วัตถุประสงค์:. หัดใช้ lambda expression และ method reference

กิจกรรมที่ 1

1.1 For-Each รับ **Consumer** โดย consumer ไม่คืนค่ากลับมา

1.2 **Predicate** คืนค่า boolean เขียน lambda expression เพื่อแสดงผลเฉพาะจังหวัดที่ลงท้ายด้วย ค่า str (ส่ง str มาเป็น “buri”)

1.3 **Function** รับค่าและคืนค่า เขียน func เพื่อแสดงผลเฉพาะชื่อจังหวัด n ตัวอักษรแรก (หากชื่อจังหวัดไม่เกิน n ตัวอักษร แสดงทั้งหมด) ตัวอย่างการเรียกใช้ค่า n = 4

1.4 **Supplier** ไม่รับพารามิเตอร์ ตัวอย่างให้สุ่มตำแหน่ง เพื่อตัดตัวอักษรตำแหน่งนั้นออกจากชื่อจังหวัดนั้นๆ

(เขียน public String [ ] randomRemoveOneCharacter())

```
public class Solution {
    List<String> cities;
    public Solution() {
        cities = Arrays.asList("Lampang", "Bangkok",
                        "Petchaburi","Tak", "Khonkaen");
    }
    public void printAllCities() {
        Consumer<String> consumer = city ->
                                System.out.println(city);
        cities.forEach(consumer);
    }

    public String[] citiesEndsWith(String str) {
        List<String> ans = new ArrayList<>();
        Predicate<String> pred = /* q1.2*/
        for (String city : cities) {
            if (pred.test(city))
                ans.add(city);
        }
        String [] result = new String[ans.size()];
        result = ans.toArray(result);
        return result;
    }
    public String[] first_n_letters(int n) {
        List<String> ans = new ArrayList<>();
        /* q1.3*/
        String [] result = new String[ans.size()];
        return ans.toArray(result);
    }
    /* q1.4*/
    static void sub_01() {
      System.out.println("test
                  java built-in functional interface");
      Solution s = new Solution();
      // consumer;
      s.printAllCities();
      // predicate
      String [] ans = s.citiesEndsWith("buri");
      System.out.println(Arrays.toString(ans));
      // function
      System.out.println(Arrays.toString(
                        s.first_n_letters(4)));
      // supplier
      System.out.println(Arrays.toString(
                        s.randomRemoveOneCharacter()));
    }
}
```

```
test java built-in functional i
Lampang
Bangkok
Petchaburi
Tak
Khonkaen
[Petchaburi]
[Lamp, Bang, Petc, Tak, Khon]
[Lmpang, angkok, Petchauri, Ta,
```

กิจกรรมที่ 2

2.1a กำหนด HalfValueInterface เป็น funtionial Interface ...implement method ให้ halfVal แสดงค่า n/2

ตอบ q2.1a

Q2.1b abstract method ของ HalfValueInterface ชื่อ _____

2.2 Consumer เป็น functional interface ที่รับค่าด้วย method accept() ...implement ให้ consumer นำ n ไปแสดงค่า n/2 (สังเกตว่า ใช้ Consumer แทนการสร้างและ implement HalfValueInterface

ตอบ q2.2

2.3 forEach() รับ Consumer ...เพื่อป้องกันความสับสน ให้ halfMe เป็น Consumer มีพฤติกรรมเหมือน 2.2 ...ดังนั้นส่ง halfMe ให้ forEach() ว่าเรียก halfMe ให้ ทำงานโดยไม่ต้องเรียก .accept() เหมือน q2.2 ได้

ตอบ q2.3

```
public static void q1_halfEachNumber() {
    List<Integer> nums = Arrays.asList(100, 105);
        HalfValueInterface q0 =
            new HalfValueInterface() {
                public void printHalf(int n) {
                    System.out.println(n / 2);
                }
            };
        for (int n : nums) {
            q0.printHalf(n);
        }

    HalfValueInterface halfVal = /* q1.1a */
    for (int n : nums) {
        halfVal.printHalf(n);
    }

    Consumer<Integer> consumer = n -> /* q2.2 */
    for (int n : nums) {
        consumer.accept(n);
    }

    Consumer<Integer> halfMe = n ->
                        System.out.println(n / 2);
    nums./* q2.3 */;

    nums.forEach(/* q2.4 */);

    NumberProcessor np = new NumberProcessor();
    nums.forEach(/* q2.5 */);
}
```

2.4 แทนที่จะ ต้องสร้าง Consumer เราสามารถเขียน lambda ของ halfMe ให้ .forEach() ได้เลย

ตอบ q2.4

2.5 NumberProcess มี printHalf(int n) ซึ่งแสดงค่า n/2 เหมือนของ HalfValueInterface ...ใช้ method reference จาก printHalf() ของ np เพื่อให้ forEach ใช้ แทนที่จำเป็นต้องสร้างจาก functional interface

ตอบ q2.5

กิจกรรมที่ 3

```java
public class TestPack8MethodReference {
    static List<Singer> singerList;
    static {
        singerList = new ArrayList<>();
        singerList.add(new Singer(n:"Aba", SingStyle.POP));
        singerList.add(new Singer(n:"Abi", SingStyle.ROCK));
        singerList.add(new Singer(n:"Abo", SingStyle.POP));
        singerList.add(new Singer(n:"Abe", SingStyle.ROCK));
    }
```

3.1 compare() ใน interface Comparator จะ return ค่า - , 0 , + สำหรับ 2 ค่าใดๆ เพื่อให้ jvm ทราบว่าค่าไหนมาก่อน / หลัง

เขียน byStylee2 ด้วย lambda expression เพื่อให้ sort() เรียงข้อมูล singer ใน singerList ตาม String ของ SingStyle (P มาก่อน R) กำหนด public String getStyleString() { return style.toString(); }

ตอบ q3.1

```java
public enum SingStyle {
    ROCK, POP
}
class Singer {
    private String name;
    private SingStyle style; // preferred enum

    public Singer(String n, SingStyle s) {…

    public String getName() {…

    public SingStyle getStyle() {…

    public String getStyleString() {…

    public void sing() {…

    @Override
    public String toString() {…

    public int compareByName(Singer s) {
        return name.compareTo(s.getName());
    }
}
```

กิจกรรมที่ 4

การเรียงสามารถเรียกทาง Collections.sort() หรือ List.sort()

4.1 เราสามารถสร้าง Comparator โดยระบุเฉพาะเกณฑ์ ด้วย method reference จาก ไวยากรณ์ Comparator.comparing(Class:Method)

(พารามิเตอร์ของ comparing เป็น functional interface)

เขียน 4.1 ให้เรียงตามชื่อนักร้อง

ตอบ q4.1

```java
public static void q3_lambda_comparator() {

  Comparator<Singer> byStyle1 = new Comparator<>() {
    @Override
    public int compare(Singer o1, Singer o2) {
       return o1.getStyle().compareTo(o2.getStyle());
    } //by Enum .ordinal()
  };
  Collections.sort(singerList, byStyle1);
  singerList.forEach(System.out::println);

  Comparator<Singer> byStyle2 =
                                    /* q3.1 */;
   Collections.sort(singerList, byStyle2);
   singerList.forEach(System.out::println);
}
public static void q4_method_reference_comparator() {
        System.out.println("--- Q4.1 by Name---");
        Collections.sort(singerList, /* q4.1 */ );
        singerList.forEach(System.out::println);
}
```

กิจกรรมที่ 5

map() ของ stream รับ function interface

5.1 เรียก getName ด้วย lambda expression เพื่อแปลงจาก stream ของ Singer เป็น stream ของ String แล้วค่อยใช้ forEach() เพื่อพิมพ์

5.2 สร้าง Function โดยกำหนดให้เป็นการ getName

ตอบ q5.2

```
public static void q5_streamMap() {
    // Two separate statements
    // Consumer<Singer> singerNames = s ->
System.out.println(s.getName());
    // singerList.forEach(singerNames);

    System.out.println("Q5.1--------");
    singerList.stream().map(s -> s.getName())
              .forEach(System.out::println);

    System.out.println("Q5.2--------");
    singerList.stream().map(/* q5.2 */)
              .forEach(System.out::println);
}
```

ส่ง

กำหนดส่ง TBA