Asteroids

Objective:
To make a simple Asteroids clone, and then extend it in an interesting way!

Getting started:
Open Terminal and type **git clone https://github.com/bradleypollard/GameDev1.git**
Open Eclipse, then:
File → Import → General → Existing projects into workspace → .../GameDev1/

What to do:
Now you should have cloned and imported the skeleton project, and hitting the play button in Eclipse should show an empty window with an FPS counter. Now it's time to make a game!

JavaDocs for Slick2D (the library we are using) can be found here, if you are not sure how any features work: http://slick.ninjacave.com/javadoc/

To keep things simple, I recommend not worrying about object orientated programming, and just implementing most of your code in the *GamePlayState* class.

1. Start by trying to load a sprite, using the *Image* class. When creating an instance of the *Image* class, it's constructor can take a string representing the path to the image file for the sprite.

Image sprite = new Image("assets/ship.png");

Get it to display on screen by calling it's *draw* method in the *Render* part of the game loop, giving x and y coordinates as arguments. **Remember:** The Y Axis starts at 0 at the top of the screen, and counts up when going downwards!

2. One of the arguments to the *Update* method is a *GameContainer*. This can be used to get input from the player, using the *getInput* method! Store the input in a variable, for example:

Input in = gc.getInput();

Try using IF statements to move the sprite's position in the update loop. If you call *in.isKeyDown(Input.KEY_W)* you can check if the W key is being held down, and then move the coordinates of your sprite accordingly!

> **a)** Once you have the sprite moving on keyboard commands, try adding acceleration and deceleration, or even rotating the sprite to add turning!

3. Now things might get a little tricky. Next up is shooting, which will require you to create a new class for the first time. The bullet will need its own x and y coordinates, x and y velocity etc.

Then, every time the player presses space, you can create a new instance of the *Bullet* class and add it to a *List* of bullets to be rendered and updated each frame.

You'll also need to think of a way to destroy the bullet after a while, otherwise you'll have hundreds of bullets somewhere off the screen! I'll leave that as a puzzle for you to solve.