**TASafeHouse**

**Help Slides Released:** Monday, October 3, 2pm

**Design Discussion** [Mini-Assignment Due: Monday, October 3, 2pm](#)

**Design Discussions:** Monday, October 3 - Wednesday, October 5

**Design Discussion Signups:** You must sign up for your design discussion time by Saturday, 10/1 at 11:59pm. Sign up following the instructions [here](#).

(Deadline times below updated on 10/3/16 at 11 PM)

**Early Due Date:** Friday, October 7, 10pm

**On-time Due Date:** Sunday, October 9, 11:59pm

*Please note that there is no late handin for this project, so the last opportunity to hand in for credit is the ontime deadline.*

To run demo: `cs015_runDemo TASafeHouse`

*Demos may not work well over ssh! Try FastX or the Sunlab.*

To install: `cs015_install TASafeHouse`

To hand in: `cs015_handin TASafeHouse`

---

**Silly Premise**

So here we are, the semester’s just underway, and the Grinch is up to their old tricks. CS15 TAs have been disappearing from all over Brownville, and the Grinch is to blame. A witness reported that Montana was taken from the SunLab “so quickly she didn’t even log out”. After failing to show up to her lab, an investigation discovered scratch marks on the inside of the Fishbowl, indicating that the Grinch had trapped Sam after her hours. Chantal was last seen walking into the Vdub after receiving a strange text about a new “Chicken Finger Saturday” - authorities later found her computer ominously circling around the dish conveyor belt. Tristin awoke in the middle of the night to find the Grinch standing over him humming the Jaws theme song. There has to be somewhere the TAs can go to escape the Grinch…but where?

Your job is to code the TAs something sturdy and secluded—something *safe*—to protect them from the clutches of the Grinch. 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 all 47 TAs need a SafeHouse, you want to design rooms in the fastest, easiest way possible. Moreover, you’re working on a tight schedule (the Grinch works fast). 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.”

**Collaboration Policy Reminder**

From the collaboration policy:

Collaboration on project design is not allowed, except for during the design discussion for this project under the supervision of your discussion leader TAs. Otherwise, no collaboration is allowed on project-specific details. You are not allowed to discuss the classes you will be using in your project, what methods you will be writing, inheritance hierarchies, the design discussion mini-assignment, or any other design components of the program. You may not discuss the implementation and debugging of code for projects with anyone except the course staff.

**Important**, 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 inheritance pertain to a specific assignment.

**New Concepts Covered**

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

**Mini Assignment**

For TASafeHouse and all future programming assignments, we will also have a corresponding Design Discussion and mini-assignment. The mini-assignment questions will get you started on carefully planning the design of your program, and will greatly help with the coding process. You will review and discuss your design in your Design Discussion. Make sure to read this handout in full before starting the TASafeHouse mini-assignment and make sure to sign up for your discussions by Saturday, October 1 at 11:59pm following the instructions linked to at the top of the handout.

**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.

**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 design discussion mini-assignment. Review lectures slides, help slides (when available), and/or go to hours if you have questions about the mini-assignment or project design.
  - 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.
    - Note: In many situations, interfaces are a great solution for generic programming. However, this assignment is specifically focused on inheritance and polymorphism.
- 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.

Coding Incrementally

Just as you designed this program in steps, you should code it in small parts. Write some code, get it to compile and run correctly, and then move on.

Here are some suggested steps for coding incrementally:

1. Get one bare room to show up in the correct location. Add walls, a ceiling and a floor then fill that room with the correct furniture.
2. Get all 9 rooms to show up in their locations with the correct furniture.
3. Get the lights to turn off and on in one room (start with a Kitchen!).
4. Get the lights to work in all three types of room.

Make sure you are not writing repetitive code! Your objects should be generic enough to make adding a different room very easy.

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.

Programming Tips & Working with the Support Code

Positioning the Rooms

The constructor for cs015.prj.TASafeHouseSupport.GraphicalRoom 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 \texttt{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 \texttt{javafx.geometry.Point2D} class (though if you are curious about how \texttt{javafx.geometry.Point2D} works, you can read the documentation available at \url{http://docs.oracle.com/javafx/2/api/javafx/geometry/Point2D.html}).

\section*{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 documentation for \texttt{cs015.prj.TASafeHouseSupport.GraphicalRoom} to see if there are any methods that will do this for you.

\section*{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 \texttt{cs015.prj.TASafeHouseSupport.GraphicalRoom} support class.

\section*{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 that room should toggle between being on and off. A method will be called each time the user flicks a light switch on or off (see the support code section for details).

\section*{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. \textbf{DO NOT} work on extra credit until you have a working version of the project, as you will not get credit for it unless your program is functional.

\section*{Running Your Code}
To run the program you must first \texttt{cd} (change directory) into \texttt{~/course/cs015/TASafeHouse} directory, then compile as you normally would by running \texttt{javac *.java} in your terminal. Run your program by typing \texttt{java TASafeHouse.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 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 on-time deadline at the top of this handout, 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. If you are not finished by the late deadline, it is always better to hand something in for partial credit than turn in nothing at all.

Stencil Class

Below is a listing of the stencil class 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 TASafeHouse project, this class will already be in your TASafeHouse directory, though you will need to edit it to run your program.

Methods:

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.*
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:

\texttt{cs015.prj.TASafeHouseSupport.GraphicalRoom}

Purpose:

This is an \texttt{abstract class} that models the graphics for an empty room in a house. You will not want to instantiate this class directly; instead, extend from it to add the desired functionality (*hint hint* check out the inheritance and polymorphism lecture).

Methods:

\texttt{GraphicalRoom(javafx.geometry.Point2D \texttt{location})}

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

\texttt{void \texttt{addLightSwitch} (cs015.prj.TASafeHouseSupport.GraphicalLightSwitch \texttt{lightSwitch})}

Adds a \texttt{GraphicalLightSwitch object to the room’s display.}

\texttt{void \texttt{addFurniture} (cs015.prj.TASafeHouseSupport.Furniture \texttt{furnitureItem})}

Adds the \texttt{Furniture object to the room’s display.}

\texttt{void \texttt{addLightSource} (cs015.prj.TASafeHouseSupport.LightSource \texttt{lightSource})}

Adds the \texttt{LightSource object to the room’s display.}

\texttt{void \texttt{buildWalls} ()}

Adds walls to the room and displays them.

\texttt{void \texttt{buildCeiling} ()}

Adds a ceiling to the room and displays it.

\texttt{void \texttt{buildFloor} ()}

Adds a floor to the room and displays it.
Name:

cs015.prj.TASafeHouseSupport.GraphicalLightSwitch

Purpose:

This is an abstract class that models the graphics for 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 GraphicalLightSwitch.

Methods:

GraphicalLightSwitch(cs015.prj.TASafeHouseSupport.GraphicalRoom room)
The constructor initializes the light switch. Passing an instance of the light switch to the addLightSwitch method of a GraphicalRoom object will make it show up on the screen in that room. You must create a constructor for a subclass of GraphicalLightSwitch for the light switch to do something useful!

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

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

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 - you can instantiate one directly.

Methods:

LocationGenerator()
The constructor creates the generator.

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

   javafx.geometry.Point2D getTopLeft()
   javafx.geometry.Point2D getTopCenter()
   javafx.geometry.Point2D getTopRight()
   javafx.geometry.Point2D getCenterLeft()
   javafx.geometry.Point2D getCenterCenter()
   javafx.geometry.Point2D getCenterRight()
   javafx.geometry.Point2D getBottomLeft()
   javafx.geometry.Point2D getBottomCenter()
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
- cs015.prj.TASafeHouseSupport.Mirror

**Kitchen:**
- cs015.prj.TASafeHouseSupport.KitchenSink
- cs015.prj.TASafeHouseSupport.Countertop
- cs015.prj.TASafeHouseSupport.CeilingFan
- cs015.prj.TASafeHouseSupport.Dishwasher
- cs015.prj.TASafeHouseSupport.Stove

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:
void showIlluminated()
void showDarkened()

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 enumerated in the assignment specification section of the handout and have good design and code style.*

To meet minimum functionality for TASAfeHouse:

- The TASAfeHouse appears on screen with 3 Bedrooms, 3 Bathrooms, and 3 Kitchens
- Each room contains a CeilingLight
- The user is able to switch all of the lights on/off in at least one type of room