

A. หัดใช้ lambda expression และ method reference

กิจกรรมที่ 1

1.1 กำหนด HalfValueInterface

เป็น functional Interface

...implement method ให้ halfVal

แสดงค่า n/2

ตอบ q1.1

n -> System.out.println(n / 2);

1.2 Consumer เป็น functional

interface ที่รับค่าด้วย method

accept() ...implement ให้

consumer นำ n ไปแสดงค่า n/2

(สังเกตว่า ใช้ Consumer แทนการ

สร้างและ implement

HalfValueInterface

ตอบ q1.2

n -> System.out.println(n / 2);

1.3 forEach() รับ Consumer ...

เพื่อป้องกันความสับสน ให้ halfMe

เป็น Consumer มีพฤติกรรมเหมือน

1.2 ...ดังนั้นส่ง halfMe ให้

forEach()

ตอบ q1.3 **forEach(halfMe);**

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

ตอบ q1.4 **n -> System.out.println(n / 2);**

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

ตอบ q1.5 **np::printHalf**

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

    Consumer<Integer> consumer = n -> /* q1.2 */
    for (int n : nums) {
        consumer.accept(n);
    }

    Consumer<Integer> halfMe = n ->
        System.out.println(n / 2);
    nums./* q1.3 */;

    nums.forEach(/* q1.4 */);

    NumberProcessor np = new NumberProcessor();
    nums.forEach(/* q1.5 */);
}
```

กิจกรรมที่ 2

2.1 map() รับ function interface

ให้เรียก getName ด้วย lambda

expression เพื่อแปลงจาก stream ของ

Singer เป็น stream ของ String แล้ว

ค่อยใช้ forEach() เพื่อพิมพ์

ตอบ q2.1

```
public static void q2_forEachSingerName() {
    singerList.stream().map(/* q2.1 */)
                .forEach(System.out::println);

    singerList.stream().map(/* q2.2 */)
                .forEach(System.out::println);
}
```

```
(singer -> singer.getName()).forEach(System.out::println);
```

2.1 map() รับ function interface ให้เรียก getName ด้วย method reference เพื่อแปลงจาก stream ของ Singer เป็น stream ของ String แล้วค่อยใช้ forEach() เพื่อพิมพ์

ตอบ q2.2

```
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 =
        Collections.sort(singerList, byStyle2);
    singerList.forEach(System.out::println);
}
```

```
(Singer::getName).forEach(System.out::println);
```

กิจกรรมที่ 3

3.1 compare() ใน interface

Comparator จะ return ค่า - , 0 ,

+ สำหรับ 2 ค่าใดๆ เพื่อให้ jvm

ทราบว่าค่าไหนมาก่อน / หลัง

เขียน byStyle2 ด้วย lambda

expression เพื่อให้ sort() เรียง

ข้อมูล singer ใน singerList ตาม

String ของ SingStyle กำหนด public String getStyleString() { return style.toString(); }

ตอบ q3.1

```
Comparator.comparing(singer -> singer.getStyleString());
```

```
Q2.1-----
Aba
Abi
Abo
Abe
Q2.2-----
Aba
Abi
Abo
Abe
Singer (Aba-SingStyle.POP)
Singer (Abo-SingStyle.POP)
Singer (Abi-SingStyle.ROCK)
Singer (Abe-SingStyle.ROCK)
Q3.1-----
Singer (Aba-SingStyle.POP)
Singer (Abo-SingStyle.POP)
Singer (Abi-SingStyle.ROCK)
Singer (Abe-SingStyle.ROCK)
```

กิจกรรมที่ 4

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

4.1 เราสามารถสร้าง Comparator ด้วย Comparator.comparing(Class::Method) ซึ่ง พารามิเตอร์ของ comparing เป็น function interface

เขียน byName ด้วย method reference

Comparator.comparing(Singer::getStyleString);

ตอบ q4.1

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

    singerList.sort( /* q4.2 */ );
    singerList.forEach(System.out::println);
}
```

4.2 เราสามารถสร้าง Compator ให้ singerList เรียงด้วย style (getStyle()) ด้วย lambda expression

ตอบ q4.2

(p, q) -> p.getStyleString().compareTo(q.getStyleString())

Q4.1-----
 Singer (Aba-SingStyle.POP)
 Singer (Abe-SingStyle.ROCK)
 Singer (Abi-SingStyle.ROCK)
 Singer (Abo-SingStyle.POP)
 Q4.2-----
 Singer (Aba-SingStyle.POP)
 Singer (Abo-SingStyle.POP)
 Singer (Abe-SingStyle.ROCK)
 Singer (Abi-SingStyle.ROCK)

ทดลองว่า เราสามารถสร้าง compareTo ให้เทียบเท่า comparator ให้ Collection.sort() หรือ list.sort() ได้หรือไม่

4.3 เรียง singerList ด้วย Collections.sort()

Collections.sort(singerList,
Singer::compareToByStyle); (yes/no) YES

4.4 เรียง singerList ด้วย singerList.sort()

singerList.sort(Singer::compareToByStyle); (yes/no) YES

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

อนึ่ง จริง ๆ ควรเปลี่ยน 4.4 เป็น byStle เพราะ 4.3 กับ 4.4 เรียงเหมือนกัน จึงไม่เห็นความเปลี่ยนแปลง (อาจพลักถูก)

กำหนดส่ง TBA