Larger Assignment, Part 1.

We are making a game called Asteroids.

It is called Asteroids and it was made quite a while ago on much older hardware and with totally different code libraries. If you haven't heard of it you can play it here: http://my.ign.com/atari/asteroids

You can read more about Asteroids on Wikipedia: http://en.wikipedia.org/wiki/Asteroids_(video_game)

Your final game should:

1. Create a bunch of asteroids (of quantity: numAsteroids)
2. Allow the user to fly a ship:
    2.1. up / down should increase decrease speed
    2.2. left and right should change direction of the ship (not make it move left / right)
    2.3. space should make the ship fire a "laser"
3. Lasers should:
    3.1. travel for a time and then disappear.
    3.2. if they collide with an asteroid cause the asteroid to explode (disappear) at which point the laser too should disappear
4. Asteroids should:
    4.1. move at a consistent speed and direction
5. When all the asteroids disappear, the player wins
6. The player should start with three lives, each time his or her ship collides with an asteroid, a life should be lost. If the player has lost all of the lives the game should end

A few things that are relevant.

1. This will only work in Processing. If you really, really love eclipse or compiling your java from the terminal. If you download the non-processing-IDE version on the website it *might* work, but no promises.

2. Each of the main classes (Player, Asteroid, Laser) has an updatePosition() method and a draw() method. The draw method is called by the main draw() method, for each type of thing. The draw method itself should call the updatePosition() method first. That method will figure out where it should be *next,* as in, the next frame. And then the draw method should proceed to draw itself. All of this really will happen 30 times a second.

3. I am using two "ArrayList"s in this code.

   ```
   ArrayList<Asteroid> asteroids;
   ArrayList<Laser> lasers;
   ```

   These are like arrays, but they take care of some of those annoying things that you hate about using Arrays. For example you can remove one element, and it will shift all the rest around. They don't have a max size, they grow and shrink as needed.

`lasers.get(0)`	returns the 1st element in the arraylist
`lasers.get(i)`	returns the ith element in the list
`lasers.size()`	returns the number of real elements in the list (int)
`lasers.add(aLaser)`	adds a Laser named aLaser to the list
`lasers.remove(i)`	removes the ith element in the list
`lasers.remove(aLaser)`	removes the element aLaser from the list if it is in the list
`lasers.clear()`	removes all the elements in the list

I believe that is all the functions you really *need* but for more about ArrayLists refer to our class on November 20th, or the javadoc: http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html

Our code will contain the following classes and methods:
Methods written in bold are complete and do not need to be edited.
But can be.

Main

- **setup()**
- draw()
- **void drawAsteroids()**
- void drawLasers()
- **void addLaser(Laser input)**
- **void removeLaser(Laser input)**
- void checkAsteroidDeath()
- void checkShipDeath()
- void resetShip()
- **void keyPressed()**
- **void showLives()**
- **void showScore()**
- **void showAsteroids()**

Asteroid

- Asteroid()
- **float getX()**
- **float getY()**
- void draw()
- void updatePosition()

Laser

- Laser()
- **float getX()**
- **float getY()**
- void draw()
- void updatePosition()

Player

- Player()
- **float getX()**
- **float getY()**
- void draw()
- void updatePosition()
- void setSpeed()
- void setDirection()
- Laser shoot()

The main file also starts with a number of parameters.
These will be things that you may want to adjust as you build the game.
You may also add to this list:

```
// Game State
int numAsteroids = 30;   // start number of asteroids (not
updated)
int lives = 3;           // number of lives
int score = 0;           // player's score

// Display Properties
int charSize = 16;       // size of the ship

// Movement
int   incVal = 100;            // amount speed increases
float amountMove = 100;        // amount to move each step
float slowDown = 100;          // the ship slows down over time
                               // (I don't know why)
float directionChange = 100;   // how much the ship rotates with
                               // left-right keypress

// Mode
boolean playMode = true;       // is the game playable?
```

Remember to comment your methods and follow our class coding standard.

There is lots of Asteroids code online, and in general I recommend you avoid it. The sample code I have provided will be much more in line with what we have learned in this class.

Updates:

None so far.

Asteroids: Bonus possibilities.

Asteroids break into other asteroids. +5

Spec:

When a laser/weapon hits an asteroid of some (large) size
that asteroids breaks into two (or more) asteroids of smaller size.
(number of asteroids should go up)

When a laser/weapon hits an asteroid of some (small) size
that asteroid disappears.
(number of asteroids should go down)

Structural note:

most changes will be in the Asteroid class,
some may be needed in the main game code.

Alien ship. +10

Spec:

New class.
Name up to you, suggestions: Alien, Enemy, etc.

Ship should look distinctive from Player ship.
Ship should fire weapon (that looks different from Player Laser - though it can still be the Laser class or a new class.)
Think about if it fires towards the player? Does it always hit? Does it *ever* hit?

Ship should take more than one hit to be destroyed. (Number up to you.)
Ideally ship color/graphic style *could* display based on number of hits left before death.

Ship *could* move in a somewhat interesting way. (i.e. not a straight line)

Ship should give larger score bonus than asteroids.
Ship should not appear right at beginning of game.
New enemy ship should appear some amount of time after last enemy ship death.

Also think about what happens if the player dies while an enemy is on screen, does the enemy stay? does the enemy just disappear?

Does it at least need to be moved away from the center if it might blow up the player again?

Structural note:

most changes will likely be in the added Enemy class, though modifications will be required to the main code, and the Laser class.

Graphics. +5

Spec:

Add graphics, make them look nice. It doesn't need to space themed. The more aesthetically pleasing the better. This is subjective. I will be subjective.

Structural note:

most changes will likely be in the draw() methods for each class.

Graphics that change +5 more

Spec:

Add *transitions* (either in code or in graphic files) that happen at points of change:

- The beginning of the game.
- The ship exploding.
- A laser fizzling out.
- A laser hitting an asteroid.
- The enemy exploding.
- The end of the game.
- And more...

Structural note:

most changes will likely be in the draw() methods for each class as well as the main game program.

Powerups. +10

Spec:

Create a PowerUp class that makes random objects appear on the screen (hopefully not too frequently).

If the player touches a PowerUp he or she gains that ability for a limited amount of time (the amount of time must be limited! This game is already quite easy.)

There must be at least three different powerups, though the actual choices in your game are up to you.
Possibilities (not a complete list you can make up others):

- Rainbow lasers.
- Multiple laser fire.
- Steady stream of laser.
- Mines.
- Shields.
- Extra lives.
- Freeze time (the asteroids stop moving).
- Mini fighter drones that go kill asteroids.
- And more...

Each of these should look unique (a player should know what they will get before they drive over it --- optionally you could also have a mystery powerup in the mix, but not always).

Ideally, since these are all time limited, there is some sort of on-screen awareness for the player of how much time is left before it expires.

Structural note: this will not only require an additional PowerUp class but changes to other classes as well.

Other additions +? points.

You can add other additions to your Asteroids game. If I find them interesting I will add additional points. They should be well documented in the source so that I understand what they do.

If you add additional button functionality, please make a note in a readme.txt file that comes with your game.