Lab: Objects

In this lab, you will create a page named "objects.jsp" that has the same layout as your home page (index.jsp) and you will create your own class. **Requirements for your class (see page two for sample class code):**

1. Your class shall have at least 3 private data members, including at least 1 string & at least 1 number (int or double).
2. **Methods:** last week (methods lab), all of your methods were static methods which means that the methods were not associated with any object. Essentially they were just a collection of functions that had everything they needed provided by input parameters. This week, we are writing classes that will have data members associated, classes for which we must instantiate objects before we can use them. So, in your class, **all the methods will not be static.**
   a. Your class shall have at least two constructor methods: one will be the default constructor (the constructor that will be invoked when an object is instantiated without any input parameters) and a second constructor that takes at least one input parameter. Remember that a constructor has the same method name as the class itself and there is no return type specified – this is because a constructor actually returns the instantiated object.
   b. Each private data member shall have a public getter and a setter method (see page two for examples).
   c. Your class shall have a method that alters data members in a more complicated way than a normal setter method (for example, the changeSalary method in the Employee class on the next page). This method shall “make sense” within the context of the class you decided to implement. The comments of this method shall clearly explain what calculation is being performed.
   d. Your class shall have a “showAll” method that returns a string that labels and displays values of all data members (also as shown in the Employee example). **Numeric values shall be nicely formatted.**
   e. **OPTIONAL:** Add a method that checks to see if data members of one object have the same values as the data members in another object (it compares “this” object to some other object, as shown in the sample code below). This method will return type boolean which means it will return either true or false.
   f. **OPTIONAL:** Add a method that copies from one object to another object (it copies all of “this” object’s data members to some other object, as shown in the sample code below).
3. You may select any kind of class you like, but try to come up with a class that represents the type of product that your web site will be reviewing. You may not create a class like the example code zip files or any code presented in the powerpoints.
4. Your package, your class, and your data members shall be named using good programming style (meaningful names). Your getter and setter methods shall follow naming conventions (as shown in the Employee example below). All classes shall start with a Capital letter (and CamelCase). All variable and object names shall start with a lower case letter (camelCase). Class data members shall not start with a type designation (e.g., str, int, dbl), but other variables shall start with a type designation.

**objects.jsp and labs.jsp:**

5. objects.jsp shall declare and instantiate 2 objects of your class, then demonstrate that all methods of your class work properly (so, your objects.jsp page shall perform an exhaustive test of all the methods in your class). You may just “hard code” test values as demonstrated in the testEmployee.jsp page, as shown below – you are not required to get user input.
6. **Before each invocation of a method that changes a value of an object, objects.jsp shall display on the jsp page what you are doing (e.g., ‘setting the salary of the boss to $20,000’).** After each invocation of a method that changes a value of an object, objects.jsp shall display (on the jsp page) the result of the showAll method on that object. For educational reasons (so you truly understand that changes in the data of one object do not affect the data of the other object), objects.jsp shall alternate between the two objects as you modify properties and invoke the showAll method (as was done in the sample jsp page below).
7. Objects.jsp shall display (in the browser) an explanation of the calculations of your method that alters data members in a more complex way than a normal setter. This helps us to grade your work.

8. You should already have a page called "labs.jsp" that is referenced by your nav bar and that has the same layout as your index.jsp page. To the top of your labs.jsp page add a blog entry for this week’s work, briefly describing the work you did this week. Into this blog entry, add a link to objects.jsp

9. Publish your web application using the publishing instructions for web apps WITH classes. Test published code, then attach zip file to BB by the due date.

Employee.java

```java
package myPkg;
import formatPk.*; // you need this line, if you want to use code from another package/class

public class Employee {
    // Data members, also called instance variables.
    private String name = "";
    private String title = "";
    private double salary = 0.0;

    // This method is called the constructor method (no return value, has same name as class).
    // It's purpose is to initialize the object...
    public Employee(String myName, String myTitle, double mySalary) {
        this.name = myName;
        this.title = myTitle;
        this.salary = mySalary;
    }

    // This is an overloaded constructor. When you call the constructor with no inputs,
    // you get an empty employee record (Default values as shown above, e.g., "" and 0)
    public Employee() {
    }

    public String getName() { // returns a String telling the employee's name
        return this.name;
    }

    public String getTitle() { // returns a String telling the employee's title
        return this.title;
    }

    public double getSalary() { // returns a double (real number) telling the employee's salary
        return this.salary;
    }

    public void setName(String newName) {
        this.name = newName;
    }
}
```
public void setTitle(String newTitle) {
    this.title = newTitle;
}

public void changeSalary(double pctChange) { //0.03 is 3% increase, negative is decrease.
    this.salary = this.salary * (1.0 + pctChange);
}

public String showEmployee() {
    return "Employee Name is " + this.name
    + "<br/>Employee Title is " + this.title
    + "<br/>Employee Salary is " + FormatUtils.formatDollar(this.salary) + "<br/>";
}

// copy all the data members of this Employee to destEmployee
public void copyTo(Employee destEmployee) {
    destEmployee.name   = this.name;
    destEmployee.title  = this.title;
    destEmployee.salary = this.salary;
}

// return true only if all the data members of otherEmployee
// have the same value as this Employee.
public boolean isEqual(Employee otherEmployee) {
    if (this.name.equalsIgnoreCase(otherEmployee.name) &&
        this.salary == otherEmployee.salary &&
        this.title.equalsIgnoreCase(otherEmployee.title))
        return true;
    else
        return false;

    // Could have also done this - the body would be a single return statement:
    //  return (this.name.equalsIgnoreCase(otherEmployee.name) &&
    //       this.salary == otherEmployee.salary &&
    //       this.title.equalsIgnoreCase(otherEmployee.title));
}
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page language="java" import="myPkg.Employee" %>

<!DOCTYPE html>
<html>
<head>
</head>
<body>
<h1>Using Employee Objects</h1>%
  // Declare and initialize two Employee objects
  Employee emp1 = new Employee("Freda Jones", "Accountant", 45000.00);
  Employee emp2 = new Employee("Mark Smith", "Bookkeeper", 25000.00);
  out.println("<h3>Two employee objects have been created.</h3>");

  out.println("<h3>Employee 1 is </h3>" + emp1.showEmployee());
  out.println("<h3>Employee 2 is </h3>" + emp2.showEmployee());

  emp1.setName("Freda White");
  out.println("<h3>Changed the name of employee 1 to Freda White. Employee 1 is now </h3>" + emp1.showEmployee());

  emp2.setTitle("Salesman");
  out.println("<h3>Changed the title of employee 2 to Salesman. Employee 2 is now </h3>" + emp2.showEmployee());

  emp1.changeSalary(0.03);
  out.println("<h3>Just gave employee 1 a raise of 3 percent. Employee 1 is now </h3>" + emp1.showEmployee());

  emp2.changeSalary(-0.04);
  out.println("<h3>Just gave employee 2 a paycut of 4 percent. Employee 2 is now </h3>" + emp2.showEmployee());

  out.println("<h3>Do the two employees have the same data members? </h3>" + emp1.isEqual(emp2) + "</h3>");

  emp1.copyTo(emp2);
  out.println("<h3>Just copied employee 1 to employee 2. Employee 2 is now </h3>" + emp2.showEmployee());

  out.println("<h3>Do the two employees have the same data members? </h3>" + emp1.isEqual(emp2) + "</h3>");%
</body>
</html>
APPENDIX - How to Add a Class to a Package to a NetBeans Web App

In the Netbeans project pane (upper left of screen),

- right click on “source packages” and select “New – Java Package”. Call it what you like, but give it a representative name. The package below is called “myPkg”.

- Then right click on the new package and select “New – Java Class” (again, give it a representative name, classes are supposed to start with a capital letter according to naming convention). There are several classes below, Circle, Employee, and FormatUtils.

- Then (inside your new class file), enter the code. In the example below, the circle class has a private data member (radius), and several methods. Note that you do not use `static` when declaring methods for this “objects lab”. 

```java
package myPkg;

public class Circle {
    final double PI = 3.14;
    private double radius;

    public Circle (double myRadius) {
        this.radius = myRadius;
    }

    public void setRadius(double myRadius) {
        this.radius = myRadius;
    }

    public double getArea() { // Area of a cir
        return this.radius * this.radius * PI;
    }

    public double getCircumference() { // Circ
        return this.radius * 2.0 * PI;
    }
}
```
• In your JSP page, you must use an import statement (see line 2 below) or else you will not be able to reference any methods of your class. For example,

<%@page language="java" import="myPkg.Circle" %>

```jsp
<%@page content-type="text/html" page-encoding="UTF-8" %>
<%@page language="java" import="myPkg.Circle" %>
<!DOCTYPE html>

<body>

%%

String strCircle1Radius = "";
String strCircle2Radius = "";

String circle1Msg = "";
String circle2Msg = "";
String circleEqualMsg = "";
String msg = "";

// check if it's post back (after submit clicked)
if (request.getParameter("circle1Radius") != null) {
    try {
        strCircle1Radius = request.getParameter("circle1Radius");
        strCircle2Radius = request.getParameter("circle2Radius");

        Circle myCircle1 = new Circle(Double.parseDouble(strCircle1Radius));
        Circle myCircle2 = new Circle(Double.parseDouble(strCircle2Radius));

    } catch (NumberFormatException e) {
        circle2Msg = e.getMessage();
    }

```