School of
Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

Assignment 5

Intro to Programming: Assignment 5

- Due Dec 8 23:30

Goals

This assignment will give you experience in defining classes for objects with fields and constructors.

Resources and links

- Download Zip file of necessary code and data.
- Submit your answers.
- Marks and Feedback
- Watch a video of the Gliders Assignment

Summary

- Glider Simulation:
  ➡ Write a program to simulate paper gliders. Complete the `Glider` class that defines Glider objects.

Marking: Up to 65 for the core, up to 90 for the completion, and up to 100 for the challenge.

To Hand in

You should submit your version of `Glider.java` by the due date.
If you do the challenge, then submit your version of `GliderSimulation.java` also.

Glider Simulation

The `GliderSimulation` program is a simple animation of paper gliders that start on the left side of a window, and glide across to the right.

The structure of the program is very similar to the `CartoonStrip` program and the "new objects" exercises in assignment 2: there is one class (`Glider`) that represents individual gliders, and another class (`GliderSimulation`) that creates two Glider objects, and has a loop that makes them to move. For the `CartoonStrip` program, the class for the individual objects (`CartoonCharacter`) was written for you, and you had to write the class that controlled the objects. For this program, it is the opposite: the "controlling" class is written for you, but you have to write the `Glider` class for the objects.

Read the `GliderSimulation` program to see how it makes the two new Gliders, and calls methods on then. Each time round the main loop, the program will
- clear the graphics pane
- `move` both gliders by one step.
- `draw` the gliders in their new position and redraw the room
- check whether a glider has gone too high, in which case it makes it slow down; or too far, in which case it replaces the glider by a new one.

The `makeNewGlider` method creates a new glider at the left edge of the window with a random height and a random speed.

The `Glider` class represents a glider that is flying to the right. This is the class that you must complete.

The state of a glider consists of
- its horizontal position,
- its height above the floor (from the floor to the bottom of the Glider), and
- its speed. The speed is how far it will move to the right in each step.

Each new `Glider` will have a different starting height and a different speed. Therefore, the constructor for `Glider` has two parameters for the initial height and speed.

The `Glider` class has five methods:
- The `draw()` method should draw the glider at its current position (as stored in the fields).
- The `move()` method should make the glider move one step to the right. Depending on the speed, it should also move the glider up or down.
  - Gliders on the floor (height is 0) don't move at all
  - Gliders going at MID_SPEED don't change their height.
  - Faster gliders will rise (the faster they go, the faster they rise).
  - Slower gliders will fall (the slower they go, the more they fall).
- The `getX()` method should return the horizontal position of the glider
- The `getHeight()` method should return the height of the glider above the floor.
- The `setSpeed(double sp)` method should change the speed of the glider.

The `Glider` class has a `test` method written for you that will help you test your code by creating a glider and putting it through its paces. You can use BlueJ to call the `test` method. Call it repeatedly to check that
- gliders report when they go past the right side of the box
- fast gliders slow down when they go above the top of the box
- slow gliders get to the ground and stop moving.

Core

- Complete the definitions of the fields and the constructor,
- Complete the `draw` method,
- Complete a simple version of the `move` method which just moves the glider to the right (doesn't change its height)
- Complete the `getX` method,

This will make gliders that move straight across the screen, and disappear when they hit the right wall

Completion

- Improve the `move` method to make the glider go up or down, depending on its speed,
- Improve the `move` method so that the glider will not move if its height is 0, and never has a negative height.
- Complete the `getHeight` and `setSpeed` methods so that the simulation can slow the glider down if it goes past the ceiling in the simulation.

Note: you will have to think carefully to work out how much the glider should go up or down on each step so that very slow gliders go down at a steep angle, very fast gliders go up at a steep angle, and medium speed gliders only go up or down a little bit.

Challenge

The Glider program was inspired by the 1988 macintosh game called "Glider" (look it up on wikipedia, or get "Glider Classic" for iOS) but this program has none of the gameplay.

Extend your program to introduce some game play to it.
- Allow the player to speed up or slow down the gliders (using buttons or sliders).

- Add an obstacle that the glider has to get around without crashing into.
- Add a scoring element that increases the score for every glider that gets to the right hand wall, and loses points if the glider crashes.

Note, these changes require changing the simulation program. You may also need to change or add methods in the Glider class.