DUE DATE: FEB 21TH, 2020 AT 11:59PM

Instructions:

- Read the **`SubmissionProcedures.PDF`** and the **`Standards.java`** carefully BEFORE you start.
- Let me know about any typos, clarifications or questions you may have right away. Don't wait.
- Read the whole assignment before you start.
- Start right away. The sooner you get stuck, the sooner you can get help to get un-stuck.

Assignment overview

Learning outcomes:

- Become more comfortable editing existing code
- Understand how to manipulate 2D arrays
- Understand how to draw simple graphics using StdDraw.
- Learn the procedure to load simple data from files in Java.

Additional Notes:

Read **Standards.java** and **SubmissionGuidelines.PDF** Carefully & before you start.

Several Classes will be provided as a starting point. You can either continue from your own solution (I will give a couple of bonus marks for this if you want to adapt your current code to work) or start from the handout version. Note, I have made some additional changes to this code base. If using my files, create a new .**git** repository for this project. I will not accept projects handed in without the git repo. I also am not prompting you to commit your code, you should be able to do this frequently on your own. Usually the best time is when you get something new running, or after you fix each bug you encounter (fully fixed and tested code with notes).

Please provide detailed commenting and commit your code often!!!

Assignment Notes

Be sure to read the assignment carefully. Include a **`readme.txt`** document that specifies the latest commit that will compile and run. I will **NOT** mark your assignment without the **`git`** repo included so do not "forget" to do it. This assignment builds on our previous assignment, adding several features including simple graphics and file loading. Do not skip steps. Also, I am aware not all the methods you create are directly used in this assignment. You still need to create and test them, we will use them in the future. There are also several methods you might need to create that are not specified.

You also may want to make backups if you are unsure about git. You can take the whole folder (**`.git`** file included) and put it in a **`.zip`** file if you are worried about breaking things.

Git Tips:

- Do not commit any file that should not be there. It is harder to remove files that have been added then to not add them in the first place. The commit is meant to be a reflective phase where you review your work. You CANNOT UNDO a commit and anything that gets put in will be in the record forever.

- The append option on a commit will allow you to revise or add additional files to the LAST commit you made, after that it is in the permanent record.

- Each git commit should include the Phase and a clear and complete summary of what code was added.

- If you find a bug, include that bug as its own commit. You should include the specific error (eg. Null reference) and how you fixed it. This will be a useful reference when you encounter the same error again.

- You may need to enable hidden files in your Windows/Mac file explorer in order to see the **`.git`** and **`.gitignore`** files.

Tips:

Start early. Read the assignment fully before beginning. We will go through it in the first class after it has been handed out. That is your chance to ask questions

Seriously, start early. I know you are busy but it is much easier if you get stuck early and can get help troubleshooting.

Notes on Academic Misconduct:

Your code should be done entirely by you. Any suspicion of Academic Misconduct or plagiarism will be fully investigated and can lead to serious academic penalties including 0 on the assignment for a first office, or an F in the course. Third offences could lead to expulsion from ICM. If you have any questions as to what qualifies as Academic Misconduct, please discuss it with me.

Things that are considered Academic Misconduct include:

- Copying code from old assignments
- Copying code from classmates or other people
- Telling or being told the specific steps required to solve a problem
- Copying code off the internet.

What you are allowed to do:

- Discuss examples given as handouts, from the textbook or class notes with others.
- Get together to study and help each other with basic understanding of concepts (eg. What is an object).
- I am considering Git to be a tool you are using, so feel free to help each other with usage of SourceTree and git.

Phase 0:

When you first start your project, before writing any code, you will create a new **`git`** repo in the same folder. You should ignore any **`.class`** files, or any files associated with your text editor (basically ignore every file type that appears other then **`.java`** or **`.txt`**)

Create a new local **`git`** repo, then make your first commit here, which will simply be the **`.gitignore`** file that was created with a message (such as "Initial commit" with a list of the types of files ignored.

Phase 1: Map Object

Complete the **Map** class. A **Map** holds the state of the background tiles for a single **Place** as a **2d array** of **char** values. A **char** value of **'X'** is an obstacle while a char value of **'.'** is an open space.

Each **Map** object will have a **2d array** of type **char** variable **currentMap**

Provide the following methods:

Constructor which accepts an **int index** which loads the array of test maps in the Map class by index. Eg, an **index** of **0** loads the first "**map**" **String** from the array of example maps.

public void drawMap()
This will draw all the graphics for the background. The **currentMap** provides the **char** keys, you will have to check the specific **char** and load the corresponding image file and draw it. Make sure your canvas is **512x512** and all your art files should be **64x64 pixel .png** files. You will only need two for now. Draw some images to use (you can use ms paint)

Tip: We are using **512** and **64** as sizes for a reason, also, **StdDraw** anchors the image by the centerpoint, so you will need to calculate the correct position to place it.

You will want to create the following methods before you do **drawMap**, to help

private String lookUpArt(char tileCode)

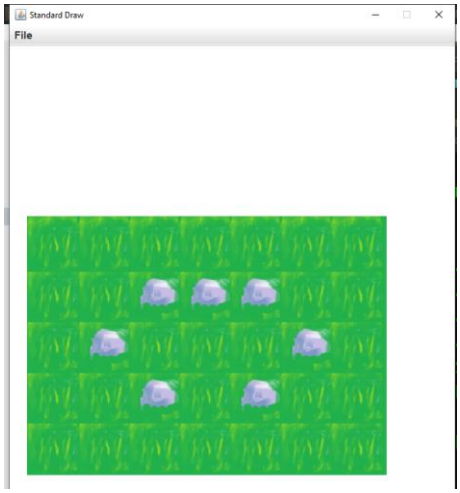
convert the **char** to the **String filename**. Include an image file to use for a default if invalid letters are detected (which I will test for).

Example:

```
// Running this code:
Map m = new Map(Map.getBoard(1));
m.drawMap();
StdDraw.show(20);
```

Loads the map from this template:

```
".......",
"..X.X..",
".X...X.",
"..XXX..",
"......."
```

A …nd would produce output (see screenshot - 

>).

public static char[][] stringToChar(String[] strings)
Converts an array of **Strings** to a **2d array** of **chars**. Keep in mind a **String** is just an **array** of **chars**, and can use **theString.length()** method

private static char[][] getBoard(int index)
Load the specified test board from the **testBoards array** by **index** and convert it to a **char[][]** before returning it.

You should be able to run the test method in **Map** when this is done. It should load the specified image file. You should also create yourself two image files **(64x64 pixels**) as a **.png** to use for this.

Phase 2: FileLoader class to Load Person objects

In this phase, you will implement the **FileLoader** class. It will provide some **static** methods for loading a **Person** from either a **String array** within **FileLoader**, or a **File** via **BufferedReader** and **FileReader** classes.

- Don't forget to import the **Scanner and java.io.*** libraries

public static PersonList loadPeople()

Load all people contained in the **testPeople String array** as **Person** objects within a **PersonList**

public static PersonList loadPeople(String fileName)

Load all people defined in the given file.

public static Person loadPerson(int index)
First load test data from the **String array** to parse (no File IO) and return a **Person**

private static Person parsePersonString(String aPerson) throws Exception
Converts a **String** describing a **Person** to a **Person** object. You will need to set the **delimiter** of the **Scanner** to use **","**

Phase 3: Place upgrades

A **Place** object should store both a **PersonList** (people currently at that **Place**) and a **Map** (the graphics/floorplan for that **Place**).

Create a **Place** constructor that accepts not only the **name** and **description**, but also the **Map** and **PersonList** as an overloaded version of the **Place constructor**.

You should also add a **drawMap**() method to **Place** that drawing the graphics contained in the current **Map** object.

Edit the main method of **PartySim** so that the **Map** is loaded from the **String array** in **Map** by **index** and the **PersonList** is loaded from the provided file instead of being randomly generated. The **mingle** method should work the same as before but instead use the newly imported PersonList from the file. For now, the main **PersonList** in **PartySim** and the **PersonList** in **Place** will be the same, but we may change it later so don't worry about it.

Phase 4: Drawing All Graphics.

Add two **double** values to **Person**, the current **x** and **y** position of the **Person**. These should be between 0-1.0. You should also add a **String artFile** variable which will hold the file to load when drawing this person. Add methods **setPosition**(**double xPos,double yPos), getXPosition()** and **getYPosition().** You should also add a **drawPerson** method which will draw the **Persons** art to the canvas at the specified position.

Edit the main method in **PartySim** to draw the **map** and then all **people** at the position they are at on the board. This will be a static image (not animated yet).

You will likely have to add additional methods to **Person** to accept or return these values. You should also update the file loading files and test data to accept / read in an optional **x /y position** and **filename** (if they are provided). If none are provided, include default values of respectively.

Hand in instructions:

Place all your **`.java`** files, the **`.git`** repo and the **`.gitignore`** files in a single .zip file with the name LASTNAME_FIRSTNAME_A2.zip (NOT RAR or other archive formats) in the hand-in folder on Moodle with your **`readme.txt, honesty declaration and test_output.txt`**

DO NOT:
- DO NOT hand in **`.class`** files

- DO NOT hand in **`bluej`** files or other various editor files **`(.vs, .proj, etc).`**
- DO NOT ignore the hand in instructions.

Congratulations, you are done. Be proud of yourself and relax.

Bonus:
Consider and suggest additional features that we could add. Include notes about how you would design the feature and what the expected output would be. Alternatively, add a new feature of your choosing. It should be well documented and build upon the existing functionality.

Hand In Additional Notes:

- It is especially important that your classes and methods are very accurate. Make sure your class names are **`Uppercase`** and your **`methodNames`** are lowercase and don't contain typos. **If your program doesn't compile for the marker you will lose significant marks**, even if it works on your machine.

- Also be sure you remove any package headers added by IntelliJ or Eclipse etc as these will break compilation.

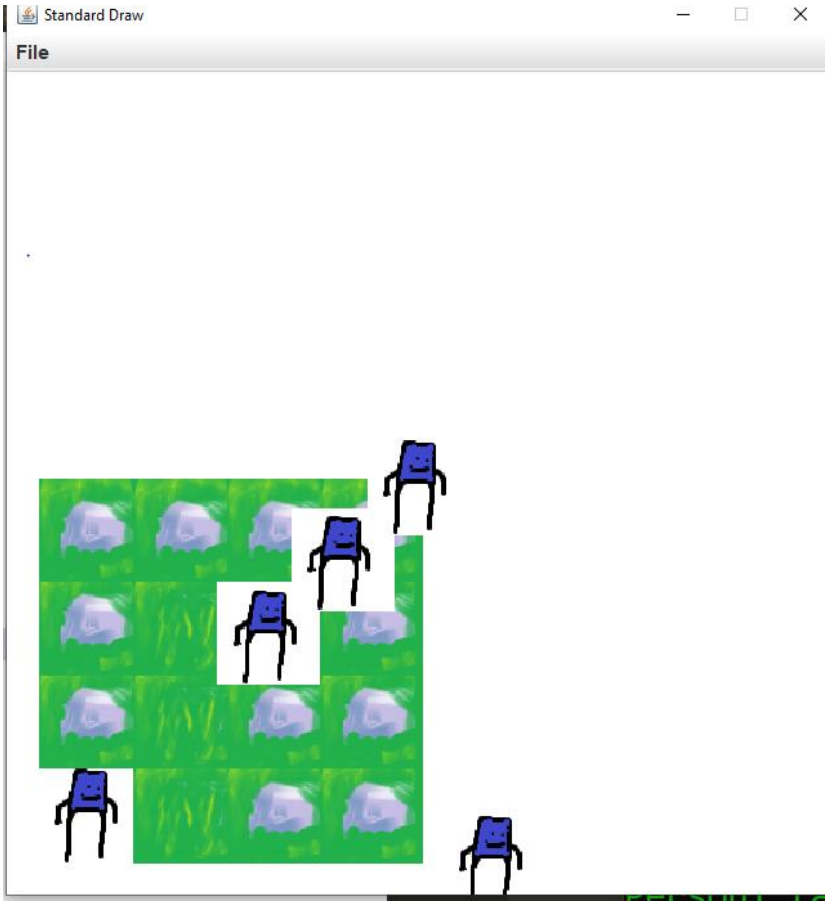
- See the **`SubmissionGuidelines.pdf`** document for full instructions. See the **`Standards.java`** document for formatting guidelines.

- Include a **`README.txt`** file when you hand in the assignment to tell the marker which phases were completed, so that only the appropriate test can be run. This can also include messages about errors you are encountering.

- For example, if you say that you completed Phases 1-3, then the marker will compile your files to check for completion of those phases. If it fails to compile and run, you will lose all of the marks for the test runs (at least half of the assignment). The marker will not try to run anything else, and will not edit your files in any way.

- Submit a completed Honesty Declaration with EACH assignment. Your assignment will not be marked without it. I will provide an easy text based version you can use so you don't have to mess around signing PDFs.

Good luck! Don't Panic!



Final output example (with provided data).