วัตถุประสงค์

A. เข้าใจโครงสร้าง หลักการ ของ abstract class

B. สามารถบอกความแตกต่างของผลการสืบทอดจาก class กับ จาก abstract class

C. เข้าใจโครงสร้าง หลักการ ของ interface

กิจกรรมที่ 1

1.1 กำหนด interface CanSwimIntf และผลการเรียกใช้

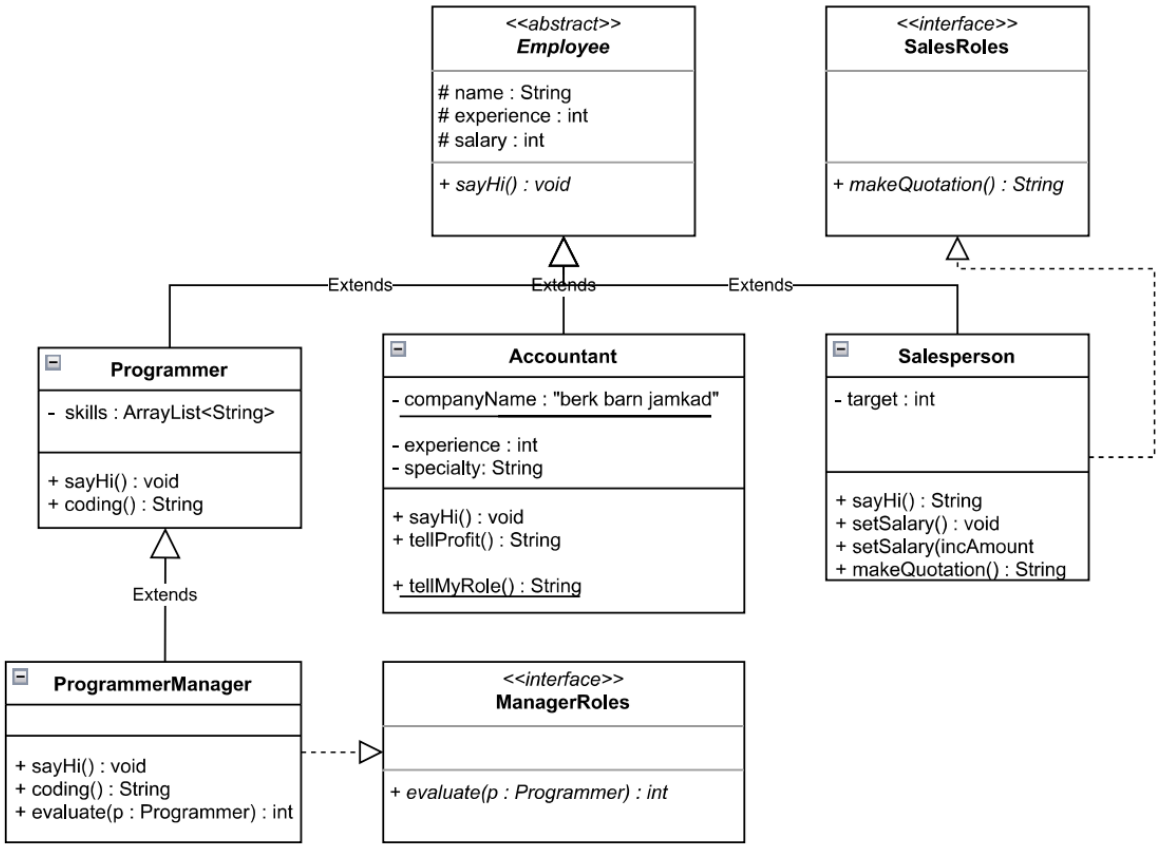
Lec4Fish และ Lec4Duck ดังแสดง

```
1  package packA;
2  public interface CanSwimIntf {
3      public void swim();
4  }
```

1.2 เขียน Lec4Fish.java และ Lec4Duck.java

```
11  static void q1() {
12      Lec4Fish fish = new Lec4Fish();
13      fish.swim();
14      fish.fishMethod();
15      // flexing my tail back and forth
16      // do i?
17      Lec4Duck duck = new Lec4Duck();
18      duck.swim();
19      duck.duckMethod();
20      // waddling
21      // quack
22  }
```

กิจกรรมที่ 2



2.1

ลบ EmpTmp.class และ rename EmpTmp.java เป็น Lab3EmpTmp.java

(เพื่อป้องกันการฟลุ๊ค รัน งานนี้ผ่าน)

2.2 สร้าง Employee.java ให้เป็น abstract class ให้ sayHi() เป็น abstract

2.3 แปลง Programmer, Salesperson, Accountant ให้สืบทอดจาก Employee นี้แทน (ดังนั้น ทั้ง 3 จะ implement sayHi()) และต่างมี method เฉพาะคลาสตนเอง

2.4 เราสามารถสร้าง interface เช่น SalesRoles เพื่อกำหนด methods สำหรับ class ที่เป็นตำแหน่งที่เกี่ยวข้องกับการขาย ต้องมี ...แต่ในที่นี้บังเอิญซ้ำกับที่เราเคยมีใน Salesperson ไปแล้ว ดังนั้นการระบุ implements SalesRoles ก็ไม่ต้องเปลี่ยนแปลงอะไรกับ Salesperson.java

2.5 สร้าง interface ManagerRoles.java แล้วสร้าง class ProgrammerManager ตาม diagram

2.6 public int evaluate(Programmer p) ให้เพิ่ม salary ของ p 15% (int) และแสดง new salary นั้น

```java
    public static void main(String[] args) {
        q1();
        System.out.println(x: "-------------------");
        q2();
    }

    static void q1() {
        ProgrammerManager pmanager = new ProgrammerManager(n: "CodeReviewer", exp: 9, sal: 550);
        System.out.println(pmanager);
        Employee e = pmanager;
        e.sayHi();
    }

    static void q2() {
        ProgrammerManager pmanager = new ProgrammerManager(n: "CodeReviewer", exp: 9, sal: 550);
        ArrayList<Programmer> aList = new ArrayList<>();
        aList.add(new Programmer(n: "Keng", exp: 2, sal: 300));
        aList.add(new Programmer(n: "Somsri", exp: 3, sal: 400));
        aList.add(new Programmer(n: "haha", exp: 4, sal: 600));
        for (Programmer p : aList) {
            int newSalary = pmanager.evaluate(p);
            System.out.println(newSalary);
        }
    }
```

```
ManagerProgrammer [name=CodeReviewer, experience=9salary=550]
Coding in [solidity, typescript]
-------------------
345
460
690
```

สรุปหลักการ interface พอสังเขป

หลักการคล้าย abstract class คือสร้าง interface เพื่อกำหนดโครงสร้างของ method ที่จำเป็นใช้งานแต่ยังไม่ได้กำหนดรายละเอียดการทำงานใดๆ ลงไป ให้กับ method นั้น (abstract method) method ใน interface จึงเป็น method ว่างเปล่า ซึ่งในภายหลังจึงมีการกำหนดรายละเอียดของ method เหล่านั้นลงไป ถูกกำหนดโดยคลาสที่ implenent interface นั้นๆ

กำหนดส่ง TBA