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

- .\LabXX\yyy.java which contains driver class (main class)
- .\LabsXX\pack which contains required class

Task1:

Given 2 arraylist of same type, Create an arraylist named lis1c which contains all elements combined from lis1a and lis1b contents, 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);
    //[lilly, tulip]
    System.out.println(lis1a);
    // [lilly, tulip, tulip, daisy]
}
```

Task2:

Passing a collection to a collection constructor (in this case lis2b) activates copy-constructor. Should A or B be displayed?

```
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");

    // pick your answer what would be the below result
    if (lis2b.get(0).equals(sb)) {
       System.out.println("lis2b[0] also becomes " + sb); //A
    } else {
       System.out.println("copy constructor
             creates a new copy of lis2a[0] for lis2b"); //B
    }
}
```

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 the 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 pack 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();
// Dog(beagle, 2300) Dog(jack, 1440) Dog(pomeranian, 1200)
}
```

Task 5:

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

Complete task5().

```
static void task5() {
  System.out.println("–task5---");
  System.out.print("The number of unique element is ");
  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)));

  /* your code */          // 3
}
```

Task 6:

Complete task6() such that it displays the frequency of dogs' breed.

```
static void task6() {
  System.out.println("–task6---");
  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)));

  HashMap<Breed,Integer> map = new HashMap<>();
  /* your code */
  for (Entry<Breed, Integer> ele : map.entrySet()) {
    System.out.println(ele.getKey()
                          + "\t" + ele.getValue());
  } //pomeranian 1    beable 2    jack 1

}
```

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 task7()

```
static void task7() { // access time
    int N = 10_000;
    ArrayList<Integer> lis = new ArrayList<>();
    Integer [] arr = new Integer[N];
    LinkedList<Integer> llis;

    for (int i = 0; i < N; i++) {
        lis.add(i+1);
    }
    Collections.shuffle(lis);
    lis.toArray(arr);
    llis = new LinkedList<>(lis);

    task7_timer(arr, lis, llis);
}
```

```
static void task7_timer(Integer [] arr, ArrayList<Integer> lis,
                                        LinkedList<Integer> llis) {
  int factor = 10;
  int num_iter = 100_000 * factor; // in case your CPU is too powerful
  int [] index = {0, (arr.length/4), (arr.length/2), (3*arr.length/4)};
  long start, stop = System.nanoTime();
  int pos = 0,x;
  for (int i = 0; i < index.length; i++) {
    pos = index[i];
    start = System.nanoTime();
    for (int j = 0; j < num_iter; j++)
        x = arr[pos];
    stop = System.nanoTime();
    System.out.printf("Array accessing at %d takes %s\n",pos
                                ,String.format("%,d",(stop-start)));
    start = System.nanoTime();
    for (int j = 0; j < num_iter; j++)
        x = lis.get(pos);
    stop = System.nanoTime();
    System.out.printf("ArrayList accessing at %d takes %s\n",pos
                                ,String.format("%,d",(stop-start)));
    start = System.nanoTime();
    for (int j = 0; j < num_iter; j++)
        x = lis.get(pos);
    stop = System.nanoTime();
    System.out.printf("LinkedList accessing at %d takes %s\n"
                                ,String.format("%,d",(stop-start)));
    System.out.println("-----------------");
  }
}
```

(for task7)

Task 8:

Answer the following questions.

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

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

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