# Hands on Java 11 OCP Certification Prep

# Lab 2 – New Features

Note: You can use whatever IDE (or command line) you feel most comfortable with for this lab.

## Step 1 - private interface methods

1. Create an interface with method signatures:
   1. **default int getRandomNumberTickets(int max)** – returns a random number of tickets between 1 and max inclusive
   2. **default int getNumberStrikes()** – returns a random number between 0 and 3 inclusive
2. Create a class named **Giants** that implements this interface. Call both methods and print the results.
3. Refactor the interface to use a private method with the following signature and have the two existing interface methods call it
   1. **private int getRandom(int min, int max)**

## Step 2 - Underscore

1. Try renaming getRandomNumberTickets() to \_ and compile. Note that you get a compiler error.
2. Try renaming getRandomNumberTickets() to \_\_ (double underscore). Note that this fixes the compiler error.

## Step 3 – effectively final try with resources

1. Create a class called **OracleCodeOne**
2. Create a text file containing the predicated high temperature for the next three days here in San Francisco. (It’s always nice here. Feel free to type use 69, 70 and 71 rather than looking this up)
3. Write a method that:
   1. Creates a Path or File variable referencing the file
   2. Creates a Stream<String> or Reader variable referencing the Path or File.
   3. Reads the file and prints the average temperature.

Tip: You can use the Files.lines() method and mapToDouble() to do this with streams.

* 1. Ensures the resource is closed using a try with resources.

## Step 4 - var

1. Copy your **OracleCodeOne** code into a class called **OracleCodeOneWithVar**
   1. Switch to use var in every place it is allowed
   2. Compile and run
   3. Confirm you get the same results

## Bonus Step (5): Streams

See how many of these you can write:

1. Create a class with a method that takes a **String** and returns a **Stream<String>** of 0 elements (if null) and 1 element (if not null)
2. Write a stream that counts down from 10 to 1. Use this stream to print a countdown followed by “blast off”
3. Write a stream that counts odd numbers starting with 1. Print out the 40th element of the stream.
4. Create a file with a few sentences. Read it into a stream and print all the words until you get to one that is six characters or more. (Do not print the six character word.)
5. Using the same file in the previous sentence, print just the first word that is six characters or more. (do not use dropWhile for this)
6. Now print the first word that is six characters or more followed by the next three words of any length.
7. Create an infinite stream of randomly generated positive integers between 1 and 1000. Print all of the even numbers before the first odd number.
8. Create a stream of ten randomly generated numbers from 1 to 100. Print all of the entries after (and including) the first number greater than 50.