DUE DATE: MARCH 27, 2019 AT 11:59PM

Instructions:

- You must complete the "**Blanket Honesty Declaration**" checklist on the course website before you can submit any assignment.
- Only submit the **java files**. Do ***not*** submit any other files, unless otherwise instructed.
- To submit the assignment, upload the specified files to the **Assignment 4** folder on the course website.
- Assignments must follow the **programming standards** document published on UMLearn.
- After the due date and time, assignments may be submitted but will be subject to a late penalty. Please see the ROASS document published on UMLearn for the course policy for late submissions.
- If you make multiple submissions, only the **most recent version** will be marked.
- These assignments are your chance to learn the material for the exams. Code your assignments independently. We use software to compare all submitted assignments to each other, and **pursue academic dishonesty vigorously**.
- Your Java programs must compile and run upon download, without requiring any modifications.

Assignment overview

In this assignment, you will make a simple version of Space Invaders – an old arcade game. See the Wikipedia entry, and a gameplay example. You can play a version online here. In the game, a swarm of aliens moves back-and-forth across screen, moving down once the swarm has reached the end of the screen. The player, at the bottom, moves back-and-forth, shooting the aliens.
The game is won if the player successfully shoots all the aliens. The game is lost if the aliens reach the bottom.
To do this, you will use one or more ArrayLists to keep track of all the moving objects on the screen. Objects will be added to the ArrayList(s) as new objects are added to the screen, and removed from the ArrayList(s) as objects are no longer on the screen.
A 'Display' class has been provided for you, which sends commands to your code, much like it is done in Processing. You are encouraged to read the code in Display.java to see how the objects you are passing to it are handled and displayed, and to see an alternative to using the StdDraw class for drawing to the screen in Java.

class SpaceInvaders

When the Display.java program runs, it creates an instance of class SpaceInvaders. This is the object that controls the game state. It must implement the following methods:

- public SpaceInvaders(int height, int width) – constructor to build the game, that plays in a screen that has the size of the provided height and width.
- public void update() – this is called 30 times per second to update the placement of all your items, akin to the draw() method in processing. In this method:
  - Move any bullets that are on the screen by a set amount of your choosing
  - Check if the bullet has made contacted with any aliens. A single bullet can only connect with a single alien. Once the bullet has contacted an alien, both the alien and bullet are removed from the screen.
  - Move any aliens on the screen by a set amount of your choosing. The aliens always move at the same speed (in the classic video game they speed up, this is not a requirement for this assignment). The aliens always move as a single unit – they all move left, or all move right. When the aliens reach the side of the screen, they must move down some amount (up to you), and reverse direction.
- public ArrayList<Sprite> getItems() – this method returns an ArrayList of all the items that are to be drawn onto the screen. Details of class Sprite are below.
- public int status() – Returns the status of the game. This is used by the Display object. Return the appropriate constant to tell the Display class what action to take.
  - Display.CONTINUE – the game is not complete. The game loop continues.

   - Display.WIN – the game was won. The Display class will show the appropriate message and quit.
   - Display.LOST – the game was lost. The Display class will show the appropriate message and quit.
- public void move(int direction) – This method is called when the player presses the left or right arrows. If the player presses the left arrow, direction is set to Display.MOVE_LEFT. If the player presses the right arrow, direction is set to Display.MOVE_RIGHT. Move the ship left or right appropriately.
- public void shoot() – This method is called when the player presses the spacebar. When called, the player's ship checks if it can shoot a new bullet. They player is allowed to have a maximum of 2 bullets. If there are 2 bullets on the screen, do nothing.

The SpaceInvaders class controls all aspects of the game. It must track all the objects drawn on the screen in an Arraylist containing Sprite objects. When the player shoots a bullet, it needs to be tracked until it hits an alien, or has gone past the top of the screen. When an alien is shot, it must be removed from this list. The ship will be in this list, too.

The SpaceInvaders object checks for collisions between the bullets and aliens. Use a simple straight-line distance between the center point of the bullet and the center point of the alien. Choose an appropriate "close" distance of your choosing to decide if a bullet has hit an alien. This will make the 'hit box' of the aliens a circle shape.

When starting the game, create a n by m grid of evenly-spaced aliens, storing the Alien objects in your ArrayList of Sprites. The game will start immediately when the program is opened.

The implementation of this object is quite open. You must maintain at least one ArrayList of Sprites that is given to the Display class, but can maintain more ArrayLists if it simplifies your code. Write helper methods to keep your methods short. Do not keep nulls in your ArrayLists, only keep instantiated objects. Do not use partially-filled arrays.

abstract class Sprite, and subclasses

All items drawn to the screen are a subclass of class Sprite.

You will need to add some accessor methods to update the X and Y coordinates. To work with the Display class, the class Sprite must implement the following instance methods:

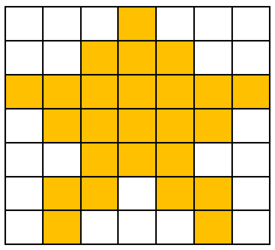
- public int getX() – gets the x coordinate.
- public int getY() – gets the y coordinate.
- public Color[][] getColorGrid– gets a 2D array of Color which can later be used to draw the Sprite on the window. You will have to import java.awt.Color to use the Color object.

All item that are drawn to the window are subclasses of type Sprite. You must make a minimum of 3 subclasses of class Sprite:

1. An object for the player's ship
2. An object for an alien
3. An object for a bullet

You are provided shapes for the ship, and an alien. Feel free to modify these shapes, or add more alien objects that are different shapes. Using a 2D Color array, make a shape for the bullet.

The 2D array of Color is a 2D 'pixel' representation of an image. For instance, a star might look similar to:



In the 2D array, a null value means that the pixel at that location is "empty". Every other element should contain a Color for that pixel (defined by a Color object). There are colour options available in Display.java that you can use, or you can look in to the java.awt.Color object to create other colours. Look at the examples in the Display class on how to create a 2D array of Color.

Bonus +5%

In the original Space Invaders, the aliens shoot back. Update your assignment so that any living aliens has a random chance on any update() of creating a new Bullet (or similar object, this is up to you) that can hit the player's ship. Alien bullets do not harm other aliens. If the alien bullet hits the player's ship, the player loses.

This is an all-or-nothing bonus. No part marks will be given for incomplete solutions. Make a note when handing in your assignment that you have completed the bonus.

Notes

Assignment 4 tests your usage of ArrayLists. The implementation of how you use the ArrayLists to play the game is left to you. The number, shape, size and speed of aliens is up to you. The point in which the aliens "drop and reverse directions" is also up to you. It is ok for the aliens to move slightly outside the bounds of the display window in their cycle.

You do not need to change the colours of the sprites – but it does make the game more fun.

Hand in

Submit your Java files, you do not need to hand in Display.java. ***Do not submit .class or .java~ files!*** If you did not complete the tasks detailed in the assignment, use the Comments field when you hand in the assignment to tell the marker which parts were completed, so that only the appropriate tests can be run. The marker will ***not*** try to run anything else, and will ***not*** edit your files in any way. ***Make sure none of your files specify a package at the top!***