Week 4

Start time 08:30

Break 10:15 – 10:30

<https://github.com/DannyDevlin19671/continued-ed.git>

Recap of Last week

<https://labs.neueda.com/docs/java-essentials/>

Lambda Functions

<https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html>

<https://docs.oracle.com/javase/tutorial/java/generics/index.html>

**Task 1: Implement a Functional Interface for a Basic Arithmetic Operation**

**Task:**  
Create a functional interface named Calculator with a single abstract method division(double a, double b). Then, use a lambda expression to implement a division operation and test it.

**Task 2: Sort a List of Custom Objects Using a Lambda Comparator**

**Task:**  
Create a Person class with fields name and age. Given a list of Person objects, use a lambda expression to sort the list by age in ascending order and print the sorted list.

**Task:**  
Use a lambda expression to sort the list by name in ascending order and print the sorted list.

**Task:**  
Use a lambda expression to sort the list by age and name in ascending order and print the sorted list.

**Task 3 (Advanced): Filter and Sum Even Numbers from a List Using Streams and Lambdas**

**Task:**  
Given a list of integers, use the Stream API along with lambda expressions to filter out even numbers and calculate their sum.

**Collections**

**Task 1: Create and Iterate Over an ArrayList**

**Task:**  
Write a program that creates an ArrayList of strings, adds a few fruit names, and prints each element using a for-each loop.

**Task 2: Remove Duplicates from an ArrayList**

**Task:**  
Create an ArrayList of integers containing duplicates. Then, use a HashSet to remove the duplicates and print the unique values.

**Task 3: Sort an ArrayList of Strings**

**Task:**  
Create an ArrayList of strings with unsorted names. Sort the list in alphabetical order using Collections.sort and then print the sorted list.

Complex Tasks

**Example 1: Grouping Employees by Department and Sorting by Salary**

**Task:**  
Create an Employee class with fields for name, department, and salary. Then, given a list of employees, group them by department and sort each department’s employees by salary in descending order. Finally, print the grouped and sorted results.

**Example 2: Implementing an LRU Cache Using LinkedHashMap**

**Task:**  
Create a generic Least Recently Used (LRU) cache by extending LinkedHashMap. The cache should automatically remove the oldest entry when a defined capacity is exceeded. Demonstrate the cache with some put and get operations.