AP Computer Science Mr Hanley
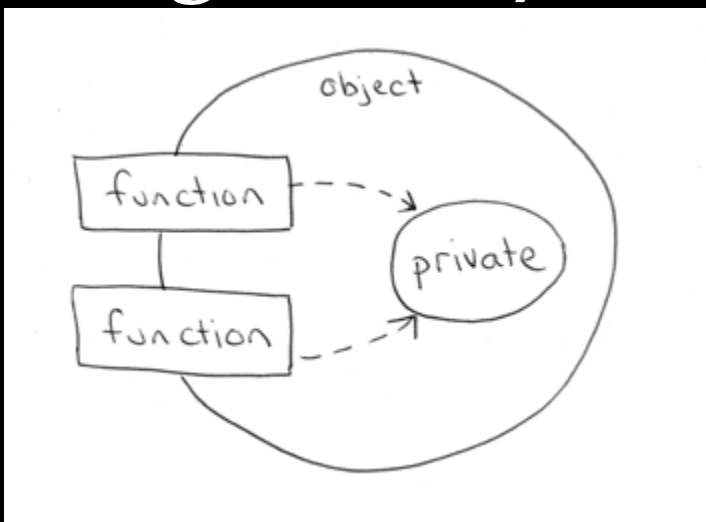
[The Hood](#)

```
Developing Classes

Ver: 3.111111119

Last Updated:11/16/2021 3:04 PM
```

Assignment 7:Developing Classes

Binary

Ones Comp

Twos Comp

In this project, we will be exploring object oriented programming by creating some simple classes and then writing tester classes to use them to create objects...let's go!!!!!!

Keep in mind in this paradigm shift there will be some differences in coding philosophy.

1.   OO classes will NOT have a main method, this is only for test classes

2.   OO classes will communicate via messages, by return types and parameters, they will NOT use System.out.println

3.   OO classes are written to be reused in the most general way possible

4.   Testers will use the public interface of the OO classes by calling constructors, mutators and accessors.

Phase 1: During Phase 1 We will be writing some classes and then a tester for each!

Make a new project called Assign7_OOPractice

- Let's start with a Car class.
  - a. Implement a class **Car** with the following properties. A car has a certain fuel efficiency (measured in miles/gallon or liters/km – pick one) and a certain amount of fuel in the gas tank. A Car also has a name which is typically the manufacturer and model like Chevy Trailblazer. The efficiency is specified in the constructor(one arg), and the initial fuel level is 0.

    Supply a method drive that simulates driving the car for a certain distance, reducing the fuel level in the gas tank, and methods getGas, returning the current fuel level, and addGas to tank up.
    Example usage.
    Car myBeemer = new Car(29); //29 miles per gallon
    myBeemer.addGas(20); //tank 20 gallons
    myBeemer.drive(100);
    System.out.println(myBeemer.getGas()); //print current fuel
    Must have a one arg constructor that takes in the fuel efficiency!
    If you want to have more constructors, you are encouraged to.

b. Write the CarTester class. This will have a main method and will create a couple of Car objects and then set their variables by calling addGas and drive.
   Use your zero arg and one arg constructor

Example output:
Here is the name:Honda Accord
Adding 5 gallons
Driving 100 miles…

Here is the new amount of gas 1.6666666666666665

- Write a **Student** class. (separate file in same project)
  For the purpose of this exercise, a student has a name and a total quiz score(points). Supply a one arg constructor that accepts the name and a default(zero arg) constructor that sets the name to "".
  Supply methods getName(), addQuiz(int score), getTotalScore() (this is the total points) and getAverageScore() (should return an int rounded to the nearest whole number. IF THE STUDENT HAS NO QUIZZES, RETURN -1 SO THE CLIENT PROGRAM CAN PRINT NO QUIZZES)) To Compute the average score, you will need to store the number of quizzes that the student took. (Don't worry how many points they are worth, just calculate the average).
  Supply a clear() method that clears the gradePoints and quizzes taken back to 0 but leaves the name the same.
  Usage
  Student s = new Student("Billy");
  s.addQuiz(70);
  s.addQuiz(84);
  s.addQuiz(100);
  System.out.println(s.getTotalPoints()); //prints 254

System.out.println(s.getAverageScore()); //prints 85
s.clear();
System.out.println(s.getTotalPoints()); //prints 0
System.out.println(s.getAverageScore()); //returns -1 so client can print NO QUIZZES OR SOME OTHER MESSAGE!!

    a. Make a **StudentTester** Class(just a main method and create a Student and run through the methods as above)

- Write a **Planet** class (models an INDIVIDUAL planet in our solar system)

A Planet in our solar system has a name, a number of earth days to orbit the Sun. If we indicate an initial position in the constructor, we should be able to predict a number of complete rotations around the Sun that the planets have made in "earth days". For example, if we initialize all 8 or 9 planets at day 1. By Day 500, some will have orbited the sun and some may still be working on their first orbit. For each planet, have them keep track of the number of days passed, the number of complete revolutions and the number of days towards the next revolution.

    Make a **PlanetTester** class that creates two planets, Earth and Mars and add 500 days of rotation to them. This is what should happen...

```
Planet p1 = new Planet("Earth",365.26);
Planet p2 = new Planet("Mars",686.98);
p1.addDays(500);
p2.addDays(500);
System.out.println("Earth Progress " + p1.getNumOrbits() + "
orbits " + p1.getDaysTowardNext() + " days towards next
orbit");
System.out.println("Mars Progress " + p2.getNumOrbits() + "
orbits " + p2.getDaysTowardNext() + " days towards next
orbit");
```

```
Output
Earth Progress 1 orbits 134.74 days towards next orbit
Mars Progress 0 orbits 500.0 days towards next orbit
```

- Write a class of your own choosing...pick some topic and name your class appropriately
  Ideas;

  Your class must include at least two pieces of data.
  You must have at least 2 constructors.
  Your class must have some sort of calculation or logical algorithm similar to the Employee, Car, Student and Planet.

  Suggestions;
  Violin: has a maker, date manufactured and condition.
  getValue is some combination of how old, the maker and condition

  Product for Sale: has a price and description.
  Has a method for discounting depending upon certain conditions

  Bank Account: Has a balance and an owner name
  Has a method for computing monthly interest based on balance and rate(monthly interest increases the balance when it is applied)

  College Admission: Has a percentage of students allowed in out of state and in state. Can compute different tuitions for out of state and in state.

Cylinder: Has a radius and length.
Can compute volume, can increase and decrease in size.
Given a desired volume and radius, can set up length.

Pizza: Can have up to 5 toppings. Can return true or false if it is Vegetarian friendly by observing toppings. Can specify specialty requests in constructor. For example, new Pizza("MeatLovers") would result in the toppings being automatically assigned. Same for new Pizza("Chicken Bacon Ranch").

Risk Game Territory: Based on the traditional board game, one of the territories you can conquer. Has a name, associated Continent and between 1-5 adjacent territories. Unsure of what methods would be???

   a. Write a Tester for your class

# Phase 2: Add fancy heading sections in each of your classes

Step1: Get your NetBeans IDE editor ready by setting up some spicy code templates. This is found under Tools...Options Editor, Code Templates
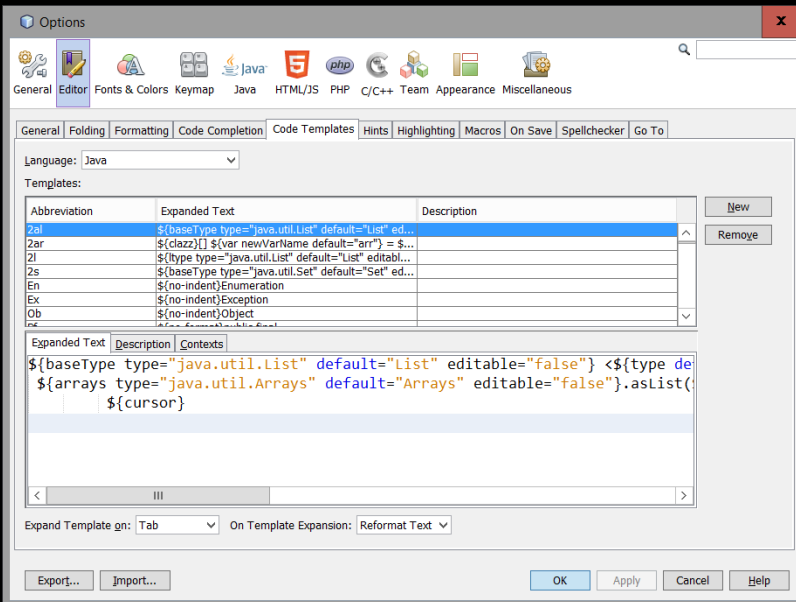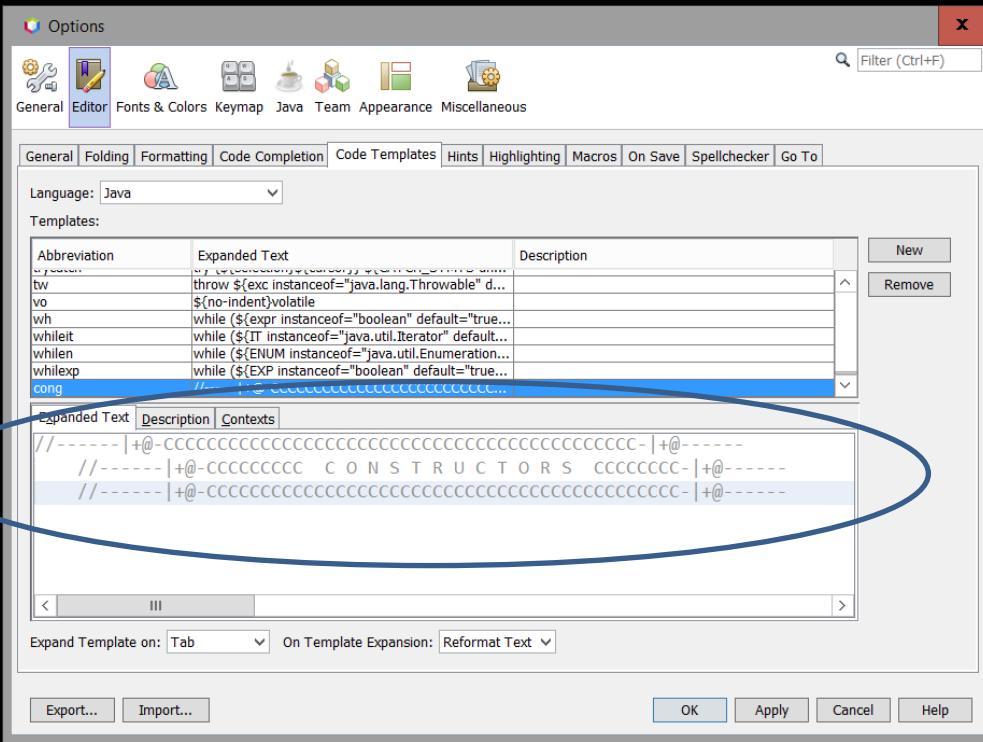
We will be using code templates for **instance variables, constructors, constants, mutators, accessors and static variables.**

Let's start with instance variables, sometimes called fields.

Click on the new button and let's create a code template for **ins**
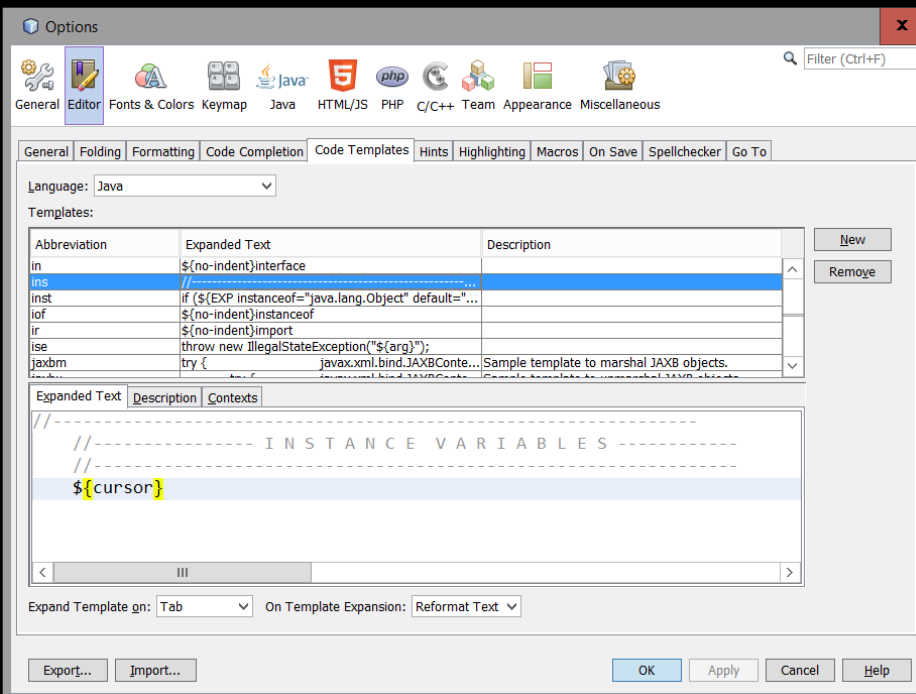
When you do it, TAB the second two rows like this example

```
//----------------------------------------------------------------
//------ I N S T A N C E   V A R I A B L E S / F I E L D S --------
//----------------------------------------------------------------
```

Constructors Heading->

Click on the new button and let's create a code template for con

```
/////////////////////////////////////////////////////////////////
/////////////////    C O N S T R U C T O R S     ////////////////
/////////////////////////////////////////////////////////////////
```

Constants Heading ->

Click on the new button and let's create a code template for const

```
//ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
//ccccccccccccc         C O N S T A N T S      ccccccccccccccccccc
//ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

Mutators Heading ->

Click on the new button and let's create a code template for mut

```
//MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
//MMMMMMMMMMMMMMM         M U T A T O R S    MMMMMMMMMMMMMMMMMMMMMM
//MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
```

Gangnam Style Constructor Heading ->(don't ask, just go with it!)

Click on the new button and let's create a code template for cong

```
//------|+@-ccccccccccccccccccccccccccccccccccccccccccccccc-|+@-----
```

```
//------|+@-CCCCCCCCC  C O N S T R U C T O R S  CCCCCCCC-|+@------
//------|+@-CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC-|+@------
```

Gangnam Style Mutator Heading ->

Click on the new button and let's create a code template for mutg

```
//<M><M><M><M><M><M><M><M><M><M><M><M><M><M><M><M><M><M><M><M><M>
//<M><M><M><M><M><M>        M U T A T O R S        <M><M><M><M><M><M>
 //<M><M><M><M><M><M><M><M><M><M><M><M><M><M><M><M><M><M><M><M><M>
```
Accessors Heading ->

Click on the new button and let's create a code template for acc

```
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAA  A C C E S S O R S   AAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```
Statics Heading ->

Click on the new button and let's create a code template for sta

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//++++++++++++   S T A T I C  V A R I A B L E S   +++++++++++++++
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

System.out.print template ->
While we're at it, isn't it a pain in the bootie that you can't use a template for System.out.print ? You have to use the System.out.println and then erase the ln!

Not any more, add a so **code template, you know what to do, make it System.out.print();**
taking a look at the System.out.println() template I learned something….System.out.print("${cursor}");
//They put a ${cursor} inside the " " Do you know why they did this?

I do, it's a macro to tell NetBeans to place the cursor inside the " " so you can actually type whatever you want to print out immediately!! How cool is NetBeans!!!!!!!!!!!!!

Make sure all 4 classes, Car, Student, Planet and whatever your custom class looks similar to this;

```
/**
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * S-h-e-n-e-n-d-e-h-o-w-a--H-i-g-h--S-c-h-o-o-l--T-e-c-h-n-o-l-o-g-y--D-e-p-t
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 * FILE:        Employee.java
 * DATE:        Nov 3, 2014 Original
 * AUTHOR:      mr Hanley
 * VERSION:     2.1
 * PURPOSE:     Employee problem from Big Java
 *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 *
```

```java
 * m-r-h-a-n-l-e-y-c-.c-o-m~~~~~~~~~~t-e-a-m-2-0-.-c-o-m~~~~~~~~~~~~~~~~~~~~~~
 */
import java.text.DecimalFormat;

public class Employee {

    //----------------------------------------------------------------
    //-------------- I N S T A N C E   V A R I A B L E S --------------
    //----------------------------------------------------------------
    private double salary;
    private String name;

    ////////////////////////////////////////////////////////////////////
    ////////////////////////    C O N S T R U C T O R S    //////////////////////
    ////////////////////////////////////////////////////////////////////

    public Employee() {
        name = "";
    }

    public Employee(String na, double sal) {
        name = na;
        salary = sal;
    }

    //AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    //AAAAAAAAAAAAAAAAAA          A C C E S S O R S          AAAAAAAAAAAAAAAAAAAA
    //AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

    public String getName() {
        return name != null ? name : "                  ";
    }

    public double getSalary() {
        return salary;
    }

    //MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
    //MMMMMMMMMMMMMMMMMM          M U T A T O R S          MMMMMMMMMMMMMMMMMM
    //MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM

    public void setName(String n) {
        name = n;
    }

    /**
     * setSalary preconditions: sal >0
     *
     * @param sal salary which must be 0 or positive
     *
     */
    public void setSalary(double sal) {
        if (sal >= 0) {
            salary = sal;
        }
    }

    /**
     *
     * @param byPercent a number > 0 which represents the raise to annual salary
```

```
    */
    public void raise(double byPercent) {
        salary += salary * byPercent / 100;

    }

}
```

Phase 3: Unit Testing: Did you know that Junit is integrated into NetBeans and you can make a Junit test for each class that you create?

Make a Junit class for each of the Car, Student, Planet and Your Custom Class. See Mr. Hanley's website for examples.

1. # Phase 4: Integrate your classes into the Moosk and demonstrate all 4 to your teacher.

Phase 5: Integrate the IntVerifier, DoubleVerifier and BoxPrint into your assignment 7.

**The DRY principle of computer science says DON'T REPEAT YOURSELF!
You may be tempted to copy your work from the Verifier assignment into this project but then you will have 2 copies of it and then have to manage both of them.**

   a. **Go back to your Verifiers project and open up your boxPrint logic. Change the limitation on print size to 100 so you can boxPrint longer phrases.**
   b. **Go back to Assign7_OOPractice project and choose Project…Project Properties**
   c. **Click on the Libraries TreeList Option**
   d. **Click on Add JAR/Folder and navigate to your Verifiers project**

e. **You need to click on the build/classes folder of your Verifiers project(this arrow points to where you add JAR/Folder**
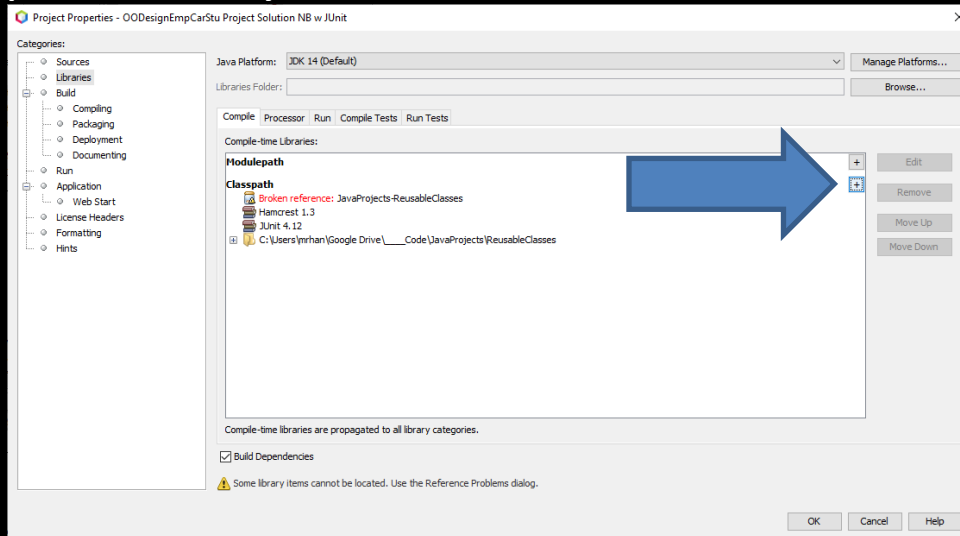
2. **Now your Assignment 7 tester class can utilize IntVerifier and DoubleVerifier**

3. **Go to the menu method of the moosk (EmpCarStuClientTemplateV2) and add an option 0 to print out your name and the period you have java**
   You should be able to do CWHUtilites.boxPrint("Mr. Hanley Period 5");

4. **Let's use your IntVerifier to make sure that the main menu choice is in an acceptable range.**

   i. **Make a global variable for an error sound effect so we can use it throughout the program.**
      Load up a sound file like Explosion.wav using the cookbook inside the menu method.

   ii. **Make an IntVerifier called mainMenuVer using the sound file and give the appropriate range (remember we print our name and period we have class with option 0).**

   iii. **Pass the Scanner object to the IntVerifier so it can use your scanner to read the keyboard.**

   iv. **Comment out the input.nextInt() command**

   v. **Replace it with your mainMenuVer.readInt() command to gather in the main menu input to select an option**

vi. **Create a DoubleVerifier for reading in the number of earth days to orbit the sun when recreating a planet**
Use a range of 1-1000000 (1 million).

vii. **Create an IntVerifier for reading in the distance you are going to drive when you prompt the user in the moosk for how far the user wants to drive their vehicle.**
Use a range of 0 (exclusive) to 1000 miles(inclusive)

Project Name	Assign7_OOPractice
Class 1 Name	Car
Class 2 Name	CarTester
Class 3 Name	Student
Class 4 Name	StudentTester
Class 5 Name	Planet
Class 6 Name	PlanetTester
Class 7 Name	Depends upon Custom Class, could be Violin, Product, Cylinder, RiskTerritory
Class 8 Name	ViolinTester or CylinderTester
Class 9 Name	Car_Junit1
Class 10 Name	Student_Junit1
Class 11 Name	Planet_Junit1
Class 12 Name	Violin_Junit1 or Cylinder_Junit1
Class 13 Name	EmpCarStuClientTemplateV1.java

RUBRIC

Rubric Car	
2 Constructors	10
drive() method	10
addGas()	10
Comment headers	5
CarTest logic	10
TOTAL	45

RUBRIC

Rubric - Student	
2 Constructors (0 arg and 1 arg) more OK	10
addQuiz()	10
getTotalScore()	10
getAverageScore()	5
clearGrades()	10
Student Test logic	10
Comment headers	5
TOTAL	60

RUBRIC

Rubric - Planet	
O arg constructor, 2 arg constructor for name and daysToRevolve around sun	10
addDays(correctly computes the number of revolutions and extra days towards next	25
Integrate into Moosk	20
Comment headers	5
TOTAL	60

RUBRIC

Rubric – Custom Class	
Has at least two constructors	10
Has some math logic associated	15
Integrated into Moosk with menu and 3 objects	20
Comments	5
TOTAL	50

RUBRIC

Rubric – Using Verifiers	
boxPrinting the Author and Period you have AP Java	10
IntVerfier for Main Menu	20
Use DoubleVerifier Somehow	20
TOTAL	50