Silly Premise

So here we are, the semester's just underway, and things are already getting out of hand. Students have begun harassing the TAs off hours. Last week, Shannon was found crying after an angry mob of students found her in the Ratty and forced her to go over polymorphism while stuffing her face with sweet potato fries. Asher was coding in the Sun Lab when a group of students approached brandishing wrist-support pads and chanting 'In-He-Ri-Tance'. Katya awoke in the middle of the night to find three students standing over her bed, subliminally sleep-hypnotizing her into giving them extra credit. Breaking point is nearing. There has to be somewhere the TAs can go to escape the angry mob of CS15ers…but where?

Your job is to code the TAs something sturdy and secluded—something safe—to protect them from the clutches of your bloodthirsty classmates. You will be responsible for constructing rooms in a TASafeHouse. The rooms will all contain some common elements, such as walls, ceilings, floors, lights, and light switches. Since you couldn't care less about creating quality rooms for
your beloved TAs (after all, what do they ever do for you???), you want to design rooms in the fastest, easiest way possible. Moreover, you’re working on a tight schedule (student insurgency is on the rise). If there’s a way to cut corners and avoid doing more work than you have to, you’re looking for it. This may require you to think about how to design the common elements of each room and how to implement each room’s unique features, but hey, that’s why you get paid the big bucks…and by bucks we mean Java knowledge.

It’s up to you to save your TAs by building them a house with quality – or at least aesthetically appealing – rooms.

As a wise man once said, “A wonderful home is like peeing on yourself: everyone can see it, but only you get the warm feeling that it brings.”

New Concepts Covered

- Polymorphism
- Inheritance
- Multiple instances of the same class
- Object communication

Assignment Specifications

Your assignment is to write a program that displays a TASafeHouse with nine rooms that match the individual specifications below. The house should have three instances of each of the three types of rooms, each located at a different position on the screen. Each room should have a light switch that the user can flick on or off with the mouse. When the light switch is clicked, all of the lights in that room should turn on if previously off, or turn off if previously on.

For TASafeHouse and all future programming assignments, we will also have a corresponding Design Questions (DQ) assignment. These questions will get you started on carefully planning the design of your program, and will greatly help the coding process. Please read this handout in full before starting the DQs.

All Rooms
Every room in your TASafeHouse must have:
- walls
- a floor
- a ceiling
- a light switch
- a ceiling light

Specific Rooms
Your TASafeHouse will contain 3 bedrooms, 3 bathrooms, and 3 kitchens.
Each bedroom must have:

<table>
<thead>
<tr>
<th>Furniture</th>
<th>Light Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>A bed</td>
<td>A table lamp</td>
</tr>
<tr>
<td>A set of drawers</td>
<td>A floor lamp</td>
</tr>
<tr>
<td>A bedside table</td>
<td>A ceiling light</td>
</tr>
</tbody>
</table>

Each bathroom must have:

<table>
<thead>
<tr>
<th>Furniture</th>
<th>Light Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>A tub</td>
<td>A mirror lamp</td>
</tr>
<tr>
<td>A toilet</td>
<td>A ceiling light</td>
</tr>
<tr>
<td>A mirror</td>
<td></td>
</tr>
<tr>
<td>A bathroom sink</td>
<td></td>
</tr>
</tbody>
</table>

Each kitchen must have:

<table>
<thead>
<tr>
<th>Furniture</th>
<th>Light Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>A ceiling fan</td>
<td>A ceiling light</td>
</tr>
<tr>
<td>A kitchen sink</td>
<td></td>
</tr>
<tr>
<td>A counter-top</td>
<td></td>
</tr>
<tr>
<td>A dishwasher</td>
<td></td>
</tr>
<tr>
<td>A stove</td>
<td></td>
</tr>
</tbody>
</table>

Each room should only contain the furnishings specific to that room—no more, no less.

**Do First**

- Read this handout carefully.
- Make sure you understand the DQs. Review lectures slides, help session, and/or go to hours if you don’t understand the DQs.
  - Specifically, make sure you understand the containment and inheritance relationships.
  - Think about what rooms have in common and what they don’t. Use inheritance and polymorphism to factor out whatever code you can into superclasses.
- Look at the demo. This should help you to get an idea of how your program should look and respond to user input in the form of mouse clicks.
- Write out pseudocode for the major methods that you will fill out.
- Keep in mind self-containment; try to avoid having objects know too much about each-other’s private implementations.
Programming Tips & Working with the Support Code

Positioning the Rooms
The constructor for `cs015.prj.TASafeHouseSupport.BasicRoom` receives a parameter of type `javafx.geometry.Point2D`. This parameter specifies where on the grid the top-left corner of the room is located. The room will correctly position itself on the screen based on this parameter. We provide you with the class `cs015.prj.TASafeHouseSupport.LocationGenerator` that can retrieve the nine screen locations for your rooms. This way, you do not need to know how to explicitly create an instance of the `javafx.geometry.Point2D` class (though if you are curious about how `javafx.geometry.Point2D` works, you can read the documentation available at [http://docs.oracle.com/javafx/2/api/javafx/geometry/Point2D.html](http://docs.oracle.com/javafx/2/api/javafx/geometry/Point2D.html)).

Adding Structural Components
Each room starts as a bare room. This is not what you want. You want it to have a floor, a ceiling, and walls. Look at the support code for `cs015.prj.TASafeHouseSupport.BasicRoom` to see if there are any methods that will do this for you.

Adding Furniture & Light Source(s)
All of the furniture and light source classes—beds, sinks, lamps, etc.—have been defined for you by the support code. The names of each of the furniture and light source subclasses you will use are listed in the support code section of this handout. Once created, you can get your furniture and light sources to show up in a room by using one of the methods provided by the `cs015.prj.TASafeHouseSupport.BasicRoom` support class.

Reacting to Mouse Clicks
In this program, just as in LiteBrite, you will need some way to detect when the user has clicked on a particular area of the screen. In LiteBrite we gave you a class that modeled a grid and a palette that could detect mouse clicks. For this program, the light switch will act similarly; if the user clicks the light switch, all the lights in the room should toggle between being on and off. A method will be called each time the light switch is either flicked on or flicked off (see the support code section for details).

Coding
Just as you designed this program in steps, you should code it in small parts. Write some code, get it to work, and then move on. First get a single room to appear in a specified location on the screen. Next, add the furniture and light sources that you want that room to contain. Then add a light switch to the room and make the room respond in some way when the light switch has been clicked. Now add another room that has a response to its light switch. Make sure you are not writing repetitive code. Your objects should be generic enough to make adding a different
room very easy. The rest of the program involves positioning three of each room in the TASafeHouse.

You might have to alter parts of the program’s design and make modifications to code you’ve already written as you work. However, the more time you spend on the design phase before you begin coding, the fewer changes you will have to make.

**Handin Info**

TASafeHouse is due on Thursday, Oct 8, at 11:59 PM. The early date is Tuesday, Oct 6, at 11:59 PM. The late date is Saturday, Oct 10, at 10:00 PM.

**Your program should be fully commented.** If you have questions about commenting your code, talk to a TA on hours or visit the Online TA (found on the course website).

You must handin this assignment electronically. To hand in TASafeHouse, type (in a shell):  
```shell
cs015_handin TASafeHouse
```

**Note:** 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.

See the Design Questions handout for information about DQ dates and handin information. As the deadline approaches, remember that more and more people will be in the Sun Lab and at TA hours, so make sure you start early and avoid the rush. Good luck!

**Extra Credit**

There is a way to factor out code so that the ceiling light is instantiated in only one place in all of your code. This will make one or more of your classes simpler and you will be awarded a few points of extra credit. Please refrain from using protected variables to achieve this. **DO NOT** work on extra credit until you have a working version of the project. You will not get credit for it unless your program is functional.

**Support Code**

**Class Name:** cs015.prj.TASafeHouseSupport.BasicRoom  
**Purpose:** This class models a room in a house. You do not want to use this class directly, instead, subclass it to perform the same desired behavior more generically (*hint hint* check out the inheritance and polymorphism lectures).
Constructor:

```java
public BasicRoom(javafx.geometry.Point2D location);
```

The constructor initializes the room and makes it appear in the frame based on the `javafx.geometry.Point2D` location that you pass it.

Methods:

```java
public void addLightSwitch(cs015.prj.TASafeHouseSupport.BasicLightSwitch lightSwitch);
```

Adds the `cs015.prj.TASafeHouseSupport.BasicLightSwitch lightSwitch` to the room's display.

```java
public void addFurniture(cs015.prj.TASafeHouseSupport.Furniture furnitureItem);
```

Adds the `cs015.prj.TASafeHouseSupport.Furniture furnitureItem` to the room's display.

```java
public void addLightSource(cs015.prj.TASafeHouseSupport.LightSource lightSource);
```

Adds the `cs015.prj.TASafeHouseSupport.LightSource lightSource` to the room's display.

```java
public void buildWalls();
```

Adds walls to the room and displays them.

```java
public void buildCeiling();
```

Adds a ceiling to the room and displays it.

```java
public void buildFloor();
```

Adds a floor to the room and displays it.

Class Name: `cs015.prj.TASafeHouseSupport.BasicLightSwitch`

Purpose:

This is an abstract class that models a light switch that can be added to a room. It has methods that are called when a user clicks on it. *These methods are abstract and must be implemented in a subclass of cs015.prj.TASafeHouseSupport.BasicLightSwitch!*

Constructor:

```java
public cs015.prj.TASafeHouseSupport.BasicLightSwitch(cs015.prj.TASafeHouseSupport.BasicRoom room);
```

The constructor initializes the light switch. Using the `addLightSwitch` method of `BasicRoom` will make it show up on the screen. You must implement this method in a subclass to have it do something useful!

Methods:

```java
public abstract void flickedOn();
```
This method is called when the user clicks a light switch, turning it on. You must implement this method in a subclass to have it do something useful!

public abstract void flickedOff();
This method is called when the user clicks a light switch, turning it off. You must implement this method in a subclass to have it do something useful!

Class Name: cs015.prj.TASafeHouseSupport.LocationGenerator

Purpose:
This class provides the nine positions for your rooms and lights. You do not need to extend from this class.

Constructor:
public cs015.prj.TASafeHouseSupport.LocationGenerator();
The constructor creates the generator.

Methods:
The following methods provide ways to return an individual javafx.geometry.Point2D, which represents a point on the screen.

public javafx.geometry.Point2D getTopLeft();
public javafx.geometry.Point2D getTopCenter();
public javafx.geometry.Point2D getTopRight();
public javafx.geometry.Point2D getLeft();
public javafx.geometry.Point2D getCenterLeft();
public javafx.geometry.Point2D getCenterRight();
public javafx.geometry.Point2D getBottomLeft();
public javafx.geometry.Point2D getBottomCenter();
public javafx.geometry.Point2D getBottomRight();

Once you have instantiated a LocationGenerator, it can be used multiple times to get a location by calling the preceding methods. Each of these methods returns the corresponding javafx.geometry.Point2D. Think of the screen as a tic-tac-toe board. If you call one of these methods and pass the position returned to the constructor of the room, it will position itself around that point.

Furniture Classes
All of the classes listed below are subclasses of cs015.prj.TASafeHouseSupport.Furniture and have no methods other than a constructor.

Bedroom:
cs015.prj.TASafeHouseSupport.Bed
cs015.prj.TASafeHouseSupport.Drawers
cs015.prj.TASafeHouseSupport.BedsideTable

Bathroom:
cs015.prj.TASafeHouseSupport.Tub
cs015.prj.TASafeHouseSupport.Toilet
cs015.prj.TASafeHouseSupport.BathroomSink
LightSource Classes
All of the classes listed below are subclasses of cs015.prj.TASafeHouseSupport.LightSource. In addition to constructors, these subclasses each have a showIlluminated() method that will display the light in the illuminated state and a showDarkened() method that will display the light in the darkened state.

All Rooms:
- cs015.prj.TASafeHouseSupport.CeilingLight

Bedroom:
- cs015.prj.TASafeHouseSupport.TableLamp
- cs015.prj.TASafeHouseSupport.FloorLamp

Bathroom:
- cs015.prj.TASafeHouseSupport.MirrorLamp

Methods:
- public void showIlluminated();
- public void showDarkened();