CS215 Spring 2010 Program 3 Due 3/25 2010
StarExplorer -- a game framework.

Primary learning objectives:
- Creating and using subclasses in C++
- Creating and managing a dynamic data structure.
-

Ancillary learning objectives:
- Experience with common data exchange formats -- JSON
- Using provided 'object libraries'
- Writing 'to a spec'
- Introductory sorting

In this assignment you will be modifying an existing object, adding some specific functionality, and creating a new companion object. The objects are eventually destined for a web-based game, but for now you only need to provide some basic functionality, and verify the correct output with a web-based tester.

The game was originally designed to allow users to create spacecraft, explore the universe, collect items and meet aliens. However, marketing has just convinced management that the game needs more excitement, specifically they want to add the ability to configure and use weapons. While the final details of the game are still to be determined by the game designers, they have made a few decisions which you can begin to implement.

First a Weapon class needs to be implemented, it should manage the following attributes:
- The amount of Damage the weapon can do, as an integer.
- A count of the pieces of ammunition, or number of 'fires' the weapon has remaining.
- An indicator signifying that the Weapon has been fired at least one time.
- The maximum effective range of the weapon, also as an integer
- The type of the weapon, currently one of:
  - `BB`
  - `Missle`
  - `Laser`
  - `Nuke`
  - `Phase Cannon`
  - `Anti-Matter Blast`
  - `Singularity Grenade`

And provide the following functionality:
- get/set all attributes
- return the weapons 'power factor' (damage * ammunition)
- return a 'string representation' of itself (see below)

The existing Explorer ship class will need to be extended to be able to utilize your new Weapon class. Though an object was already created to represent an Exploring ship, it was created by a consultant who did not leave the source, only the class description and object file. So in order to extend its functionality, you will need to create a subclass. The subclass will need to add the ability to manage a set of weapons.

The Explorer class should be extended to a BattleExplorer class, which will allow for the addition of any number of Weapon objects. The Objects should stored internally as an array of Weapon objects. The array should be able to grow to any size, and will start out with a fixed size of 4. Once all 4 available slots are occupied, adding a 5th weapon will double the size of the array. Adding a 9th weapon will double the size again and so on. The array should always be kept in order, with the weapon having the largest powerFactor first (i.e. at index 0 of the array).

In order transmit its state to the server, the Explorer class should be able to dump out a representation of itself into a string in a particular format. The format looks something like this:

```
{ Attribute :Value, Attribute:Value }
```

Attribute Value can be another Name:Value pair. It can also be a list of items using the following format:

```
{ Attribute : [ item , item , item ] }
```

Each 'item' can be a single value, or an Name:Value pair. All of these can be logically nested.

The existing Explorer class also utilizes a Senor class and a Propulsion class. Though the source for Explorer is not available, the source to these two classes still exists, and they might be useful as a reference in building your Weapon class.

You can test the output of your class via a web page. Visit http://sweb.uky.edu/CS215/StarExplorer/ and paste in the output of your program in the field provided. If it returns a short description of your ship and the phrase 'Valid Weaponized Explorer', then your code is functioning correctly.

Attached is an example of what the JSON output should be for a proper Weaponized Explorer. Note both the syntax of the output, and the name and value data being displayed. Be sure that your code generates output in this format, and using these names.

Sample Valid JSON output

```
{
  "Explorer": {
    "name":"E1",
    "value" : 0 ,
    "fuel" : 0 ,
    "crew" : 0 ,
    "speed" : 0 ,
    "xpos" : 0 ,
    "ypos" : 0 ,
    "zpos" : 0 ,
    "propulsion" : [
       { "type":2, "consumption":0, "maxSpeed":0, "powerFactor":0 } ,
       { "type":4, "consumption":0, "maxSpeed":0, "powerFactor":0 }
    ],
    "sensors" : [
       { "type":5, "consumption":0, "maxRange":0, "deployed":false }
    ] ,
    "weapons" : [
       { "type":"bb", "ammo":878, "damage":923, "maxRange":709, "fired":false },
       { "type":"Missle", "ammo":73, "damage":658, "maxRange":930, "fired":false },
       { "type":"Missle", "ammo":492, "damage":42, "maxRange":987, "fired":false }
    ]
  }
}
```