# AP Computer Science

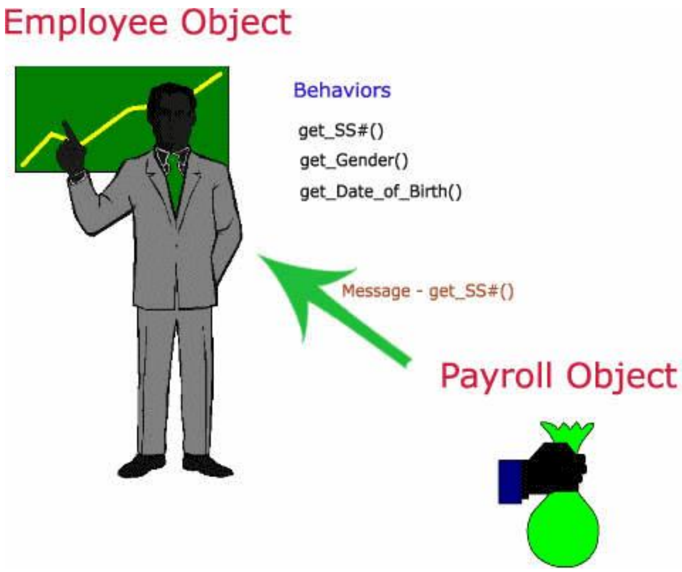
## mr Hanley

## Assignment $8/1000_2/10_8/8_{16}$ Version: 2.0

## Last Updated: 1/20/2019 11:00 PM

| Binary | Ones Comp | Twos Comp |
|---|---|---|
| □□□□■□□■ | ■■■■□■■□ | ■■■■□■■■ |

## Class Enhancements



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.

**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. **Keep track of the total number of employees ever created AND the total salary for all employees**. Each time a constructor is called, update the class variables. Have a method to print out the totals to the screen. Do the same for the car (maybe total miles driven) and student (cumulative average

* Dy mic Memory *Big O Notation*Stacks *Extreme Programming*Selection Sort*Insertion Sort*Waterfall Model

String*Arrays*ArrayList*Client Server*Artificial Intelligence*Inheritance*Files*Video Games*Short circuit evaluation* might require 3 statics??? )classes. Test from the test classes.

But how do I get access to a class variable?

```
public class EmployeeTest {
    public static void main(String[]args) {
        System.out.println(“Before adding employees”);
        System.out.println(“Total Emp” + Employee.totalEmployees);
        System.out.println(“Total Salaries” + Employee.totalSalary);
        Employee joe = new Employee(“Ian Fisher”,52000);
        Employee joe = new Employee(“Bobby Fisher”,102000);
        System.out.println(“After adding employees”);
        System.out.println(“Total Emp” + Employee.totalEmployees);
        System.out.println(“Total Salaries” + Employee.totalSalary);
    }
}
```

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
}
```

Optional: add the finalize method to your classes to adjust your total salary and employee count

**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 nec get reduced even though a reference goes out of scope☹

3. Add exception handling to your classes. Throw illegal argument exceptions if the parameters are illegal. Pick one of your mutator methods for each class and modify it so that it throws an IllegalArgumentException. Modify your test classes to use a try catch block to handle the exception.
   Below is an example;

```
//From Employee.java file
public void setSalary(double sal) throws IllegalArgumentException
 {
   if (sal>=0)
   {
     salary = sal;
   }
   else throw new IllegalArgumentException("Can't have negative salaries");
```

```
    }

    //From EmpTest.java file
    public class EmpTest
    {
     public EmpTest()
     {
      Employee e1 = new Employee();
      System.out.println(e1.getEmployees());
      try
      {
       e1.setSalary(-10);
      }catch(IllegalArgumentException ia)
      {
       System.out.println(ia);
      }
}
```

| | |
|---|---|
| **Project Name** | **Use your existing project with classes** |
| **Class 1 Name** | Employee(given to you) |
| **Class 2 Name** | Car |
| **Class 3 Name** | Student |
| **Class 4 Name** | Planet |
| **Class 5 Name** | <YOUR UNIQUE CLASS> |
| **Class 6 Name** | Moosk Updated |

| **Rubric** | |
|---|---|
| **Redefine toString in 4 classes** | **20** |
| **Test toString** | **5** |
| **Define class variables in 4 classes (2 for employee<br>1 for salary, 1 for num employees)<br>1 for Car, total miles driven<br>2 for Student totalPointsAllStu and totalQuizzesAllStu<br>1 for Planet daysPassed** | **20** |
| **Test class vars** | **20** |
| **Modify 1 setter or Constructor in each class to throw an exception** | **5X4 = 20** |
| **Test exceptions** | **10** |
| **Comments** | **15** |

| | |
|---|---|
| **TOTAL** | **110** |

***Recursion*Linear Search*Binary Search*Grid World Case Study*File Processing *nlogn*Hangman***