# Class Techniques - Part Two

## Overview

This lab focuses on the various class techniques discussed in the chapter - data classes, enums, extensions, and operator overloading.

## Roadmap

There are 4 exercises in this lab, of which the last exercise is "if time permits". Here's a brief summary of the tasks you'll perform in each exercise; more detailed instructions follow later in this lab doc:

1. Define a data class and use its standard methods
2. Define and use an enum type
3. Define an extension property for the data class
4. (If Time Permits) Implement operator overloading

## Exercise 1: Define a data class and use its standard methods

Define a data class version of an Employee class, containing the following properties:

* The name of the employee (fixed)
* The name of the company he/she works for (fixed)
* The employee's salary (can change)

Create an Employee instance. Print out each of the properties in the Employee instance. Then try to modify a property - verify only the salary is modifiable.

Create another Employee instance, coincidentally with the same values. Test whether it's equal to the first instance, via == and equals. Should these operations return true or false? What's the difference between using == and equals?

Create another Employee instance as a copy of the first one, but with a different salary. Use the copy method to achieve this effect. Destructure the object into separate variables holding just the name and salary, and print these details on the console.

## Exercise 2: Define and use an enum type

Define an enum type named EmploymentType, with three possible values as follows:

* Permanent
* Casual
* Contractor

Modify your Employee class so that an employee has an employment type (the default employment type should be permanent). Also, employees should be able to change their employment type.

Create an employee with one of the employment types, change the employment type, and then print the employee's details on the console. Verify the details are displayed correctly.

## Exercise 3: Define an extension property for the data class

In this exercise you'll define an extension property for the Employee data class (the purpose of this exercise is to get some practice writing extensions - you could equally well have defined a member inside the Employee class instead).

The extension property should be named taxStatus and should return a String as follows, depending on the employee's salary:

* If the employee earns less than 20000:  
   <name> earns <salary> [NO TAX PAYABLE]
* If the employee earns 20000 or more, but less than 40000:  
   <name> earns <salary> [NORMAL-RATE TAX PAYABLE]
* If the employee earns 40000 or more:  
   <name> earns <salary> [HIGHER-RATE TAX PAYABLE]

In your client code, declare some Employee instances with various salaries. Get the taxStatus property for each employee, and display the info on the console.

## Exercise 4 (If Time Permits): Implement operator overloading

Implement operator functions to support the following operators for employees:

* emp1 + num // Adds num to emp1's salary
* emp1 - num // Subtracts num from emp1's salary
* emp1 > emp2 // Compare salaries for employees (ditto for other relational operators)

Test these operators thoroughly in your client code.