette.