วัตถุประสงค์

A. ทดลองการใช้ Collections ของ java

กิจกรรมที่ 1 (Ex.1 – Ex.5) ศึกษา โครงสร้างของคลาส และ การส่ง reference เป็น parameter

คลาส Employee ประกอบด้วยแอททริบิวท์ String id, String name และ Integer salary โดย salary มีค่าเป็นผลรวมของค่า int ของ char ใน name

```
public Employee(String n) {
  name = n;
  int sal = 0;
  for (int j = 0; j < name.length(); j++)
      sal += name.charAt(j);
  salary = sal;
  int memAddress = System.identityHashCode(this);
  id = = ""; //Integer.toHexString(memAddress);
}

@Override
public String toString() {
  return id + " [" + name + "(" + salary + ")]";
}
```

1.1

Generate hashCode() และ equals()

ทดสอบ hashCode() จะพบว่า id1 และ id2 มีค่าเท่ากัน เพราะขณะนี้ instance attribute ทั้ง 3 ค่ามีค่าเหมือนกัน

โชคดีที่สภาพแวดล้อมที่อยู่ขณะนี้สามารถใช้ตำแหน่งของอ๊อบเจกต์เป็น id ได้

ลองใช้ identityHashCode() เพื่อ name และ salary เท่ากันแต่เป็นคนละอ๊อบเจกต์ ให้ค่า hashCode ต่างกัน โดยแก้โค้ดให้ใช้ memAddress เป็น id

(หมายเหตุ การเก็บ id เป็น String ค่า hex ซึ่งประมวลผลได้ช้ากว่า numeric เป็นความขี้ลืมส่วนตัว)

```
static void ex1() {
  // hasCode() vs System.identityHashCode()
  Employee emp1 = new Employee("yindee");
  Employee emp2 = new Employee("yindee");
  int id1 = emp1.hashCode();
  int id2 = emp2.hashCode();
  String s1 = Integer.toHexString(id1);
  String s2 = Integer.toHexString(id2);
  System.out.println(s1);
  System.out.println(s2 + " " + s1.compareTo(s2));
}
```

1.2

การส่ง reference เป็นพารามิเตอร์ java จะสร้าง copy ของ reference นั้น ทำให้ e ใน ex2_1 เป็น local variable ชนิด reference ไปยังอ๊อบเจกต์เดียวกับที่ emp1 จาก ex2 อ้างถึง

การเปลี่ยนแปลงค่าใดๆ ย่อมส่งผลถึงอ๊อบเจกต์นั้นเมื่อ ex2_1() ทำงานเสร็จ

```
static void ex2() {
  // java creates a copy of the reference when
  // an obj is passed to a method.
  // changes to an obj will be effected.
  Employee emp1 = new Employee("preeda");
  System.out.println(emp1);
  ex2_1(emp1);
  System.out.println(emp1);
}

private static void ex2_1(Employee e) {
    e.setName("aba abi abo abe");
}
```

(หมายเหตุ primitive ไม่ส่งผล)

1.3 เนื่องจาก list เป็นคนละ reference กับ singers การเปลี่ยนแปลงของ list ย่อมไม่ส่งผลถึง singers

(สืบเนื่องจาก 1.2 หลายเมธอดใน java การประมวลผลกับตัวแปร เกิดการเปลี่ยนแปลงของค่าในอ๊อบเจกต์ โดยไม่ต้อง return ออกมา)

```
static void ex3() {
  // changes to an obj will be effected when
  // passing as a parameter but not a collection
  ArrayList<Employee> singers =
                              new ArrayList<>();
    singers.add(new Employee("aba"));
    singers.add(new Employee("abi"));
    singers.add(new Employee("abo"));
    singers.add(new Employee("abe"));
  ex3_1(singers);
  System.out.println(singers.get(0));
  // cha cha cha
  System.out.println(singers);
  // not empty because it's list who changes
}

private static void ex3_1(List<Employee> list) {
  list.get(0).setName("cha cha cha");
  // changes takes effect
  list = new ArrayList<>();
  // changes takes "no" effect on collection
```

1.4 ในกรณีที่จำเป็นต้องนำ reference จาก method ออกมาใช้ สามารถใช้ return โดยมี reference ไปรับ

สังเกตว่า list ที่ return ออกมาไม่จำเป็นต้องเป็นการรับเข้าไปเหมือน 1.3

```
static void ex4() {
  // what if we do need to change the referenced
  // object to another
  ArrayList<Employee> singers = new ArrayList<>();
        singers = ex4_1();
        System.out.println(singers);
  }

private static ArrayList<Employee> ex4_1( /*
                            List<Employee> list */ ) {
    ArrayList<Employee> list = new ArrayList<>();
    // whether list is passed in or newly allocated,
    // it's a local reference.
        list.add(new Employee("aba"));
        list.add(new Employee("abi"));
      return (ArrayList<Employee>) list;
}
```

1.5

การสร้าง reference ใหม่นั้น เราไม่ได้สร้าง สมาชิกใหม่ เรียกคอนเซ็ปต์นี้ว่า shallow copy

1.5.1

การสร้าง list ใหม่ด้วย .clone() นั้นเราได้ reference ของ ArrayList ใหม่

1.5.2

การสร้าง reference ใหม่ ด้วยการส่ง Collection ให้ constructor ดังตัวอย่าง tmp อ้างถึง list เดียวกับที่ singers อ้างถึง

1.5.3 ตัวอย่างแสดงการสร้าง reference ใหม่ ด้วย subList()

ศึกษาเรื่อง shallow copy เพิ่มเติมที่

https://howtodoinjava.com/java/collections/arraylist/arraylist-clone-deep-copy/

```
static void ex5() {
  // collection clone is shallow copy
  ArrayList<Employee> singers = new ArrayList<>();
    singers.add(new Employee("aba"));
    singers.add(new Employee("abi"));
    singers.add(new Employee("abo"));
    singers.add(new Employee("abe"));

  @SuppressWarnings("unchecked") // by clone()
  List<Employee> tmp =
                (ArrayList<Employee>) singers.clone();
  tmp.get(0).setName("cha");
  System.out.println(singers);

  // by new ArrayList<>();
  tmp = new ArrayList<>(singers);
  tmp.get(1).setName("cha cha");
  System.out.println(singers);

  // by subList()
  tmp = singers.subList(0, singers.size() - 1);
  tmp.get(2).setName("cha cha cha");
  System.out.println(tmp);
}
```

กิจกรรมที่ 2 (q6 – q15)

2.1 ศึกษา .addAll() แล้ว implement

private static List<Employee> q6_1(List<Employee> l1, List<Employee> l2);

ให้ join คือ List ของ Employee ทั้ง 4 โดยข้อมูลใน singers1 และ singer2 ไม่เปลี่ยน

```
static void q6() {
  List<Employee> singers1 = new ArrayList<Employee>(
              Arrays.asList(new Employee("aba"),
                            new Employee("abi")));
  // Arrays.asList() returns fixed sized ...but
  // ArrayList is not fixed, hence can call addAll()
  List<Employee> singers2 = new ArrayList<>();
  singers2.add(new Employee("abo"));
  singers2.add(new Employee("abe"));

  List<Employee> join = q6_1(singers1, singers2);

  System.out.println(singers1);
  System.out.println(singers2);
  System.out.println(join);
  singers1.addAll(singers2);
  System.out.println(singers1);
}
private static List<Employee> q6_1(List<Employee> l1, List<Employee> l2) { … }
```

2.2 implement q7_1() โดยให้ singers เหลือ แต่ element แรก

```
static void q7() {
  List<Employee> singers = Arrays.asList(
            new Employee("aba"), new Employee("abi"));
  System.out.println(singers);
  singers = q7_1(singers);
  System.out.println(singers);
}
private static List<Employee> q7_1(
                        List<Employee> list) { … }
```

2.3 ใช้ HashSet เพื่อให้ q8_ans เป็น list ที่ไม่มี element ซ้ำ

```
static void q8() {
  //q8 - q11 yindee preeda pramote from Lab9_xxyyyy
  List<Employee> list1 = Arrays.asList(yindee,
                                        pramote);
  List<Employee> list2 = Arrays.asList(pramote,
                                         preeda);
  Set<Employee> empSet;
  /* your task */
  List<Employee> q8_ans = new ArrayList<>(empSet);

        System.out.println(q8_ans);
    }
```

2.4 ใช้ retainAll() เพื่อ ให้ empSet1 เก็บเฉพาะ element ที่ปรากฏในทั้ง list1 และ list2

```
static void q9() {
  Set<Employee> empSet1 = new HashSet<>(
             Arrays.asList(yindee, preeda, pramote));
  Set<Employee> empSet2 = new HashSet<>(
              Arrays.asList(yindee, pramote));

  /* your task */
  System.out.println(empSet1);
}
```

2.5 ใช้ removeAll() เพื่อ ให้ empSet1 เก็บเฉพาะ element ที่ปรากฏเฉพาะใน list1 (ไม่ปรากฏใน list 2)

```
static void q10() {
static void q9() {
  Set<Employee> empSet1 = new HashSet<>(
              Arrays.asList(yindee, preeda, pramote));
  Set<Employee> empSet2 = new HashSet<>(
               Arrays.asList(yindee, pramote));

  /* your task */
  System.out.println(empSet1);
}
```

2.6 แปลง empSet ให้เป็นอาร์เรย์

```
static void q11() {
  Set<Employee> empSet = new HashSet<>(
      Arrays.asList(yindee, preeda, promote, preeda));
  Employee[] q11_ans = new Employee[empSet.size()];

  /* your task */

  for (Employee e : q11_ans)
      System.out.print(e + " ");
  System.out.println();
}
```

สำหรับข้อ 2.7 – 2.10

สร้าง empList ใน main class ตามตัวอย่าง

```
public class TestCollections {
  static ArrayList<Employee> empList;
  …
  static {
    empList = new ArrayList<>();
    try (Scanner input = new
      Scanner(Paths.get("packExercise//names.csv"))) {
        input.nextLine(); // skip first row
        while (input.hasNext()) {
            String row = input.nextLine().trim();
            empList.add(new Employee(row));
        }
    println("There are " + empList.size() +
                                        " employees.");
    } catch (IOException e) { e.printStackTrace(); }
    …
}
```

2.7

เก็บค่า Employee 3 คนแรก ไว้ใน map โดย ใช้ name เป็น key

```
static void q12() {
  int n = 3;
  Map<String, Employee> map = new HashMap<>();
  for (int i = 0; i < n; i++) {
    /* your task */
  }
  System.out.println(map);
}
```

2.8 จากข้อที่แล้ว implement ให้ keySet เก็บ key ของ map

```
static void q13() {
  int n = 3;
  Map<String, Employee> map = new HashMap<>();
  for (int i = 0; i < n; i++) {
    /* your q12 task */
  }
  /* your task */
  System.out.println(keySet);
}
```

2.9 implement ให้ q14_ans เก็บ Employee ที่เงินเดือนน้อยที่สุด 5 อันดับแรก

```
static void q14() {
  PriorityQueue<Employee> pq
  = new PriorityQueue<>(
   (e1, e2) ->
     Integer.compare(e1.getSalary(), e2.getSalary()));
  pq.addAll(empList);
  List<Employee> q14_ans = new ArrayList<>();
  for (int i = 0; i < 5; i++) {
      /* your task */
  }
  System.out.println(q14_ans);
}
```

2.10 comment คำตอบของคำถามต่อไปนี้ใต้ static void q15() {

หาค่าเฉลี่ยการทำงานระหว่าง pq กับ sort ...เทคนิคใดทำงานเร็วกว่ากัน (ไม่ต้องห่วงเรื่องนัยยะว่าเร็วกว่ากันแค่ไหน ... อยากดูผลโหวตจากการรันของทุกคน)

```
static void q15() {
  int sum = 0;
  int cnt = 0;
  Iterator<Employee> it;
  Long pqStart = System.nanoTime(); // System.currentTimeMillis();
  sum = 0;
  PriorityQueue<Employee> pq = new PriorityQueue<>((e1, e2) ->
                    Integer.compare(e1.getSalary(), e2.getSalary()));
  pq.addAll(empList);

  it = pq.iterator();
  while (it.hasNext()) {
      sum += it.next().getSalary();
      cnt++;
  }
  System.out.println("PQ (" + sum + ") takes "
                + String.format("%,d", System.nanoTime() - pqStart));

  Long listStart = System.nanoTime();
  ArrayList<Employee> clone = new ArrayList<>();
  sum = 0;
  clone.addAll(empList);

  Collections.sort(clone, (e1, e2) ->
                   Integer.compare(e1.getSalary(), e2.getSalary()));

  it = clone.iterator();
  while (it.hasNext()) {
       sum += it.next().getSalary();
       cnt--;
  }
  System.out.println("ArrayList (" + sum + ") takes "
                + String.format("%,d", System.nanoTime() - listStart));
  if (cnt != 0)
            System.out.println("error on number of elements");
}
```

คำสั่ง

ส่ง Lab9_xxyyyy.java

กำหนดส่ง TBA