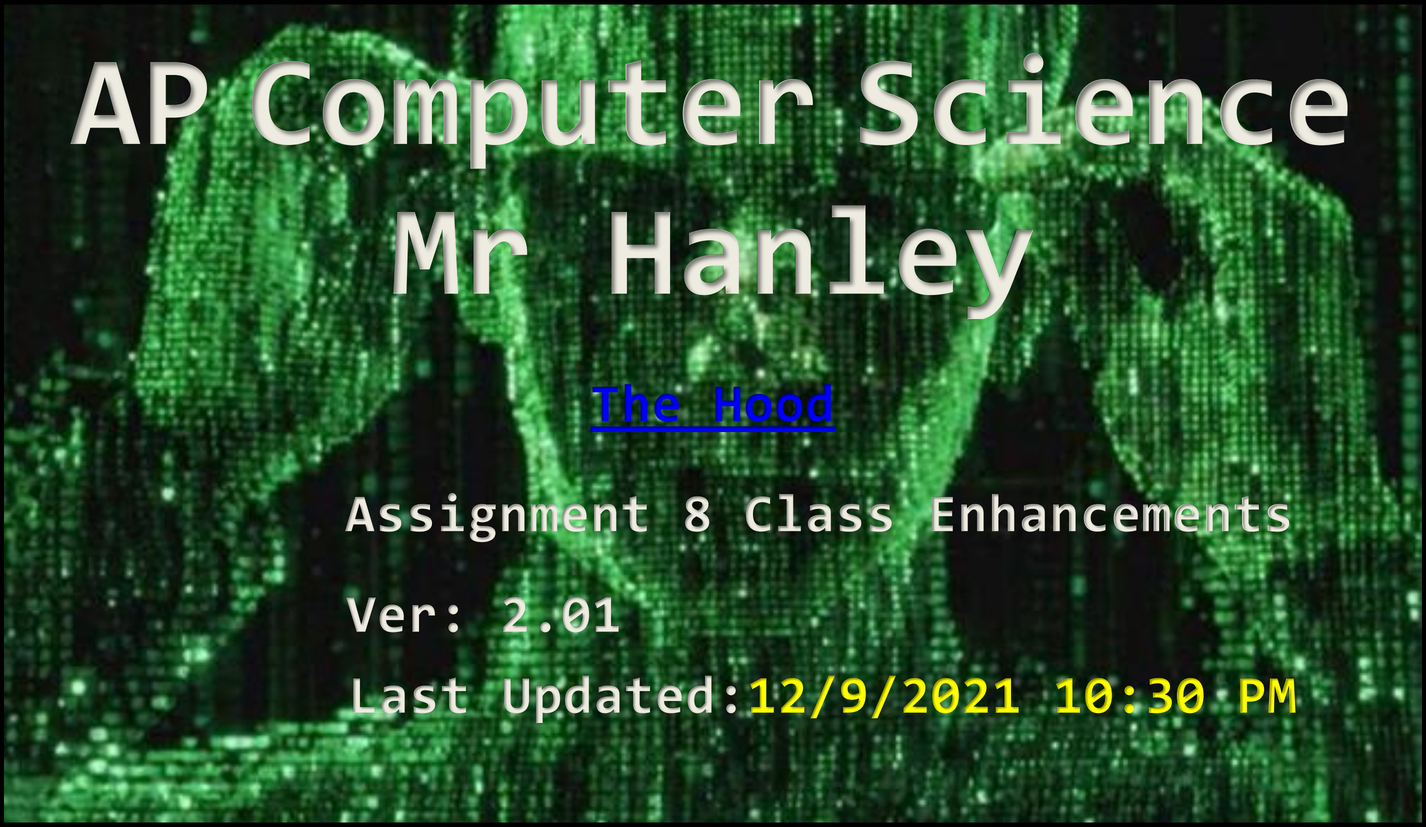


AP Computer Science

Mr Hanley

The Hood

Assignment 8 Class Enhancements

Ver: 2.01

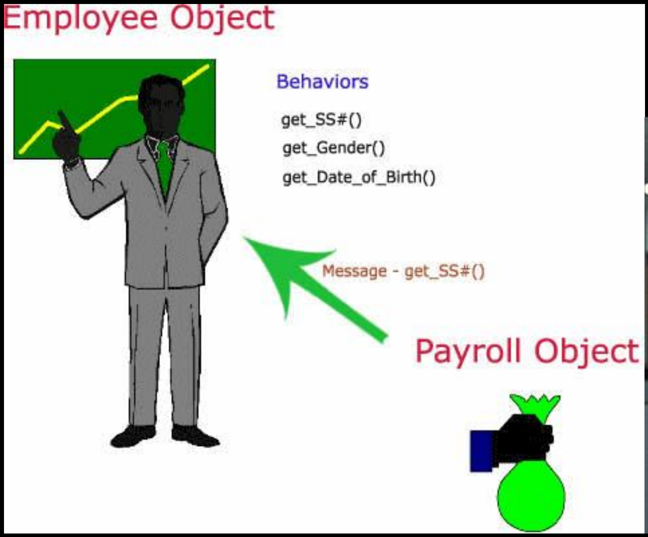
Last Updated:12/9/2021 10:30 PM

Assignment 8:Class Enhancements

Binary ⬜⬜⬜⬜⬜⬜🟥⬜

Ones Comp 🟥🟥🟥🟥🟥🟥⬜🟥

Twos Comp 🟥🟥🟥🟥🟥🟥🟥⬜



1. **The toString method is inherited from Object, which all java classes are descendants from. Redefine toString for the classes you developed to display the information from each object. Test the methods using your test classes.**
   Here are some example toString calls for Car, Student, Planet and Activity

```
Testing Car toString()...
Name: Honda Accord    MPG: 30.0  Fuel: 1.6666666666666665

Testing Student toString()...

________________________
Shikhar
total points:    272
total quizzes    3
average:   90

________________________
Testing Planet toString()...
Earth                          365.26                    4                239.0
```

2. Class variables are used to keep data that is shared among all instances of a class. MUST BE PUBLIC! Add class variables to each of your classes.

   This example shows how to use a static variables in an employee class where each Employee has a name and a salary.
   It also demonstrates toString and throwing exceptions.

```
public class Employee_with_assign8_changes {
 //%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 //---------------- S T A T I C   V A R I A B L E S ---------------
 //%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 //static variable to keep track of total employees and salaries
 public static double totSal = 0;
 public static int totEmp = 0;
 //-----------------------------------------------------------------
 //---------------- I N S T A N C E V A R I A B L E S  ------------
 //-----------------------------------------------------------------

 private double salary;
 private String name;

 //------|+@-CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC-|+@------
 //------|+@-CCCCCCCCC  C O N S T R U C T O R S  CCCCCCCC-|+@------
 //------|+@-CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC-|+@------
 public Employee_with_assign8_changes() {
     totEmp++;
 }
 public Employee_with_assign8_changes(String na, double sal) {
     //Increase the total number of employees
     totEmp++;
     //Increase the payroll
     totSal = totSal + sal;

     name = na;
     salary = sal;
 }
 //AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
 //AAAAAAAAAAAAAAAAA       A C C E S S O R S      AAAAAAAAAAAAAAAAA
 //AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

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

```
    public String toString() {
        DecimalFormat df = new DecimalFormat("###,###,###");
        String salDisplay = df.format(salary);
        salDisplay = "$"+salDisplay;
        return "Name  :" + name + "\nSalary:" + salDisplay;
    }

    ////<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>

    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) throws IllegalArgumentException {
        double oldSal = salary;

        if (sal >= 0) {
            salary = sal;
            totSal -= oldSal;  //subtract old salary from total salary
            totSal += salary;  //add in their current salary

        } else {
            throw new IllegalArgumentException("Can't have negative salaries");
        }
    }
```

...more stuff not shown

```
public class EmployeeTest1
{
  public static void main(String[] args)
  {
    Employee_with_assign8_changes e1 = new Employee_with_assign8_changes("Pat LaCourse",1e5);
    System.out.println(e1);

    Employee_with_assign8_changes e2 = new Employee_with_assign8_changes("Susan 
Schwarz",2.5e5);
    System.out.println("Total Employees:"+Employee_with_assign8_changes.totEmp);
    System.out.println("Total Payroll:"+Employee_with_assign8_changes.totSal);

    try {
    e1.setSalary(-10);
    }
    catch(Exception e) {
      System.out.println(e.toString());
    }
  }
}
```

```
Output
------
Name  :Pat LaCourse
Salary:$100,000
Total Employees:2
Total Payroll:350000.0
java.lang.IllegalArgumentException: Can't have negative salaries
```

Suggestions for appropriate class variables:
Car: **make a static for the totalMiles driven for each car....create two cars and then drive each one. Print out your static variable from your tester and make sure the total miles are added in.**

Student: **Make two static variables totalPointsAllStu and totalQuizzesAllStu. Keep track of all points earned for all students and all quizzes taken for all students. Print out the class average. Don't forget when clearing a students grades to subtract their scores from the totalPointsAllStu as this impacts the class variable. Same with totalQuizzesAllStu.**

Planet: **If we have 8 (or 9) planets in our solar system, there is no need to store the number of days that have passed in the simulation 8 or 9 times. Make the daysPassed a static so all planets share this variable (addDays() however now cannot update this variable since we call addDays() for each planet.**
Here is how I handled this issue:
From my tester(I allow the tester to manually update the static)

```
System.out.println("Please enter in a number of dayz to add");
double dayz = input.nextDouble();

Planet.currentEarthDay = Planet.currentEarthDay + dayz;
```

Custom: **Think of one or two statics for you custom class-> print them out from your tester with tester by <YOURCUSTOMCLASS>.staticVarName**

Test from the test classes.

NOTE: although these are not covered on the ap exam, you can write a method for a class that will get called right before the object is destroyed or garbage collected.

```
public void finalize()
{
    //reduce object count here
}
```

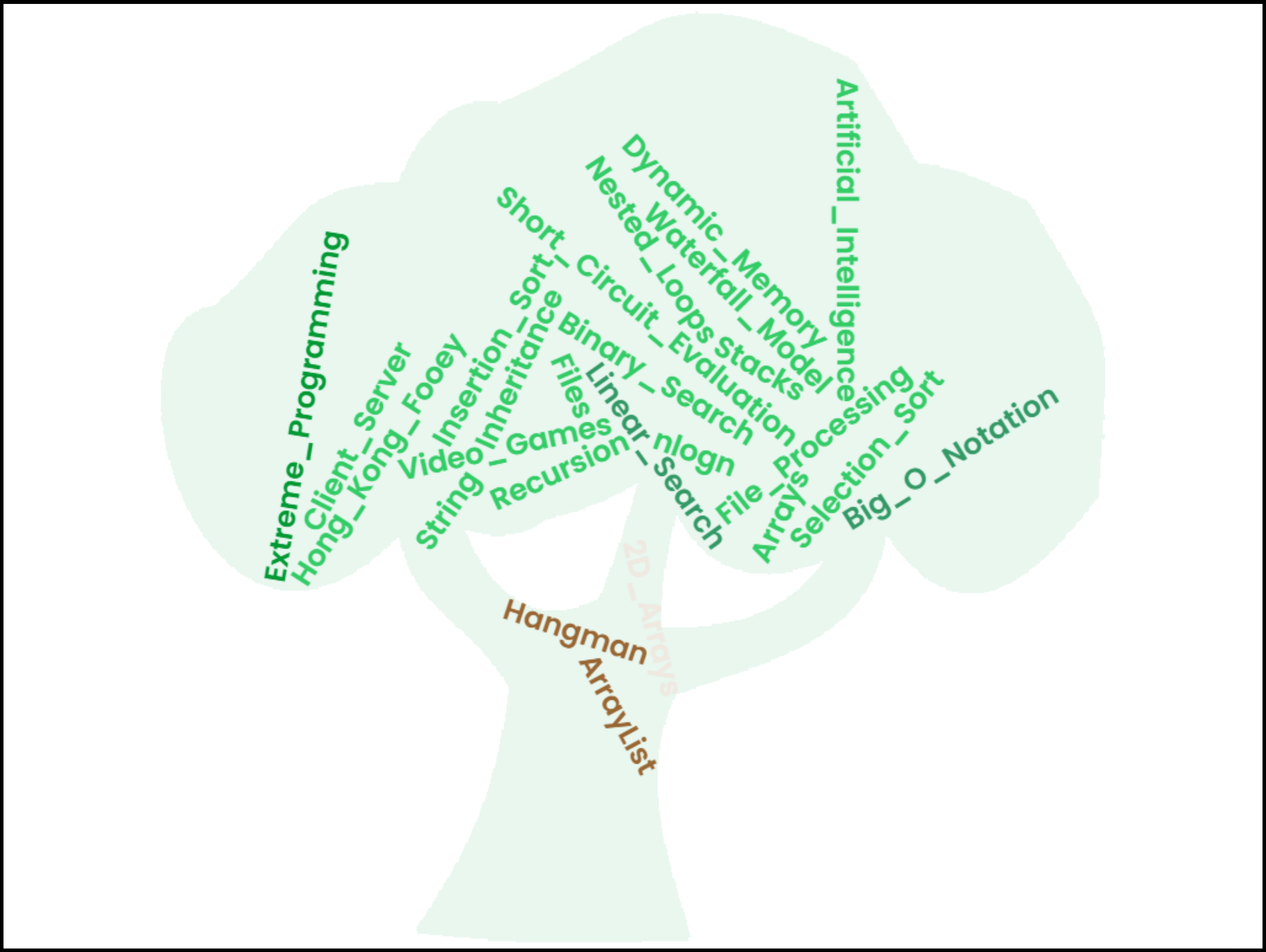
UPDATE: 11/5/2015: Unfortunately finalize does not get called with any certainty and you cannot force it to be called. So when looking at total salaries for example, they will not necessarily get reduced even though a reference goes out of scope☹

3. Add exception handling to each of your classes.
  Pick a mutator method and have it throw an exception when someone tries to set a value that is out of range. See the setSalary for Employee above.
  Add a try catch block to your tester so it will catch the IllegalArgumentException

4.

Project Name	**Reusing Assign7_OOPractice**
Class 1 Name	Modifying Existing Car, Student, Planet and Custom
Class 2 Name	Modifying Tester classes

RUBRIC	
Redefine toString in 4 classes	20
Test toString	5
Define class variables in 4 classes 1 for Car, total miles driven 2 for Student totalPointsAllStu and totalQuizzesAllStu 1 for Planet daysPassed At least one for Custom Class	10
Modify 1 setter or Constructor in each class to throw an exception	20
Test exceptions	10
Comments	15
TOTAL	110


Artificial_Intelligence
Dynamic_Memory
Waterfall_Model
Nested_Loops
Stacks
Short_Circuit_Evaluation
Extreme_Programming
Client_Server
Hong_Kong_Fooey
Video_Games
Insertion_Sort
Inheritance
String
Binary_Search
Files
Linear_Search
Recursion
nlogn
File_Processing
Arrays
Selection_Sort
Big_O_Notation
2D_Arrays
Hangman
ArrayList