AP Computer Science
Homework Set 4
Class Composition

P4A. (BlueJ) In this program we will write a `MyPhone` class to represent your company's new smart phone. Your `MyPhone` will include some information about the phone as well as your music library. We'll do this by adding an instance variable that is an array of type `Song` to your `MyPhone` class. You can use your `Song` class from P3A in this project. This can be done by copying the `Song.java` file from your **P3AProject** folder into your **P4AProject** folder.

Using one object as a "container" (the `MyPhone` class) to store another type of object (the array of `Songs`) is called "class composition" or the "building" of an object using other objects as its components.

Class `MyPhone` should include the following:

a. an instance variable for its color,
b. an instance variable for its memory capacity (in GB),
c. an instance variable of type array that can hold 4 `Song` objects called "`songLibrary`" that will store the songs in the `MyPhone,`
d. a two-argument constructor to initialize the instance variables `color` and `memoryCapacity`. You can populate the `songLibrary` in the constructor with four songs of your choice in the two-argument constructor.
e. a "brain" method called "`totalPlayTime()`" that will return an integer for the total number of seconds for all songs stored in the `songLibrary`.
f. a "brain" method called "`deleteAllSongs()`" that will remove all songs from memory by placing "`null`" into each element in the array.
g. a `toString()` method to display the object's three instance variables (including `songLibary`) in a user-friendly format.

Write a `MyPhoneDriver` to perform the following:

i. create a `MyPhone` object called `phone`,
ii. initialize its instance variables using its multi-argument constructor. Pick your color, memory capacity. The `songLibrary` will be populated in the multi-argument constructor.
iii. print the information for your `MyPhone` object using the `toString()` method of the `MyPhone` class,
iv. print the result of the MyPhone method `totalPlayTime()`.
v. call the `deleteAllSong()` method for your phone,
vi. print the information for your phone a second time to verify that all songs have been deleted.

P4B. (BlueJ) Upgrade the `MyPhone` to a `MyPhone2` to use an `ArrayList` instead of an array. Complete the following to complete the upgrade:

a. Modify the two-argument constructor to create and populate the `ArrayList songLibrary`.
b. Modify the `totalPlayTime()` method so that the method works correctly with the `ArrayList songLibrary`.
c. Modify the `deleteAllSongs()` method so that the length of the `ArrayList songLibrary` is 0 after all the songs have been removed.

P4C. (BlueJ) CollegeList…revisited. In this program, we'll write a program to model your college choices along with an algorithm that will help narrow down your choices according to a complex statistical analysis. Complete the following items below to help you filter your college choices:

a. Create an `ArrayList` called "`collegeList`" that can hold objects of type `String`.
b. Populate `collegeList` with the names of at least 5 colleges that you are applying to.
c. Write a `for loop` to print the names of the colleges in `collegeList`.
d. Write a second `for loop` to remove each school whose name has four letters.
e. Write a third `for loop` to print the names of colleges in `collegeList` again. Schools with names that consist of 4 letters should not appear in the printout.
f. Add "`UMass`", "`Army`", and "`Navy`" to the end of `collegeList` and run your program again. Are all schools with a name of length 4 removed?

P4D. (BlueJ) Write a program that creates a class called "`ClassRoster`" that will model a teacher's class roster. "`ClassRoster`" will store the name of the class as well as all "`Students`" registered for the class. You will be sent a template for this project.

Here are the specs for the `ClassRoster` class:

1. Include a `String` instance variable "`courseName`" that will store the name of the class, e.g., "AP Computer Science", "Shakespeare", "AP World History", etc.
2. Include an `ArrayList` instance variable called "`studentList`" that can store objects of type `Student`.
3. Write a one-argument constructor to initialize the name of the class AND create and populate the `ArrayList studentList` with at least 5 students.
4. Write the `ClassRoster` method `studentWithMaxGPA()` that returns the last name of the student in the `studentList` who has the max GPA. The header for this method is shown below:

```
public String studentWithMaxGPA()
```

5. Write the `ClassRoster` method `dropStudent` that removes all `Students` whose grade level is below (i.e., less than) the minimum specified by the parameter `minGradeLevel`. The method header for this method is shown below:

```
public int dropStudent( int minGradeLevel )
```

   The method should return the number of `Students` dropped from the `studentList`.

6. Write the `ClassRoster` method `addStudent` that adds the `Student` specified by the parameter `newStudent`. The method header for `addStudent` is shown below:

```
public void addStudent( Student newStudent )
```

P4E. (BlueJ) AP 2013 #1 – DownloadInfo

P4F. (BlueJ) AP 2015 #2 – HiddenWord

String method review:

```
public int length()
public String substring( int from, int to )
public String substring( int from )
public int indexOf( String inputString )
public compareTo( String inputString )
public boolean equals( String inputString )
```

P4G. (BlueJ) AP 2016 - #2 (LogMessage)

String method review:

```
public int length()
public String substring( int from, int to )
public String substring( int from )
public int indexOf( String inputString )
public compareTo( String inputString )
public boolean equals( String inputString )
```

P4H. (Greenfoot) Your next mission on the `MagicLewShip` is to visit stars in both the known and unknown universe and collect luminosity data for each star. Data for each star will be analyzed after it is collected.

Below are the parameters for your mission:

a. Design a `Star` class that will consist of the following:
   i. an integer instance variable representing the star's luminosity.
   ii. a zero-arg constructor, getter and setter methods for `luminosity`, and a `toString()` method for the class. The zero-arg constructor should initialize the `luminosity` value of the star to a random integer between 500 and 1000.
   iii. the `act()` method should decrement the luminosity of the star by 1 every time the star acts.
   iv. the `act()` method should remove the star from interstellar space if the luminosity value of the star is less than zero. This can be done by using the `World` method `getWorld().removeObject()` method.
   v. the act() method should print the value of the star's luminosity value using the `World` method `getWorld.setText()`. Print the luminosity value at the star's location as given by `getX()` and `getY()`.

b. Upgrade your `Spaceship` class to include an instance variable of type `List` called `starData` that can hold objects of type `Star`. Include an upgraded zero-argument and three-argument constructor that will create the `ArrayList` of type `Star`.

c. Write the `Spaceship` method `calcAverageLuminosity()` that will calculate and return the mathematical average of the luminosity data found in `starData`. You can use the `Galaxy` method `getWorld().getObjects( Star.class )` to populate an `ArrayList` of all of the `Star` objects in the `Galaxy`.

d. Write the `Spaceship` method `calcMaxLuminosity()` that will find and return the maximum luminosity data found in `starData`.

To test the operation of your upgraded Spaceship, complete the following:

i. Create and add to your `Galaxy` at least 6 `Star` objects, each one with a random luminosity between the value of 1 and 1000.

ii. Add an instance of your upgraded `Spaceship` class to the `Galaxy`.

iii. Call method `calcAverageLuminosity()` and print its output using `getWorld().showText()` within the `act()` method.

iv. Call method `getMaxLuminosity()` and print its output using `getWorld().showText()` within the `act()` method.

P4I. (Greenfoot) Your next mission to locate and remove `SpaceJunk` objects that are below a minimum altitude and represent the risk of falling out of orbit and colliding with the Earth and/or unsuspecting computer scientists. Google "Space Junk Map NASA" to find out what space junk actually is and to see the amount of space junk around the earth that has been mapped by NASA.

Below are the parameters for your mission:

a. Design a `SpaceJunk` class that models a piece of space junk that is orbiting the earth. Class `SpaceJunk` will consist of a single integer instance variable representing the space junk's `altitude`. Provide a one-arg constructor, getter and setter methods, and a `toString()` method for the class. The one-arg constructor should initialize the `altitude` of the space junk object to an integer value between 0 and 200,000 as specified in the constructor. The getter and setter methods should get and set the `altitude` instance variable, respectively. The `SpaceJunk` should move and "bounce" in random directions when it encounters the boundaries of your `Galaxy`. You can use the Actor method `isAtEdge()` to determine when it "hits" a boundary of `Galaxy.`

b. Upgrade your `Spaceship` class to include a `List` instance variable called `spaceJunkObjects` that can hold objects of type `SpaceJunk`.

c. Write the `Spaceship` "brain" method `removeSpaceJunk` that will scan the `Galaxy` for all known `SpaceJunk` objects remove those whose altitude is below the minimum as specified by the `removeSpaceJunk` method. The method will return the number of `SpaceJunk` objects that have been removed. The `removeSpaceJunk` method should have the following header:

```
public int removeSpaceJunk( int minAltitude )
```

Method `removeSpaceJunk` will use the `World` method `getWorld().removeObject( Actor a)` to remove objects from the world.

The removal will be done with your on-board eco-friendly proton beam powered by the gamma rays emitted by the annihilation of electron-positron pairs…in the spirit of finding good uses for antimatter particles.

To test the operation of your upgraded Spaceship, complete the following:

i. Create and add to your `Galaxy` at least 5 `SpaceJunk` objects, each one with an altitude between the value of 0 and 200,000.

ii. Add an instance of your upgraded Spaceship to your Galaxy.

iiii. Call method `removeSpaceJunk()` method in the Spaceship act() method to test if your proton beam is working as expected. `SpaceJunk` object(s) that that meet the removal criteria should be removed from the `Galaxy`.

iv. Print the result of the call to the method `removeSpaceJunk` using `getWorld().showText()` to verify that the number of `SpaceJunk` objects removed is correct.

Learning Objective Checklist

(please print and complete after you have had all HW Set 4 programs checked off)

Place a check next to those items that you have mastered

	Write a class definition that includes instances of another class (class composition)
	Write a class method whose output depends on the object's instance variables.
	Use `arrays` and `ArrayLists` in a class and be able to use either a `for` or `for-each` loop to traverse the array or `ArrayList`.