

# Unit 4

# The LibGDX Game Development Framework (Android Focus)

**Android Focus:** The purpose of this unit is to create Android apps using the LibGDX graphics framework. The activities focus on Android development and using game development widgets to create and modify a graphics-based game.

---

## Lesson 4.1: Creating a New World

### Preface

Lesson 4.1 introduces you to the Emu on the Loose! game. You will learn the LibGDX game development framework and use two-dimensional arrays to manipulate your game world.

### Essential Questions

1. How can developers take advantage of Object-Oriented Programming?
2. What factors should you consider when determining which data structure to use for a specific situation?
3. What makes for a good process for software development?
4. How and where is abstraction most useful and least useful?

---

## Lesson 4.2: Graphic Adventure Game

### Preface

In this lesson, you will examine the updated source code, correct a bug, and then add new LibGDX features to the game.

## Essential Questions

1. How can developers take advantage of Object-Oriented Programming?
2. How and when should inheritance be used to design object-oriented software?
3. How and when should interfaces be used to design object-oriented software?
4. What makes for a good process for software development?
5. How and where is abstraction most useful and least useful?

---

## Lesson 4.3: Independent Projects

### Preface

For your final project, you will choose what you want to develop. You can do one of the following:

- Create an app to solve a problem.
- Create an app to tell a story.
- Modify a game app.

You will use the software development cycle you have been practicing throughout the course and then deliver a final presentation of your work.

## Essential Questions

1. What factors should you consider when determining which data structure to use for a specific situation?
2. What makes for a good process for software development?
3. How and where is abstraction most useful and least useful?
4. What makes for a successful piece of software?

# Lesson 4.1: Creating a New World

## ACTIVITY 4.1.1

# LibGDX Setup

### INTRODUCTION

LibGDX is a **game development framework** that can be used to create games that can deploy across several different platforms including Android™ devices. The primary benefit of using a game development framework like LibGDX is the level of **abstraction** that it provides. This particular framework consists of several libraries that make the game development process significantly easier by handling many of the low-level details associated with games, such as rendering graphics.

In this lesson, you will further refine your Java skill set within the new context of game development by implementing the necessary classes to create the game, Emu On The Loose. In this game, the main character, Angelina, attempts to track down an Emu that is hiding in her school.

#### game development framework

A set of libraries that provides classes and methods, which implement functionality that is commonly used in creating games.

#### abstraction

Hiding data or details.

### Materials

- Computer with Android Studio™
- Android™ tablet and USB cable, or device emulator

## Procedure

### Part I: Set Up LibGDX

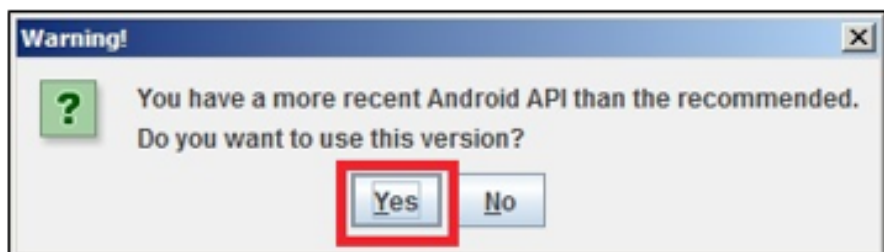
- 1 Get a copy of the *LibGDX* resource folder as instructed by your teacher.
- 2 Open the *LibGDX* folder and launch the *gdx-setup.jar* file by double-clicking on it.

- 3 In the LibGDX Project Setup window, modify the setup details as shown in the following figure and table.

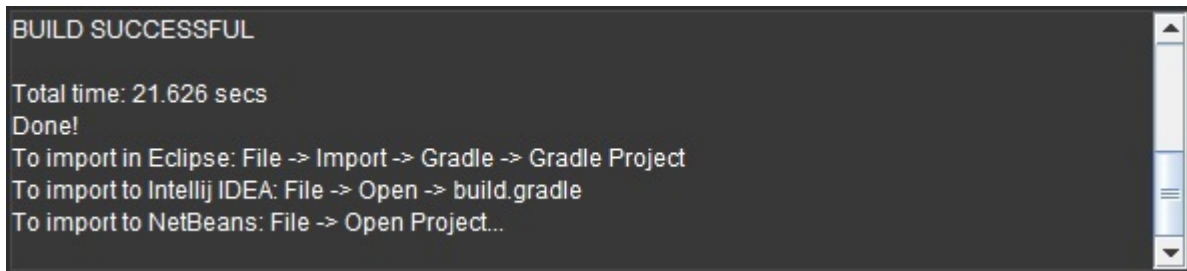


Name	emu-on-the-loose
Package	org.pltw.examples
Game class	EmuOnTheLoose
Destination	<your 411_EmuOnTheLoose path>
Android SDK	<your Android SDK path>
iOS	unchecked

- 4 After you have modified the settings, click **Generate**.
- 5 If you encounter the warnings shown below, click **Yes** for each.



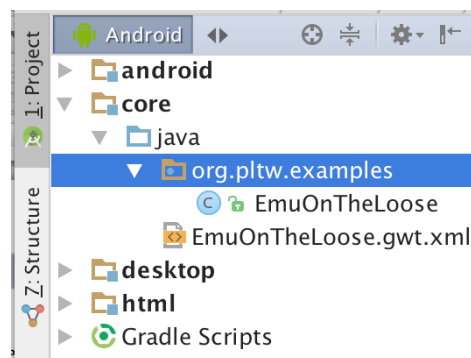
The build should take only a few moments. When the LibGDX project setup log window displays the message “BUILD SUCCESSFUL”, the process is complete.



The window also shows instructions for importing the project into a variety of other environments. Since you are working with Android Studio, you can dismiss the LibGDX Project Generator by clicking the X in the upper-right corner of the window.

## Part II: Import Project

- 6 Open Android Studio.
- 7 In the Welcome window, select **Import Project (Eclipse ADT, Gradle, etc)**.
- 8 Browse to the location that you entered for *<your 411\_EmuOnTheLoose path>* in the LibGDX setup window.
- 9 Select the *build.gradle* file and click **OK**.
- 10 If you are prompted to upgrade the gradle plugin, select **OK**.
- 11 If you are prompted to choose a version of the Android SDK, select **Use Android Studio's SDK**.
- 12 After a few moments, you may be prompted to reload the project so that language level changes can take effect. Accept these changes.
- 13 After gradle has finished building the project, click **Project > core** to expand your project file structure to see the contents of *org.pltw.examples*, as shown below.



- 14 Obtain a copy of *411\_EmuOnTheLoose\_Starter* files from your teacher and save them in a new folder named *411\_EmuOnTheLoose*.
- 15 Copy the *game* and *util* folders into *org.pltw.examples*.

#### NOTES

- The easiest way to copy the folders is to drag the folders onto the file structure in Android Studio.
- Make sure that if you change your package name, you check each file for consistency.

- 16 Open *EmuOnTheLoose.java* in your project structure in Android Studio.
- 17 From your *411\_EmuOnTheLoose* folder, copy the contents of the file *EmuOnTheLoose.java* and paste into the Android Studio project's blank *EmuOnTheLoose.java* file that you just opened.
- 18 Expand your project file structure under *android* to see the contents of *assets*. From your *411\_EmuOnTheLoose* folder, copy the following files into the *assets* directory:
  - *EmuOnTheLoose.atlas*
  - *EmuOnTheLoose.png*
- 19 Delete the file *badlogic.jpg* from the *assets* directory in the project directory structure.
- 20 Open *EmuOnTheLoose.atlas* and *EmuOnTheLoose.png*.

How do you think these two files relate?

- 21 Review the source code in the *org.pltw.examples* directory, except for *WorldModel.java* and the *renderTestObjects* method of *WorldView.java*. These will be discussed later.
- 22 Add comments to the source code for *Assets.java*, *WorldView.java* (except for the *renderTestObjects* method), and *EmuOnTheLoose.java* so that someone seeing the project for the first time could make sense of the code. Use the Internet to search for help on things that you need more information on.

## Part III: Create Texture Atlases (Optional)

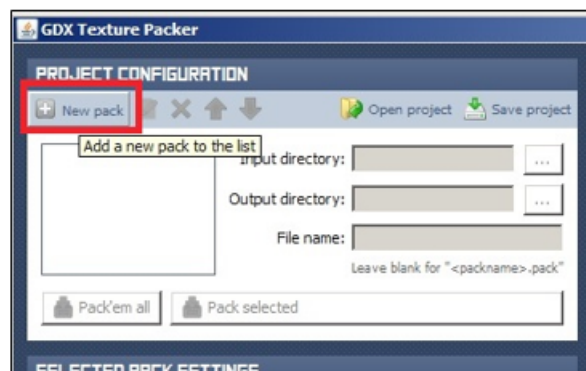
In this part of the activity, you will learn how to create your own **texture atlases**. This will be useful if you decide later that you'd like to use custom images in a game.

The assets files that you examined in the previous part of this activity were both created by a texture packer.

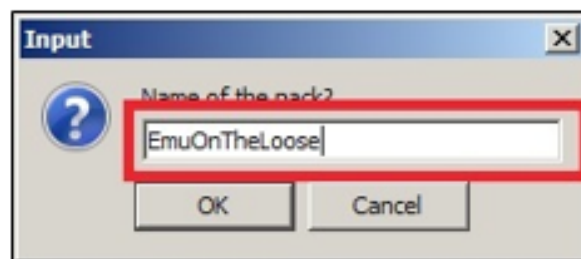
### texture atlas

Data for a set of images that have been especially configured for use on in-game objects.

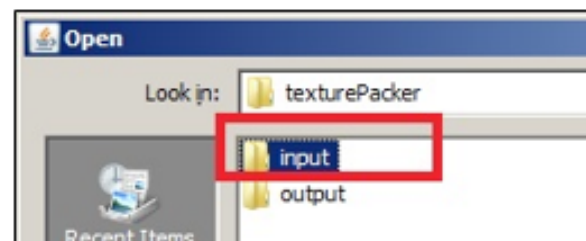
- 23 Go to your `411_EmuOnTheLoose` folder, and run the `gdx-texturepacker` file.
- 24 In the **PROJECT CONFIGURATION** panel, select **New pack**.



- 25 In the Input dialog box, name the pack “EmuOnTheLoose” and click **OK**.



- 26 For the input directory, select `texturePacker/input`. (This folder of images resides in the same place as the `gdx-texturepacker` utility that you are running.)



- 27 For the output directory, select *texturePacker/output*.
- 28 Increase the values of **Max page width** and **Max page height** to **4,096**, a power of two that will be large enough to fit all the sprites (images) in one file.
- 29 To create a texture atlas and sprite sheet using all the images in the input folder, click **Pack'em all**.
- 30 Retrieve the results from the *output* folder. The *.atlas* file is the texture atlas, and the *image* file is the sprite sheet.

## Part IV: Create 2D Arrays

---

- 31 Review the following slides.

### Two-dimensional (2D) Arrays

- Arrays that can store data in row-column organization
- Natural fit for data such as spreadsheets, game boards, classroom seats, etc.

### 2D Arrays (continued)

- **Declare** and **initialize** similar to a one-dimensional array with additional construct for the 2<sup>nd</sup> dimension

```
public String[][] board = new String[3][3];
```

- **Access** using row major ordering
  - row first, then column
  - provide an index to both dimensions

```
board[i][j] = "";
```



## 2D Arrays (continued)

In Java, 2D arrays are stored as a one-dimensional array where each element contains *another* one-dimensional array

```
public String[][] board = new String[3][3];
```

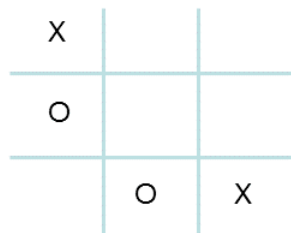
Represented as:



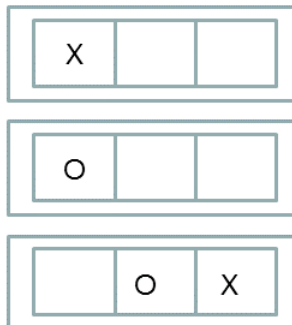
Computer Science A

© 2016 Project Lead The Way, Inc.

## A Tic-tac-toe board



```
public String[][] board = new String[3][3];
```



```
board[0][0]="X"; // row 0,col 0
```

```
board[1][0]="O"; // row 1,col 0
```

```
board[2][2]="X"; // row 2,col 2  
board[2][1]="O"; // row 2,col 1
```

Computer Science A

© 2016 Project Lead The Way, Inc.

## A game board

For a game board, the 2D array can store sprites.

```
public Sprite[][] sampleMap =  
    new Sprite[5][5];  
  
sampleMap[i][j] = tempSprite;
```

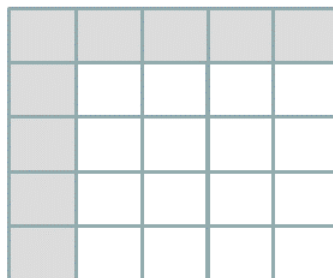
Computer Science A

© 2016 Project Lead The Way, Inc.

## 2D Arrays in LibGDX

- The walls in the room occupy all of row 0 and all of column 0 of a 2D array
- For example, the gray areas are walls in a 5x5 array

```
public Sprite[][] sampleMap = new Sprite[5][5];
```



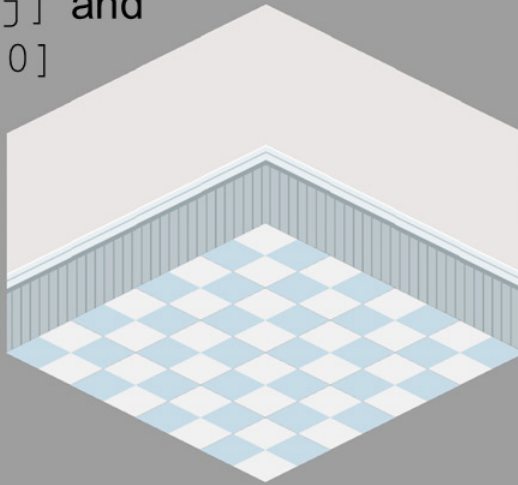
Computer Science A

© 2016 Project Lead The Way, Inc.

The code in your project creates a simple grid of floor and wall tiles. Within the model layer of the app, the wall tiles are represented in row 0 and column 0.

## 2D Arrays

The walls in the room occupy  
`sampleMap[0][j]` and  
`sampleMap[i][0]`

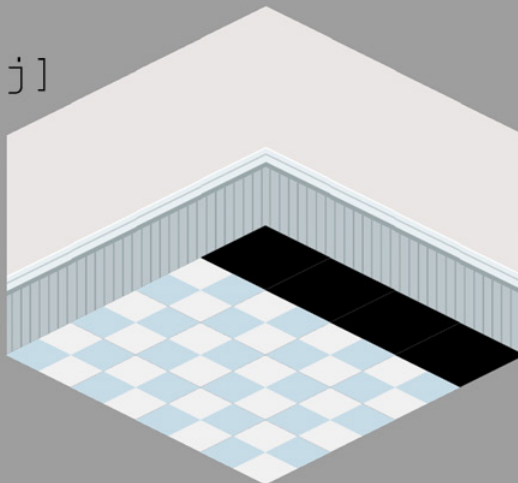


Computer Science A

© 2016 Project Lead The Way, Inc.

## 2D Arrays

The floor tiles in black are in row 1,  
accessed as  
`sampleMap[1][j]`



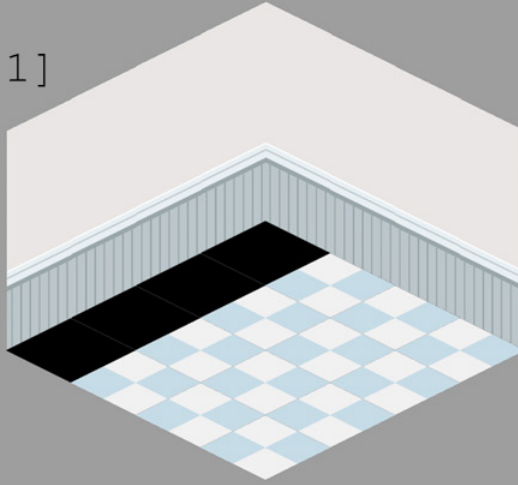
Computer Science A

© 2016 Project Lead The Way, Inc.

## 2D Arrays

The floor tiles in black are in column 1,  
accessed as

```
sampleMap[i][1]
```

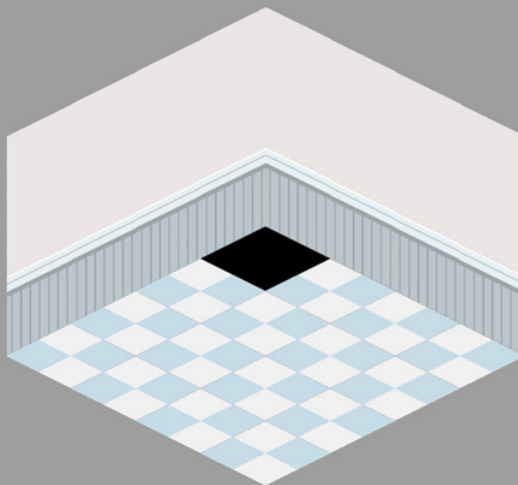


Computer Science A

© 2016 Project Lead The Way, Inc.

## 2D Arrays

What are the indices of the floor tiles in black?



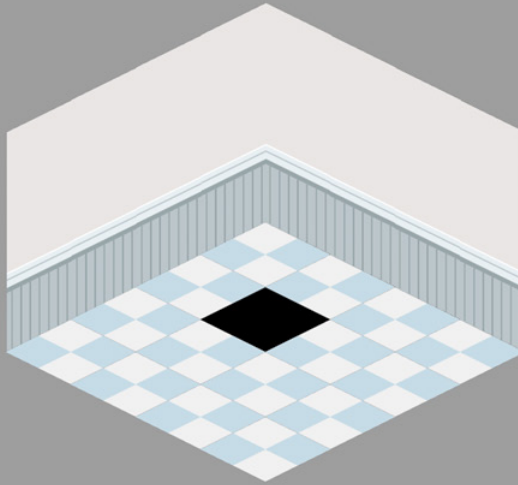
Computer Science A

© 2016 Project Lead The Way, Inc.

These tiles would be indexed by: `sampleMap[1][1]`

## 2D Arrays

What are the indices of the floor tiles in black?



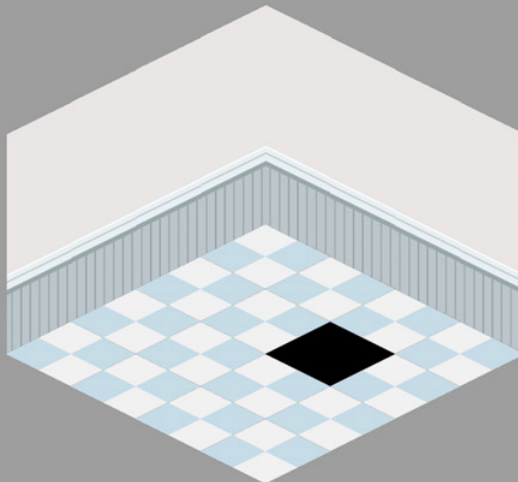
Computer Science A

© 2016 Project Lead The Way, Inc.

These tiles would be indexed by: `sampleMap[2][2]`

## 2D Arrays

What are the indices of the floor tiles in black?



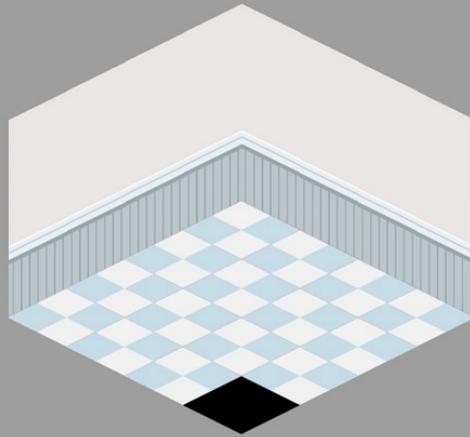
Computer Science A

© 2016 Project Lead The Way, Inc.

These tiles would be indexed by: `sampleMap[2][3]`

## 2D Arrays


What are the indices of the floor tiles in black?

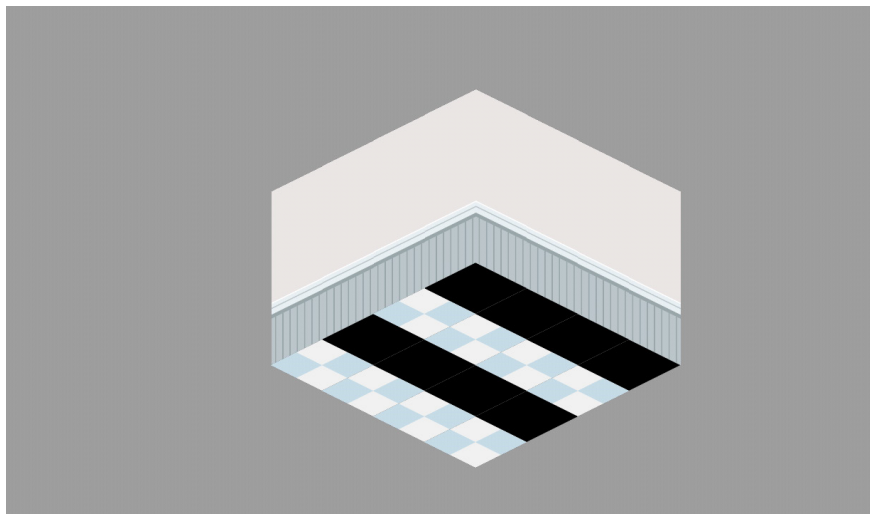


Computer Science A

© 2016 Project Lead The Way, Inc.

These tiles would be indexed by: `sampleMap[4][4]`

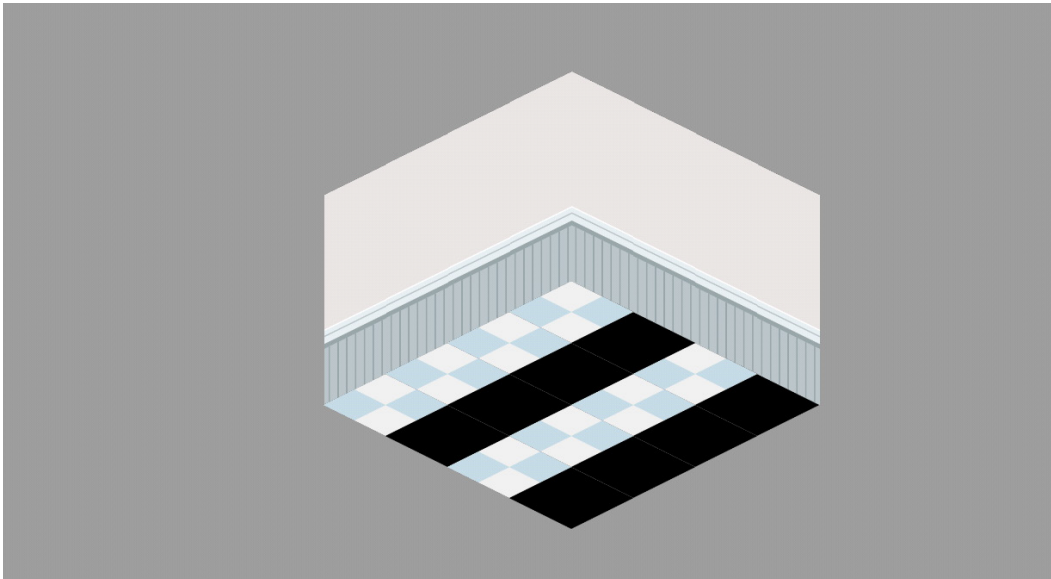
- 32 Review the content  [Introduction to 2D Arrays](#) and answer the **Check your understanding** questions.
- 33 In `Assets.java`, add `tileWood` to `AssetRoomTiles`, following the pattern that you find in that class.
- 34 Add methods in `WorldModel` for setting up black floor tiles and wood floor tiles. These will be very similar to the `blueFloorTile()` method.
- 35 Under the TODO comment in `WorldModel`, replace the call to `blueFloorTile()` with a conditional sequence that calls `blueFloorTile()` and `blackFloorTile()` so that the room rendered looks like the one below.



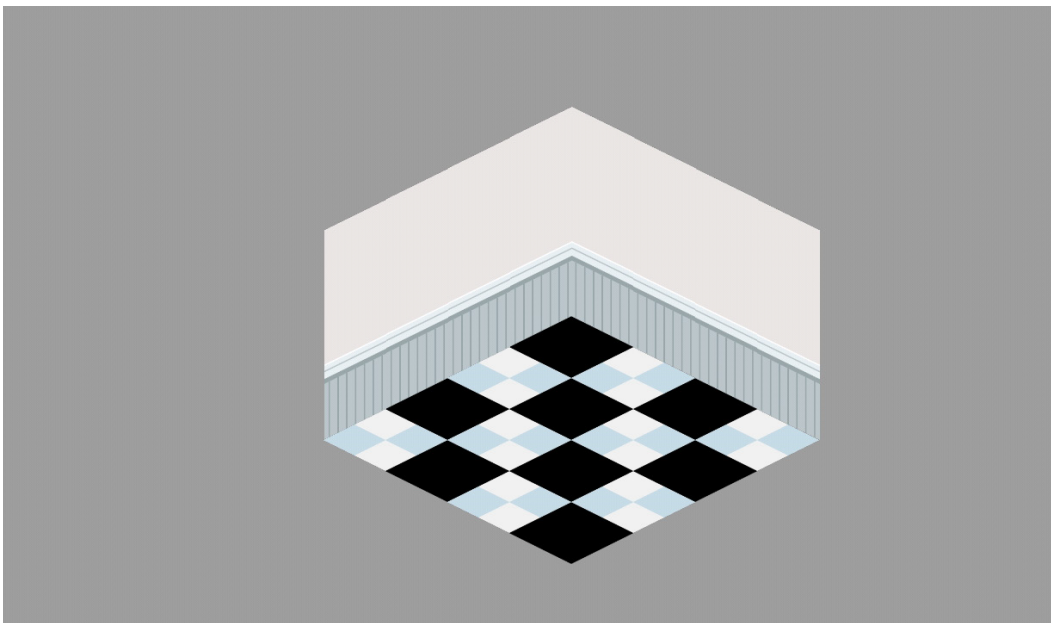
- 36 Create a method `stripeFloor(int i, int j)` that abstracts your work from the previous step.

As you write code to complete the following steps, create methods in `WorldModel` so that your instructor can clearly see the work you did for each step. Give the methods names that indicate their functionality.

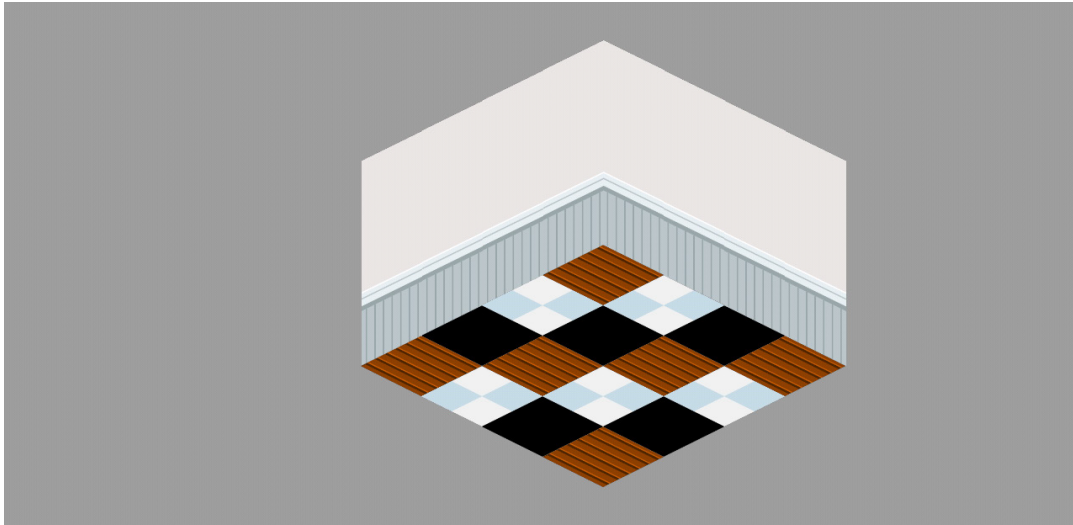
- 37 Write a method to create the following configuration of floor tiles in the room:



- 38 Write a method to create the following configuration of floor tiles in the room:



- 39 To create the following configuration of floor tiles in the room, write code using only three conditions. Modify as many classes as necessary.



## Part V: Sharing Games, Sharing Risks



The gaming industry is a multibillion-dollar industry, and people worldwide share online games and play games together. People play in their homes, at coffee shops, libraries, even at work (although playing at work will likely get a person fired). Sharing games over public, wireless networks can introduce risks to your computer and your data. Sharing games or game data on USB drives can spread viruses or **malware**.

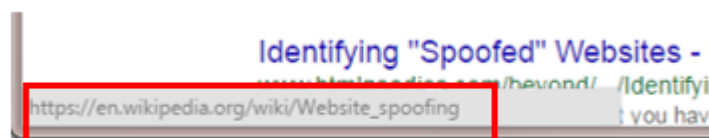
### malware

Friendly- or useful-looking software that is actually harmful to you or your computer, your phone, mobile device, etc.

- 40 Search the web to find ways to protect your data. Observe the URLs carefully; only view a site that you think is trustworthy.

The following may help you determine trustworthy sites.

- Beware of companies trying to sell you something; the first sites listed are often advertisements for a product. Also, an “Ad” icon next to a URL indicates a paid advertisement.
- Watch out for  **website spoofing** sites or  **spoofed URLs** that look official but are really fake or malicious. For example, a few years ago, a visit to **gogggle.com** (notice the misspelled double-g) would automatically download computer viruses to your computer. Check the spelling of the target website carefully. The URL of the target website is available in the status bar at the bottom of your browser:





- A `.com` extension in a URL represents a business. A cybersecurity business may have a different purpose than to inform you of *all* ways to protect your data; they may want to sell you their products. But keep in mind that the *New York Times* and the *San Francisco Chronicle* are businesses, too, and might provide valuable articles about cybersecurity.
- A `.net` extension is usually a business, but may represent a non-profit organization.
- The `.org` extension indicates a non-profit organization that is not in business to make money, such as The American Cancer Society. These sites are often good sources of information; however, they may be difficult to find, since they often do not pay for services that would make them appear in search results. As an interesting experiment, search “cyber security non-profit organizations” and notice how many `.com` sites are returned as opposed to `.orgs`.
- An `.edu` extension represents an educational institution. Most colleges and universities use this extension. Larger universities conduct research on a wide variety of topics, so their websites often hold useful resources.
- The extension `.gov` represents a government agency, either at the state or Federal level. Often, government sites (both American and international) provide public services in the form of information and advice.

**41** Share your research with another team, and work together to come up with a list of five ways to protect your data.

## CONCLUSION

1. Give an example of an app that you think might use a 2D array.