CMSC 113: Computer Science I
Midterm Project: Rescue
due on Gradescope by the beginning of class on February 27, 2018

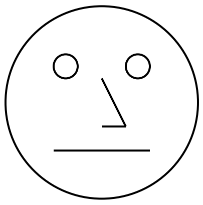
For this project, you will write the game Rescue. The aliens have decided to return all of the Robins they abducted. However, they refuse to land and let the Robins off. Instead, they drop Robins one at a time from their mother ship. Your task is to control a mattress so that all of the Robins land safely.

The Rescue Project is broken into three parts. It is best to proceed by doing one part at a time. Get everything about Part I working before going on to Part II, for example.

Collaboration Policy: You may choose to work with a partner for this project. If you work with a partner, let me know as soon as you make your decision. If you choose not to work with a partner, you must follow the guideline "If you're talking in Java, you've gone too far." Remember, I take a breach of this policy seriously – if you're uncertain, ask before making any assumptions.

Part I. Write an applet that shows a mother ship pacing back and forth horizontally across the top of the applet window. Assume the applet is 200 pixels wide by 200 pixels tall. Use a compound object `MotherShip` to represent the mother ship. Your compound object must have a method named `update`, which moves the mother ship one step forward and handles turning around at the sides of the applet. All of the logic around bouncing must be in the `MotherShip` compound object. Recall that compound objects work best if the drawn elements within the object are centered at or near the hotspot (the origin in the local coordinates of the object).

Extra challenge: Implement the `MotherShip` class as above, but make the ship speed up every time it moves. To ensure the ship speeds up in a smooth manner, use a `double` field to store the speed of the ship. In the online example, the ship's speed after an update is .01 faster than its speed before.



A lot of people have trouble getting this to work. They think that when you detect that the ship has gone past the right edge, make it move left. But that doesn't work! The ship does move left one pixel, but then moves right again. The trick is to have a field to control the ship's direction.

Part II. Add a mattress to the bottom of the applet. When the user presses the left- or right-arrow keys, the mattress moves accordingly. The mattress should always move smoothly and should always stay completely on the screen.

Use a `Mattress` compound object. Your `Mattress` class should have an `update` method that takes a parameter saying how much to move by; negative numbers move left and positive numbers move right. The method should check to make sure the mattress will stay on the screen and then move.

Extra challenge: Make it so that the applet tracks the total number of pixels the mattress has moved. Do this using a field in the `Mattress` class. The only extra field you should add to the applet is a `GLabel` field.

Part III. Add the following features:

- The mother ship should drop one Robin at a time. The Robins fall straight downward. Implement this by using a `Robin` class with an `update` method that moves it down.
- When a Robin hits the mattress, score one point and start a new Robin to fall.
- Create a label showing the current score.
- If a Robin falls below the bottom of the screen, stop the game.
- When the applet loads, the game should not start immediately. The game will start on a mouse click.

While I have made specific suggestions about code design above, make sure that, as much as possible, ship-related code goes in the `MotherShip` class and mattress-related code goes in the `Mattress` class.

Extra challenge: Send each Robin in a randomly-chosen direction. After every Robin starts its fall, choose a random *x*-velocity. The falling Robin moves at that velocity (which could be to the right or to the left) and bounces off the walls. As time passes, increase the range of possible velocities. For example, my version starts off with a random velocity between -1 and 1 and increases those limits by .3 with each Robin. So, the second Robin's velocity is between -1.3 and 1.3, and so on. Robin's *y*-velocity always stays the same.

When you have completed coding this project, make a *reflections.txt* file (following the instructions on the Warmup assignment), answering these questions:

1. Did you complete all parts of the assignment?
2. If not, which parts were you unable to complete and why?
3. How long did this assignment take you?
4. What was the most challenging part?
5. Do you have any other questions or experiences to share?

Export your project (following the instructions in the Warmup assignment) and post on Gradescope.