

Lab 9 – Part I

1. In the code folder for this lab, there is a version of `Project4_Students`, and in the `business` package there is a `Main` class containing a `main` method. The `main` method calls several other methods; each of these attempts to extract information from the `Library System`. Implement these methods using stream pipelines (the body of each method should contain a *single line of code* – namely, the stream pipeline).
2. In the code folder for this problem, there are classes `Order` and `OrderItem`, along with a `Main` class that has a `main` method. The `main` method loads test data, populating a list of `Order` instances, each of which contains a list of `OrderItem` objects. It then calls `displayAllOrders`, which displays this test data to the console using formatted output; each `Order`, together with its `OrderItems`, is shown. The last method call in the `main` method is to an unimplemented method `showAllOrderItems`. This method is supposed to display all `OrderItems` separately, apart from the owning `Orders`. Carry out this implementation (using the technique described in the slides) by embedding the `Orders` list in a `Stream` and using `flatMap`.
3. In the code for this lab (for prob3) there is an `Employee` class. The `main` method of that class creates a list of `Employee` objects for testing purposes. Finish coding the `main` method by writing code to sort the list. Sorting order should be done first by name in ascending order, then by salary in descending order. Startup code is provided in your code folder.
4. This exercise asks you to work with potentially infinite streams of prime numbers.
 - A. To begin, create a final variable `Stream<Integer> primes` that contains all prime numbers (in particular, the `Stream` is infinite). Generate the primes using the `iterate` method of `Stream` – do *not* use the `map` or `filter` `Stream` operations.
 - B. Next, create a variation of the `primes` `Stream` that can be called multiple times by a method `printFirstNPrimes(long n)`, which prints to the console the first `n` prime numbers. Note that the `Stream` `primes` that you created in part A cannot be used a second time; how can you get around that limitation? Prove that you succeeded by calling the method `printFirstNPrimes(long n)` (from a `main` method) more than once.

If you succeed, you should be able to run the following code without getting a runtime exception:

```
public static void main(String[] args) {  
    PrimeStream ps = new PrimeStream(); //PrimeStream is enclosing class  
    ps.printFirstNPrimes(10);  
    System.out.println("====");  
    ps.printFirstNPrimes(5);  
}
```