Objective(s):

1. to review java Collection
2. a glance at array vs linked list performance difference.

In this course, your working directory (relative to your current working directory would be

- .\solutions\xxx which xxx contains required files
- .\Labs which contains driver class (main class)

Task1:

Combine lis1a and lis1b contents to lis1c, i.e. the data in lis1a and lis1b is unchanged.

```
public static void task1() { // combine two ArrayList
    System.out.println("--task1---");
    ArrayList<String> lis1a = new
            ArrayList<>(Arrays.asList("Lily","Daisy"));
    ArrayList<String> lis1b = new
            ArrayList<>(Arrays.asList("Tulip","Daisy"));
    ArrayList<String> lis1c;
       /* your code */
    System.out.println(lis1c);
    System.out.println(lis1a);
}
```

Task2:

Passing a collection to a collection constructor (in this case lis2b) activates copy-constructor. Complete the code to display “shallow” to the screen if the copy-constructor results in shallow copy.

```
public static void task2() { // shallow copy matter
    System.out.println("--task2---");
    ArrayList<StringBuilder> lis2a = new ArrayList<>(
                            Arrays.asList(
                             new StringBuilder("Lily"),
                             new StringBuilder( "Daisy")));
    ArrayList<StringBuilder> lis2b = new ArrayList<>(lis2a);
    lis2b.add(new StringBuilder("30"));
    // System.out.println(lis2b);
    // System.out.println(lis2a);  // lis2a is unchanged
    StringBuilder sb = lis2a.get(0);
    sb.append("mySuffix");
    // Does lis2b.get(0) object change? Or it is not
        //affected. Check it yourself.
    // complete the task by display "shallow copy"
        //if lis2b first element is affected.
    /* your code */
}
```

Task3:

Remove all lis3's elements but the first one.

```
public static void task3() {
    System.out.println("--task3---");
    List<String> lis3 = new ArrayList<>(
            Arrays.asList("Lily","Daisy","Tulip","Daisy"));
    /* your code */
    System.out.println(lis3); // Lily
}
```

Task 4:

Looking at task1() code, you should expect that flowers and dogs are supposed to contain **only distinct elements** because they both a set.

Create Dog.java and (enum) **Breed.java** in solutions.code1 to complete the task.

Remark:

Dog constructor is Dog{Breed b, int weight}

```
public static void task4() {

  System.out.println("--task4---");
  ArrayList<String> lis4a = new ArrayList<>(
            Arrays.asList("Lily","Daisy","Tulip","Daisy"));
  HashSet<String> flowers = new HashSet<>(lis4a);
  for (String ele : flowers) {
      System.out.print(ele + " ");
  } System.out.println();

  ArrayList<Dog> lis4b = new ArrayList<>(Arrays.asList(
                        new Dog(Breed.pomeranian,1200),
                        new Dog(Breed.beagle, 2300),
                        new Dog(Breed.jack, 1440),
                        new Dog(Breed.beagle,2300)));

  HashSet<Dog> dogs = new HashSet<>(lis4b);
  for (Dog ele : dogs) {
      System.out.print(ele + " ");
  }
  System.out.println();
}
```

Task 5:

Complete task5() such that it displays the frequency of dogs' breed.

```
static void task5() {
  System.out.println("--task5---");
  ArrayList<Dog> lis5 = new ArrayList<>(Arrays.asList(
                        new Dog(Breed.pomeranian,1200),
                        new Dog(Breed.beagle, 2300),
                        new Dog(Breed.jack, 1440),
                        new Dog(Breed.beagle,2300)));

  HashMap<Breed,Integer> map = new HashMap<>();
  /* your code */
  for (Entry<Breed, Integer> ele : map.entrySet()) {
    System.out.println(ele.getKey()
                        + "\t" + ele.getValue());
  }
}
```

Task 6:

Given a list of dogs. Find the number of distinct dogs.

Complete task6().

```
static void task6() {
  System.out.println("–task6---");
  System.out.print("The number of unique element is ");
  ArrayList<Dog> lis6 = new ArrayList<>(Arrays.asList(
                        new Dog(Breed.pomeranian,1200),
                        new Dog(Breed.beagle, 2300),
                        new Dog(Breed.jack, 1440),
                        new Dog(Breed.beagle,2300)));

  /* your code */
}
```

Task 7:

Create **lis, llis, and arr** (of ArrayList, LinkedList and array) as specified. (Though it is not required for this task but we just want the data to look randomized, therefore shuffle the lis's content before applying it to llis and arr.)

You are to collect access time for at the beginning, at 25%, 50% and 75% of its position.

Write the output of task3()

```
static int N = 10_000;
static Integer [] arr = new Integer [N];
static int num_iter = 100_000 * 10;
static ArrayList<Integer> lis = new ArrayList<>();
static LinkedList<Integer> llis;
static {
  for (int i = 0; i < N; i++) {
      lis.add(i);
  }
  Collections.shuffle(lis);
  lis.toArray(arr);
  llis = new LinkedList<>(lis);
}
static void demo_arrayList(int idx) {
  int value;
  long start = System.currentTimeMillis();
  for (int iter = 0; iter < num_iter; iter++) {
      value = lis.get(idx);
  }
  long time = (System.currentTimeMillis() - start);
  System.out.println("ArrayList \ttakes " + time);
}
static void demo_linkedList(int idx) {
    /* your code */
}
static void demo_array(int idx) {
    /* your code */
}
static void task3() {
        // accessing 0, 25%, 50% and 75%
    for (int index = 0; index < arr.length;
                            index += arr.length/4) {
        System.out.println("Index is at " + index);
        demo_arrayList(index);
        demo_linkedList(index);
        demo_array(index);
    }
}
```

Index is at 0
ArrayList takes 2447800
Linkedlist takes 3146100
Array takes 2021800

Index is at 2500
ArrayList takes 2447800
Linkedlist takes 3038955900
Array takes 1742800

Index is at 5000
ArrayList takes 400
Linkedlist takes 6599283600
Array takes 82700

Index is at 7500
ArrayList takes 600
Linkedlist takes 3177414800
Array takes 500

Task 8:

Answer the following questions.

8.1 How should we explain the different time used for accessing the mid element of the lis, llis, and arr.

Arraylist takes the least time, Array comes in second and the slowest is Linkedlist, Arraylist and Array can directly access the mid point of the list but linked list takes time to travel from a reference point.

8.2 why accessing the position $(3/4)^{th}$ of the data is faster than accessing the mid element in llis.

Because linkedlist needs a reference point to start with, position(3/4)th is closer to the end of the list and position(1/4)th is closer to the start of the list so the mid point is slower to access because It's the halfway of both

Submission: (rename your work to) Dog_XXYYYY.java, Lab1_XXYYYY.java where XX is your first 2 digit of your student id and YYYY is its last four digits. And this pdf.

Due date: TBA