# Programming Assignment 6

Due: April 23 at 11:59 pm

## Exercise 1 (30 points)

For this exercise, you need to create a few classes that represent different astronomical body and compose a planetary system from them.

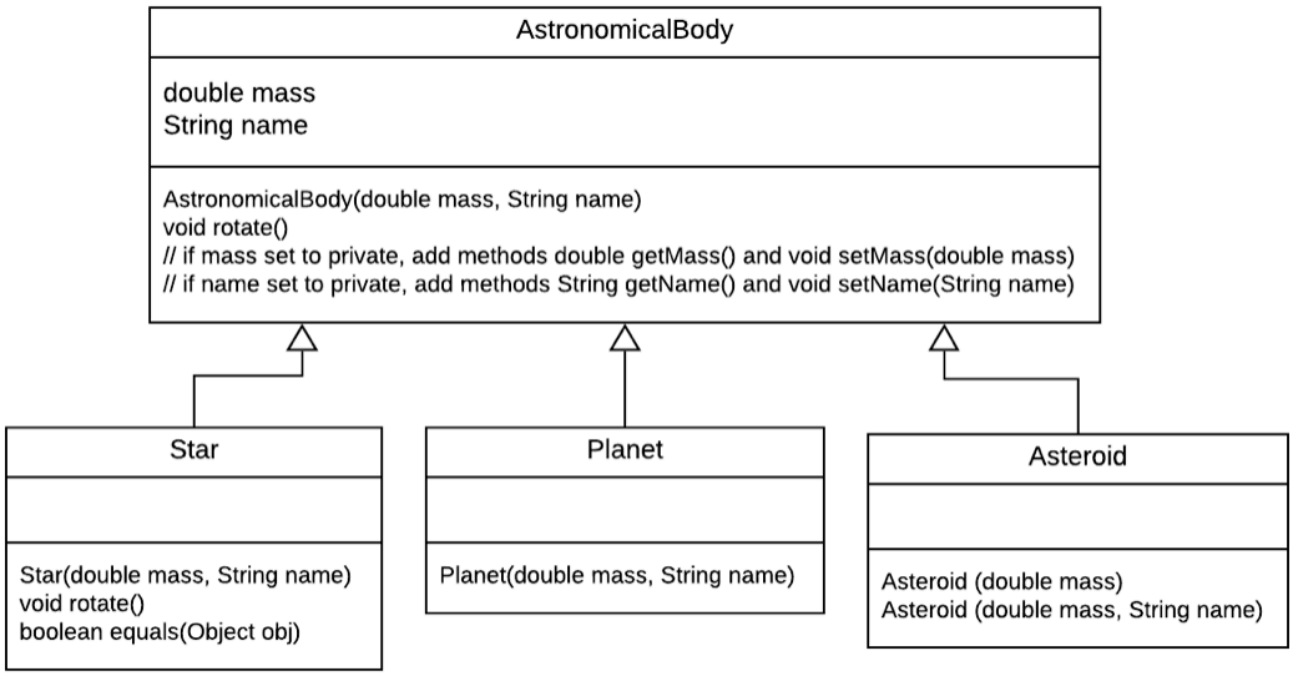
Please do not add any constructors, instance variables or methods that were not described here.

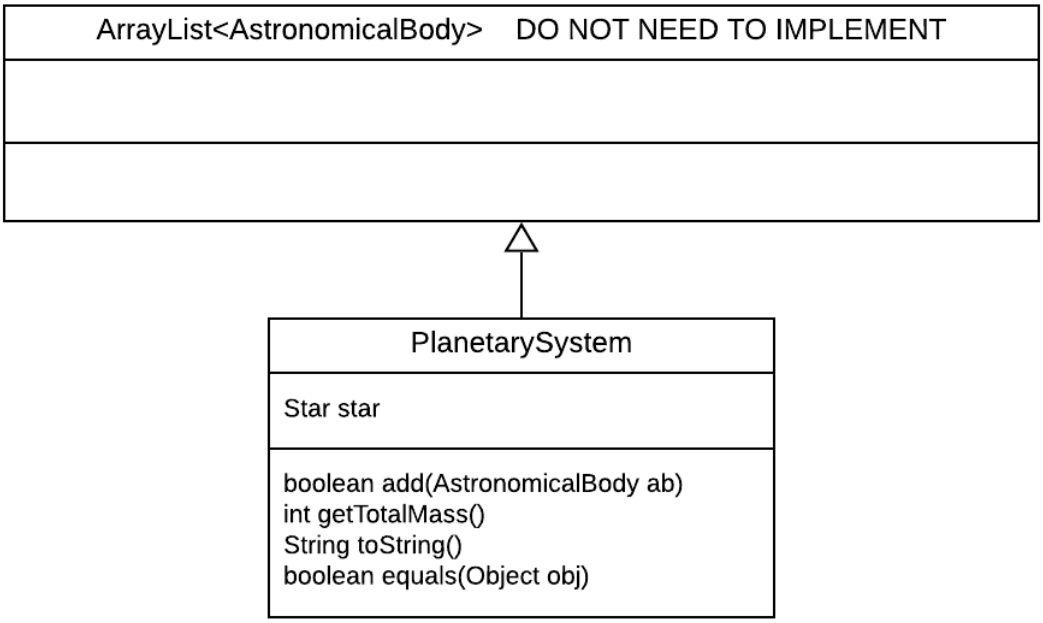
You should have 6 classes:

AstronomicalBody.java
Asteroid.java
Planet.java
Star.java
PlanetarySystem.java
PlanetarySystemTester.java

### UML diagrams

White arrows shows the inheritance. Inside are methods and instance variables that should be in these classes.
In each rectangle on diagram, the top row is class name, second section shows instance variables if any. The last section shows constructors and methods.






## Full description

**AstronomicalBody.java**
represents a general class for Asteroid, Planet, and Star.

It has two instance variables, double mass and String name. You can either set them to protected and access directly from subclasses OR you can set them private and add get and set methods for each instance variable.

It should have one constructor with two parameters and initialize mass and name of the AstronomicalBody

It should have one method rotate() which prints “Rotating around a star”

**Planet.java**
Is a subclass of AstronomicalBody. It does not introduce new instance variables or methods.

It should have one constructor with two parameters that calls superclass constructor

**Asteroid.java**

Is a subclass of AstronomicalBody. Asteroids have name, some do not. For example, asteroid Ceres has a name, but some random rocks in a space may not have a name. But they all have some mass.

Thus, Asteroid class will have two constructor.
First constructor will have two parameters, mass and name, and should call superclass constructor to initialize these variables.
Second constructor has only one parameter, mass. It should call superclass constructor and initialize mass with provided mass and set name as an empty string “”

**Star.java**
Star is a subclass of AstronomicalBody as well.

It should have one constructor with two parameters, similarly to Planet constructor.

Unlike Planet and Asteroid, Star does not rotate around a star, so you should override superclass method rotate, printing “Rotating around the center of a galaxy” instead.

Class Star should override method equals(Object obj). Two stars are equal if their names and masses are the same. Do not forget to compare string variables using equals method, not ==.

**PlanetarySystem.java**
Represents a planetary system, for example Solar system. It should extend ArrayList<AstronomicalBody>

No constructor!

It has one new instance variable Star star which represents a star of this planetary system (for the simplicity we assume that all planetary systems have only one star).

This class should override (enhance) the inherited **add(AstronomicalBody ab)** method.
1) If AstronomicalBody ab is not a Star object, then add it to PlanetarySystem as usually (call superclass method add) AND return true.
2) If the AstronomicalBody ab is an instance of Star class and the instance variable Star star was not set yet (which means that it equals null) then we add it to PlanetarySystem (call superclass method add) and set instance variable Star star to this star AND return true.
3) If the AstronomicalBody ab is an instance of Star class and the instance variable Star was already set to some other star (not a null) then we ignore this star and does not add it to this PlanetarySystem AND return false.

This class should have a new method, **getTotalMass()** which sums all the AstronomicalBody objects that this planetary system has.

This class should override **equals(Object obj)** method. Two planetary systems are equal if: 1) their stars (variable Star star) are equal AND 2) their total masses are equal AND 3) their size( ) (number of added AstronomicalBody) are equal.

This class should override **toString()** method that should return String that describes this planetary system. The details of this description is up to you.

**PlanetarySystemTester.java**
Should create 3 objects of PlanetarySystem type to test all the functionality.
You do not need to provide the real weight of planets and stars, just use some random and somewhat small values. Here's an incomplete example of main method for this class:

```
public static void main(String[] args) {
    Star sun = new Star(2000000, "Sun");
    Star aldebaran = new Star(3000000, "Aldebaran");

    Planet earth = new Planet(195, "Earth");
    Planet jupiter = new Planet(1234, "Jupiter");
    Planet aldebaranB = new Planet(4600, "Aldebaran b");
    Planet abc = new Planet(142, "Abc");

    Asteroid ceres = new Asteroid(42, "Ceres");
    Asteroid noName = new Asteroid(3);
    // you can create more AstronomicalBodies

    PlanetarySystem solarSystem = new PlanetarySystem();
    solarSystem.add(sun);

    // should not be added and should not replace
    // sun as solarSystem star
    // since solarSystem star was already set
    solarSystem.add(aldebaran);
    solarSystem.add(earth);
    solarSystem.add(ceres);
    // you can add more AstornomicalBodies
    // to your planetary systems

    // should call overridden toString
    System.out.println(solarSystem);

    // create two other PlanetarySystem
    // one should produce true when calling equals
    // method comparing with solarSystem
    // another should produce false
}
```

## Extra credit (5 points)

To receive extra credit you need to add java doc comments /** description */ to all classes, instance variables and methods (for methods describe all parameters in parameter list if any and what it returns if it is not void). Then, generate java documentation using Intellij tools to a

folder in your project. Double check that index.html works as expected showing all the information about your classes. Submit on GitHub with your solution.

## Submission

1. Push your project directory along with the source to GitHub by the due date.

2. Invite and share your project repository the Grader ( yan.chen01@sjsu.edu ) and Instructor ( **mariia.surmenok@sjsu.edu** ) or by usernames: **yanchen-01** and **msurmenok**

3. Submit a Readme.txt to Canvas including your name, repository access link, instructions to run your program (if any), **snapshot of your running program (screenshot of your output)** and citations (if any)

4. Your project directory will be graded according to the state your project directory was in at due time.