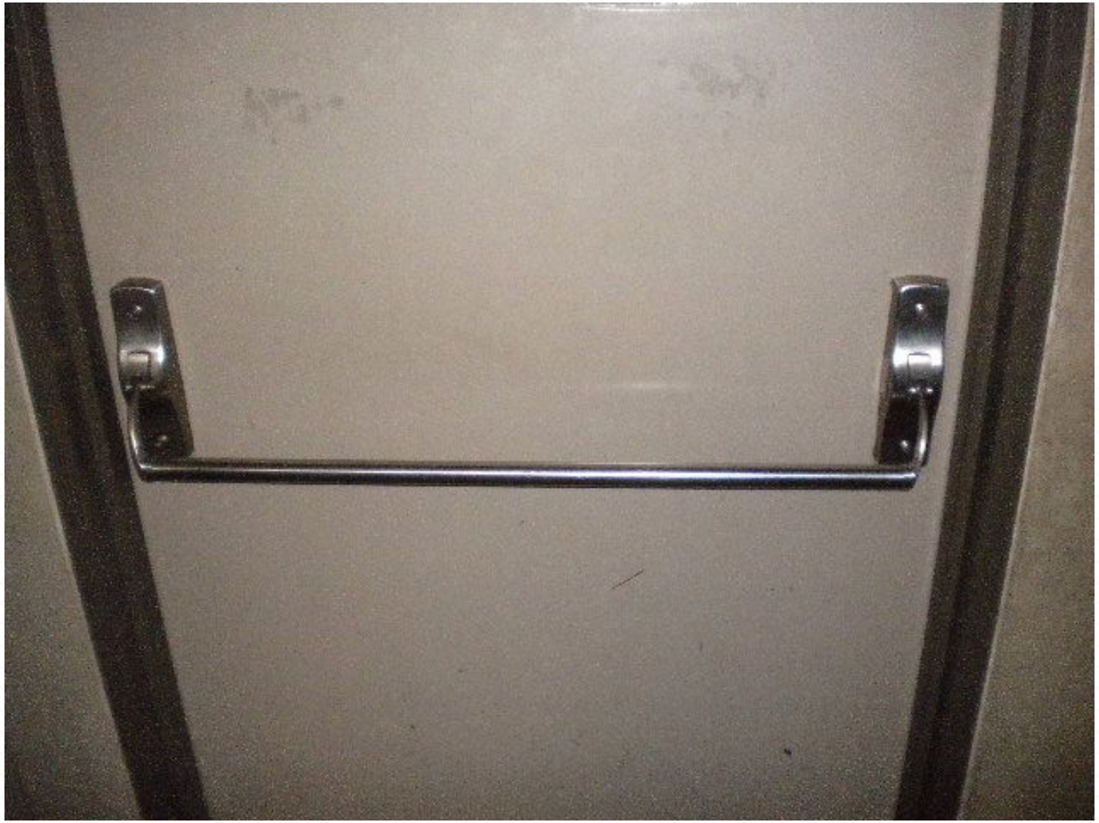
CS1114
Intro to Software Design
Michael Irwin - Fall 2019

Events/Reminders
HW #0 “due” tonight
Lab 1 this week
Reading Quiz 1 due Sunday night
Last day to add courses is 8/30

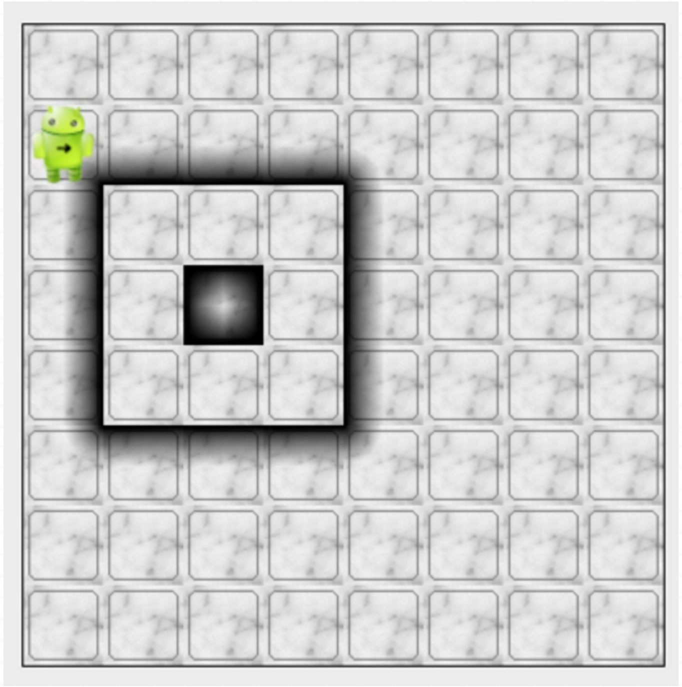
VIRGINIA TECH.

Why do we need good design?

VIRGINIA TECH™



Our scenario for the day...



The quick/simple way

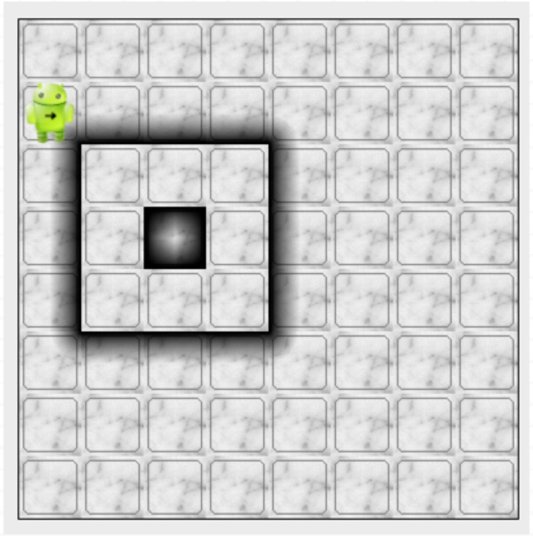
- We could simply tell the bot to move four times, turn right, move four times, turn right...

```
andy.move();
andy.move();
andy.move();
andy.move();
andy.turnRight();

andy.move();
andy.move();
andy.move();
andy.move();
andy.turnRight();
```

```
andy.move();
andy.move();
andy.move();
andy.move();
andy.turnRight();

andy.move();
andy.move();
andy.move();
andy.move();
andy.turnRight();
```



What smells does that code have?

VIRGINIA TECH™

Introducing class inheritance

- Could we make our bot smarter, so he could patrol on his own?
- Two problems with doing that…
    - Not all LightBots will patrol (cohesion)
    - We don't have the source for `LightBot` anyways :(

```java
public class NewClass extends ParentClass {
 // New stuff goes here
}
```

- The `NewClass` will inherit all methods from `ParentClass`

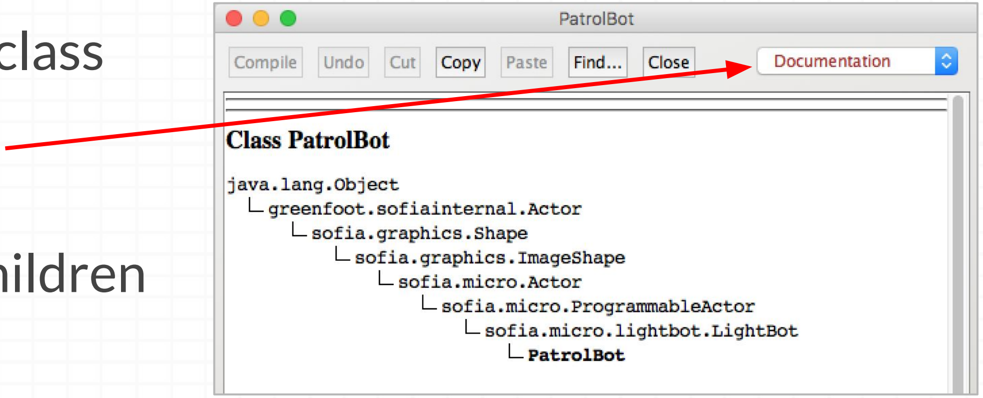
Making our PatrolBot

```
public class PatrolBot extends LightBot {
 // New stuff goes here
}
```

- `PatrolBot` inherits all methods of `LightBot`, but can have its own methods
    - Example... it has methods named `move()`, `turnRight()`, `turnLeft()`
    - We can add our own `patrolCastle()` method
- `PatrolBot` is the child or subclass of `LightBot`
- `LightBot` is the parent or superclass of `PatrolBot`

Parent vs Child

- You can view the JavaDoc for a class by switching the editor into **Documentation** mode.
- Parents are listed at top, with children branching underneath
- Every class in Java extends from Object (don't need to explicitly extend it)



Class PatrolBot

```
java.lang.Object
  └ greenfoot.sofiainternal.Actor
      └ sofia.graphics.Shape
          └ sofia.graphics.ImageShape
              └ sofia.micro.Actor
                  └ sofia.micro.ProgrammableActor
                      └ sofia.micro.lightbot.LightBot
                          └ PatrolBot
```

Creating new methods

```java
public ReturnType methodName() {
  // Method behavior goes here
}
```

```java
public void patrolCastle() {
  // Method behavior goes here
}
```

- The `public` access modifier indicates anyone can call the method
  - We'll talk about other access modifiers later in the semester
- The `ReturnType` indicates what will be returned
  - If nothing will be returned, use `void`
- The method's name should reflect what it will do
- Don't forget to document your new methods

Updating PatrolBot

- Add the `patrolCastle()` method to the PatrolBot and use code we wrote earlier
- Only change... it's no longer `bot.move()`, but simply `move()`. Why??

```java
public void patrolCastle() {
  move();
  move();
  move();
  move();
  turnRight();

  move();
  move();
  move();
  move();
  turnRight();

  // Two more times
}
```

What smells does that code have?

VIRGINIA TECH™

Making it cleaner...

- Each repetition is walking one wall
- Let’s pull that into its own method, named `walkOneWall`
- Update the `patrolCastle` to invoke that method four times

```java
public class PatrolBot extends LightBot {

  /**
   * Patrol around the castle
   */
  public void patrolCastle() {
    walkOneWall();
    walkOneWall();
    walkOneWall();
    walkOneWall();
  }

  /**
   * Walk along a single wall
   */
  public void walkOneWall() {
    move();
    move();
    move();
    move();
    turnRight();
  }
}
```

VIRGINIA TECH

We now have a nice, clean PatrolBot!

VIRGINIA TECH™