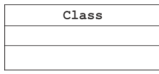
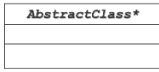
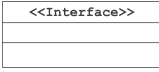


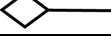


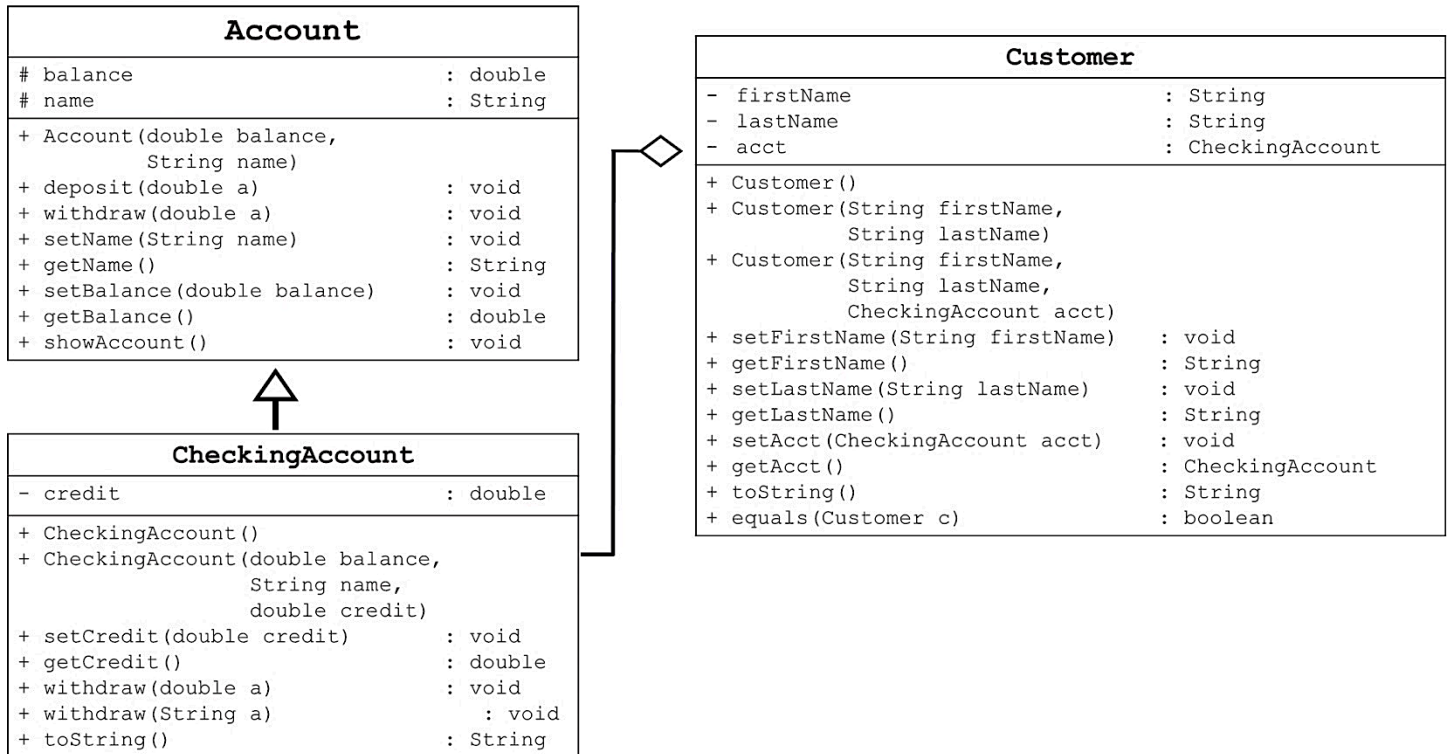
## แบบฝึกปฏิบัติ ครั้งที่ 7

- เรื่อง การเขียนโปรแกรมเชิงวัตถุ
- วัตถุประสงค์
1. เพื่อฝึกฝนการพัฒนาคلاسไม่สมบูรณ์ (Abstract class)
  2. เพื่อฝึกฝนการพัฒนาอินเตอร์เฟซ (Interface)
  3. เพื่อฝึกฝนการพัฒนาคอนสตรัคเตอร์เมธอด (Constructor)

คำอธิบายเครื่องหมายดังแสดงในตารางต่อไปนี้

ลำดับ	สัญลักษณ์	ความหมาย
1	+	เป็นการกำหนด access mode เป็น public
2	-	เป็นการกำหนด access mode เป็น private
3	#	เป็นการกำหนด access mode เป็น protected
4		บ่งบอกว่าเป็นคลาสปกติ
5		<b>ชื่อคลาสเป็นตัวเอียงหนา</b> บ่งบอกว่าเป็นคลาสไม่สมบูรณ์
6		บ่งบอกว่าเป็นอินเตอร์เฟซ
7		บ่งบอกการสืบทอด extends
8		บ่งบอกการ implements
9		บ่งบอกการเป็นส่วนหนึ่ง composite
10	<i>methodname () *</i>	<b>ตัวเอียง</b> บ่งบอกว่าเป็นเมธอดไม่สมบูรณ์
11	<b>attribute</b> หรือ <b>methodname ()</b>	<b>ตัวหนา</b> บ่งบอกความเป็น final
12	<u>attribute</u> หรือ <u>methodname ()</u>	<u>ตัวขีดเส้นใต้</u> บ่งบอกความเป็น static

ข้อที่ 1 คำสั่งให้นักศึกษาร่างคลาส Account, CheckingAccount และ Customer จากคลาสไดอะแกรมต่อไปนี้



1.1. ให้นักศึกษาร่างคลาส Account โดยกำหนดให้แต่ละเมธอดมีรายละเอียดดังต่อไปนี้

- คอนสตรัคเตอร์เมธอด Account(double balance, String name) จะนำค่าพารามิเตอร์ balance ไปกำหนดให้แอททริบิวต์ balance และพารามิเตอร์ name ไปกำหนดให้แอททริบิวต์ name
- เมธอด setName() จะนำค่าพารามิเตอร์ name ไปกำหนดให้แอททริบิวต์ name ขณะที่ เมธอด setBalance() จะนำค่าพารามิเตอร์ balance ไปกำหนดให้ค่าเป็นแอททริบิวต์ balance
- เมธอด deposit() จะนำค่า a ไปเพิ่มจากแอททริบิวต์ balance (balance + a) ก็ต่อเมื่อ a ต้องมีค่ามากกว่า 0 ถ้าใช้ให้แสดงข้อความดังต่อไปนี้

[ค่าจากตัวแปร a] baht is deposited to [ค่าจากแอททริบิวต์ name].

ถ้าไม่ใช้ให้แสดงข้อความดังต่อไปนี้

Input number must be a positive integer.

- เมธอด withdraw() จะนำค่า a ไปลบจากแอททริบิวต์ balance (balance - a) ก็ต่อเมื่อ a ต้องมีค่ามากกว่า 0 และผลลัพธ์ของ balance - a บวกต้องมากกว่า 0 ถ้าใช้ให้แสดงข้อความดังต่อไปนี้

[ค่าจากตัวแปร a] baht is withdrawn from [ค่าจากแอททริบิวต์ name].

ถ้าไม่ใช้ให้แสดงข้อความดังต่อไปนี้

กรณี  $a < 0$

Input number must be a positive integer.

กรณี  $(\text{balance} - a) < 0$

Not enough money!

- เมธอด `getName()` จะคืนค่าเป็นแอททริบิวต์ `name` ขณะที่ เมธอด `getBalance()` จะคืนค่าเป็นแอททริบิวต์ `balance`
- เมธอด `showAccount()` จะแสดงค่าของแต่ละแอททริบิวต์ดังตัวอย่าง

[ค่าจากแอททริบิวต์ `name`] account has [ค่าจากแอททริบิวต์ `balance`] baht.

กำหนดโค้ดสำหรับทดสอบความถูกต้องของคลาสข้างต้นที่นักศึกษาได้พัฒนาขึ้น

กรณีที่ 1

```
public class Main {
    public static void main(String[] args) {
        Account a1 = new Account();
        a1.showAccount();
    }
}
```

ตัวอย่างผลลัพธ์

```
error: constructor Account in class Account cannot be applied to given types;
Account a1 = new Account();
required: double,String
```

กรณีที่ 2

```
public class Main {
    public static void main(String[] args) {
        Account a1 = new Account(50000, "61070033");
        a1.showAccount();
        a1.deposit(500);
        a1.showAccount();
        a1.withdraw(40000);
        a1.showAccount();
    }
}
```

ตัวอย่างผลลัพธ์

```
61070033 account has 50000.0 baht.
500.0 baht is deposited to 61070033.
61070033 account has 50500.0 baht.
40000.0 baht is withdrawn from 61070033.
61070033 account has 10500.0 baht.
```

1.2. ให้นักศึกษาร่างคลาส `CheckingAccount` โดยกำหนดให้แต่ละเมธอดมีรายละเอียดดังต่อไปนี้

- คอนสตรัคเตอร์เมธอด `CheckingAccount()` หรือเรียกว่า default constructor จะกำหนดค่าแอททริบิวต์ `balance` และ `name` เป็น 0 และ “ ” ตามลำดับ โดยเรียกใช้ constructor ของคลาส `Account` ได้แก่ `Account(double balance, String name)` และ `credit` ให้กำหนดเป็น 0

- คอนสแตนต์เมธอด CheckingAccount(double balance, String name, double credit) จะนำค่าพารามิเตอร์ balance ไปกำหนดให้แอตทริบิวต์ balance และพารามิเตอร์ name ไปกำหนดให้แอตทริบิวต์ name โดยอาศัยเมธอด super() จากนั้นนำค่าพารามิเตอร์ credit ไปกำหนดให้แอตทริบิวต์ credit
- เมธอด setCredit() จะนำค่าพารามิเตอร์ credit ไปกำหนดให้แอตทริบิวต์ credit ก็ต่อเมื่อพารามิเตอร์ credit ต้องมีค่ามากกว่า 0 ถ้าไม่ใช่ให้แสดงข้อความดังต่อไปนี้

Input number must be a positive integer.

- เมธอด getCredit() จะคืนค่าเป็นแอตทริบิวต์ credit
- เมธอด withdraw(double a) และ withdraw(String a) จะถูก overridden โดยการนำค่า a ไปลดจากแอตทริบิวต์ balance และ credit ก็ต่อเมื่อ a ต้องมีค่ามากกว่า 0 และ

เงื่อนไข		การกระทำ
1	(balance - a) แล้วยังมียอดเงินคงเหลือ <b>ไม่ติดลบ</b>	ให้ทำงานเหมือน withdraw() ของคลาส Account
2	(balance - a) แล้วยอดเงินคงเหลือ <b>ติดลบ</b> แต่เมื่อรวมกับ credit แล้วยอดเงินคงเหลือ <b>ไม่ติดลบ</b>	เงินใน balance จะกำหนดให้เท่ากับ 0 และไปหักส่วนต่างจากเครดิตแทน
3	(balance - a) แล้วยอดเงินคงเหลือ <b>ติดลบ</b> แล้วเมื่อรวมกับ credit แล้วยอดเงินคงเหลือก็ยังคง <b>ติดลบ</b>	-

แสดงข้อความดังต่อไปนี้ กรณีตรงตามเงื่อนไขที่ 1 หรือ 2

[ค่าจากตัวแปร a] baht is withdrawn from [ค่าจากแอตทริบิวต์ name] and your credit balance is [ค่าจากแอตทริบิวต์ credit].

แสดงข้อความดังต่อไปนี้ กรณีตรงตามเงื่อนไขที่ 3

Not enough money!

- เมธอด toString() จะถูก overridden ซึ่งจะคืนค่าเป็น

The [แอตทริบิวต์ name] account has [แอตทริบิวต์ balance] baht and [แอตทริบิวต์ credit] credits.

กำหนดโค้ดสำหรับทดสอบความถูกต้องของคลาสข้างต้นที่นักศึกษาได้พัฒนาขึ้น

กรณีที่ 1

```
public class Main {
    public static void main(String[] args) {
        CheckingAccount a1 = new CheckingAccount(50000, "61070033", 5000);
        a1.showAccount();
        a1.deposit(500);
        System.out.println(a1);
        a1.withdraw("40000.0");
        System.out.println(a1.toString());
    }
}
```

ตัวอย่างผลลัพธ์

```
61070033 account has 50000.0 baht.
500.0 baht is deposited to 61070033.
The 61070033 account has 50500.0 baht and 5000.0 credits.
40000.0 baht is withdrawn from 61070033 and your credit balance is 5000.0.
The 61070033 account has 10500.0 baht and 5000.0 credits.
```

## กรณีที่ 2

```
public class Main {  
    public static void main(String[] args) {  
        CheckingAccount a1 = new CheckingAccount();  
        a1.deposit(500);  
        System.out.println(a1);  
        a1.withdraw(40000.0);  
        System.out.println(a1.toString());  
    }  
}
```

## ตัวอย่างผลลัพธ์

```
500.0 baht is deposited to .  
The account has 500.0 baht and 0.0 credits.  
Not enough money!  
The account has 500.0 baht and 0.0 credits.
```

## กรณีที่ 3

```
public class Main {  
    public static void main(String[] args) {  
        Account a1 = new CheckingAccount();  
        a1.setCredit(1000);  
        System.out.println(a1);  
    }  
}
```

## ตัวอย่างผลลัพธ์

```
Main.java:4: error: cannot find symbol  
a1.setCredit(1000);  
symbol:   method setCredit(int)
```

## กรณีที่ 4

```
public class Main {  
    public static void main(String[] args) {  
        Account a1 = new CheckingAccount();  
        a1.setName("Nook");  
        System.out.println(a1);  
        ((CheckingAccount)a1).setCredit(1000);  
        System.out.println(a1);  
    }  
}
```

## ตัวอย่างผลลัพธ์

```
The Nook account has 0.0 baht and 0.0 credits.  
The Nook account has 0.0 baht and 1000.0 credits.
```

## 1.3. ให้นักศึกษาร่างคลาส Customer โดยกำหนดให้แต่ละเมธอดมีรายละเอียดดังต่อไปนี้

- คอนสตรัคเตอร์เมธอด Customer(String firstName, String lastName, CheckingAccount acct) จะนำค่าพารามิเตอร์ firstName ไปกำหนดให้แอตทริบิวต์ firstName และพารามิเตอร์ lastName ไปกำหนดให้แอตทริบิวต์ lastName นอกจากนี้ นำค่าพารามิเตอร์ acct ไปกำหนดให้แอตทริบิวต์ acct
- คอนสตรัคเตอร์เมธอด Customer(String firstName, String lastName) จะนำค่าพารามิเตอร์ firstName ไปกำหนดให้แอตทริบิวต์ firstName และพารามิเตอร์ lastName ไปกำหนดให้แอตทริบิวต์

lastName นอกจากนี้ นำ null ไปกำหนดให้แอททริบิวต์ acct โดยต้องเรียกใช้งาน Customer (String firstName, String lastName, CheckingAccount acct) ผ่านคำสั่ง this(...)

- คอนสตรัคเตอร์เมธอด Customer() หรือเรียกว่า default constructor จะกำหนดค่าแอททริบิวต์ firstName, lastName และ acct เป็น "", "" และ null ตามลำดับ กำหนดให้ต้องเรียกใช้งาน Customer (String firstName, String lastName, CheckingAccount acct)
- เมธอด setFirstName() จะนำค่าพารามิเตอร์ firstName ไปกำหนดให้แอททริบิวต์ firstName และเมธอด setLastName() จะนำค่าพารามิเตอร์ lastName ไปกำหนดให้แอททริบิวต์ lastName และเมธอด setAcct() จะนำค่าพารามิเตอร์ acct ไปกำหนดให้แอททริบิวต์ acct
- เมธอด getFirstName() จะคืนค่าเป็นแอททริบิวต์ firstName และเมธอด getLastName() จะคืนค่าเป็นแอททริบิวต์ lastName และ เมธอด getAcct() จะคืนค่าเป็นแอททริบิวต์ acct
- เมธอด toString() จะถูก overridden ซึ่งจะคืนค่าเป็น

ข้อความดังต่อไปนี้ กรณีที่ค่าของ acct มีค่าเป็น null

```
[แอททริบิวต์ firstName] [แอททริบิวต์ lastName] doesn't have account.
```

ข้อความดังต่อไปนี้ กรณีที่ค่าของ acct มีค่าไม่เป็น null

```
The [แอททริบิวต์ firstName] account has [แอททริบิวต์ balance] baht and [แอททริบิวต์ credit] credits.
```

- เมธอด equals (Customer c) จะถูก overloaded ซึ่งจะคืนค่าเป็น true ก็ต่อเมื่อแอททริบิวต์ firstName และ lastname ของวัตถุทั้ง 2 เหมือนกันถ้าไม่ใช่จะคืนค่าเป็น false

กำหนดโค้ดสำหรับทดสอบความถูกต้องของคลาสข้างต้นที่นักศึกษาได้พัฒนาขึ้น

กรณีที่ 1

```
public class Main {
    public static void main(String[] args) {
        CheckingAccount a1 = new CheckingAccount(1000,"62070033",500);
        Customer c1 = new Customer();
        Customer c2 = new Customer("Harry","Potter");
        Customer c3 = new Customer("Harry","Potter",a1);

        System.out.println(c2);
        System.out.println(c3);
    }
}
```

ตัวอย่างผลลัพธ์

```
Harry Potter doesn't have account.
The Harry account has 1000.0 baht and 500.0 credits.
```

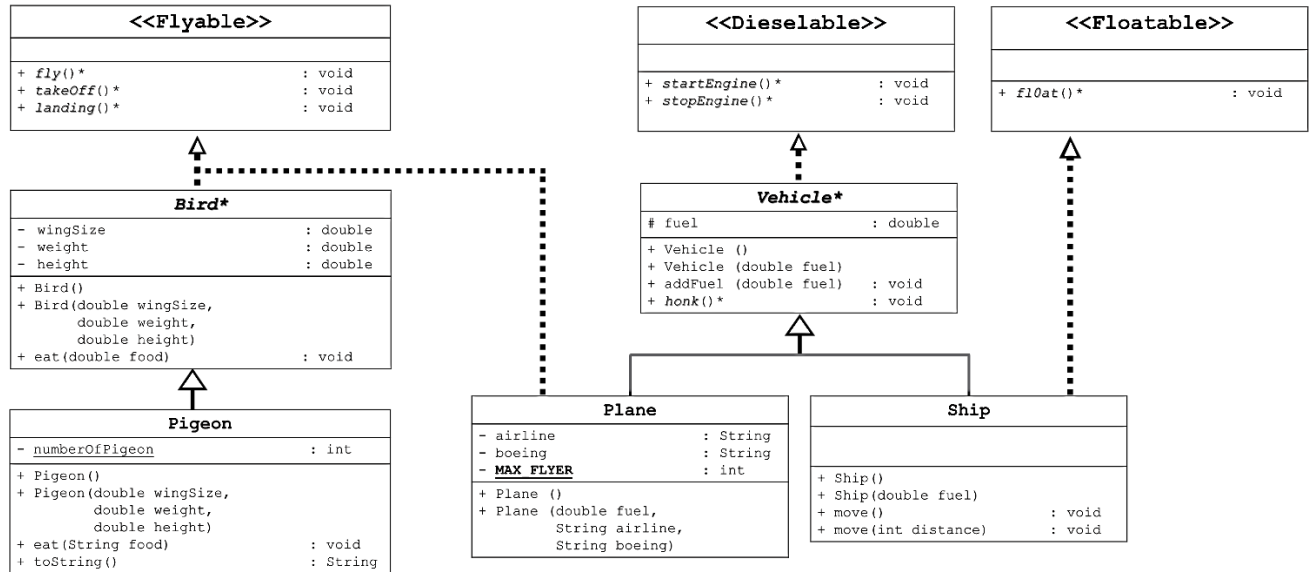
## กรณีที่ 2

```
public class Main {  
    public static void main(String[] args) {  
        CheckingAccount a1 = new CheckingAccount(1000,"62070033",500);  
        Customer c1 = new Customer();  
        Customer c2 = new Customer("Harry","Potter");  
        Customer c3 = new Customer("Harry","Potter",a1);  
        System.out.println(c1.equals(c2));  
        System.out.println(c3.equals(c2));  
    }  
}
```

## ตัวอย่างผลลัพธ์

```
false  
true
```

ข้อที่ 2 ให้นักศึกษาใช้เพื่อสร้างคลาส Bird, Pigeon, Vehicle, Plane, และ Ship นอกจากนี้ ให้นักศึกษาสร้างอินเทอร์เฟซ <<Flyable>>, <<Dieselable>>, และ <<Floatable>> จากคลาสไดอะแกรมต่อไปนี้



2.1. ให้นักศึกษาสร้างอินเทอร์เฟซ <<Flyable>>, <<Dieselable>> , และ <<Floatable>> ดังแสดงในคลาสไดอะแกรม และกำหนดโค้ดสำหรับทดสอบความถูกต้องของอินเทอร์เฟซที่นักศึกษาได้พัฒนาขึ้นดังนี้

โค้ด

```

class Main {
    public static void main(String[] args) {
        Flyable f1 = new Flyable();
        Floatable f2 = new Floatable();
        Dieselable d = new Dieselable();
    }
}
  
```

ตัวอย่างผลลัพธ์

```

Main.java:3: error: Flyable is abstract; cannot be instantiated
    Flyable f1 = new Flyable();
Main.java:4: error: Floatable is abstract; cannot be instantiated
    Floatable f2 = new Floatable();
Main.java:5: error: Dieselable is abstract; cannot be instantiated
    Dieselable d = new Dieselable();
  
```

เนื่องจากสาเหตุใดทำไมจึงเกิดข้อความ Error ดังกล่าว

object ทั้ง ๑ ตัวถูกสร้างขึ้นเพื่อเป็น interface ซึ่งมีแค่ abstract method และ method ที่ไม่ได้ถูกสร้างไว้ก่อนการรันโปรแกรม runtime ไร้



2.2. ให้นักศึกษาสร้างคลาสไม่สมบูรณ์ (Abstract Class) ต่อไปนี้ ได้แก่ Bird และคลาสสมบูรณ์ Pigeon โดยกำหนดให้แต่ละเมธอดมีรายละเอียดดังต่อไปนี้

### คลาสไม่สมบูรณ์ Bird

- กำหนดให้สร้างเมธอดประเภท setter() และ getter() แบบปกติของทุกแอททริบิวต์
- กำหนดให้สร้างเมธอด eat() โดยจะนำค่า food ที่รับเข้ามาไปบวกเพิ่มในแอททริบิวต์ weight โดยที่ค่า food ต้องมากกว่า 0 ถ้าไม่ใช่ให้แสดงข้อความว่า

Input cannot be negative number.

- กำหนดให้สร้างคอนสตรัคเตอร์เมธอด Bird(double wingSize, double weight, double height) ซึ่งจะ
  - นำค่าพารามิเตอร์ wingSize ไปกำหนดให้แอททริบิวต์ wingSize
  - นำค่าพารามิเตอร์ weight ไปกำหนดให้แอททริบิวต์ weight
  - นำค่าพารามิเตอร์ height ไปกำหนดให้แอททริบิวต์ height
- กำหนดให้สร้างคอนสตรัคเตอร์เมธอด Bird() ซึ่งจะกำหนดให้ทุกแอททริบิวต์มีค่าเป็น 0.0 โดยอาศัยคอนสตรัคเตอร์เมธอด Bird(double wingSize, double weight, double height) ผ่านคำสั่ง this(...)

### คลาสสมบูรณ์ Pigeon

- กำหนดให้สร้างคอนสตรัคเตอร์เมธอด Pigeon (double wingSize, double weight, double height) ซึ่งจะ
  - นำค่าพารามิเตอร์ wingSize ไปกำหนดให้แอททริบิวต์ wingSize
  - นำค่าพารามิเตอร์ weight ไปกำหนดให้แอททริบิวต์ weight
  - นำค่าพารามิเตอร์ height ไปกำหนดให้แอททริบิวต์ height

โดยเรียกใช้คอนสตรัคเตอร์คลาสแม่ (bird) ที่สอดคล้องกันผ่านคำสั่ง super(...) จากนั้น ค่าของแอททริบิวต์ numberOfPigeon จะเพิ่มขึ้นจากเดิม 1

- กำหนดให้สร้างคอนสตรัคเตอร์เมธอด Pigeon () ซึ่งจะกำหนดให้ทุกแอททริบิวต์มีค่าเป็น 0.0 โดยเรียกใช้งานคอนสตรัคเตอร์เมธอด Pigeon (double wingSize, double weight, double height) ผ่านคำสั่ง this(...)
- กำหนดให้สร้างเมธอด eat() โดยจะนำค่าในพารามิเตอร์ food ไปตรวจสอบ
  - ถ้าเป็น "worm" ให้บวกค่าแอททริบิวต์ weight เพิ่ม 0.5
  - ถ้าเป็น "seed" ให้บวกค่าแอททริบิวต์ weight เพิ่ม 0.2
  - ถ้าไม่ใช่ข้างต้นให้พิมพ์ข้อความว่า

Pigeon can eat only worm and seed.

- เมธอด toString() จะถูก overridden ซึ่งจะคืนค่าเป็น

Pigeon [แอททริบิวต์ weight] kg and [แอททริบิวต์ height] cm. There are [แอททริบิวต์ numberOfPigeon] pigeons.

- กำหนดให้สร้างเมธอด fly() ให้สมบูรณ์ โดยที่เมธอดดังกล่าวจะลดค่าแอททริบิวต์ weight ลง 0.25 และแสดงข้อความต่อไปนี้ **ก็ต่อเมื่อค่าแอททริบิวต์ weight มากกว่าหรือเท่ากับ 5**

Fly Fly

ถ้าไม่ใช่ เมธอดนี้จะแสดงข้อความต่อไปนี้

I'm hungry.

- กำหนดให้สร้างเมธอด takeOff() ให้สมบูรณ์ โดยที่เมธอดดังกล่าวจะลดค่าแอททริบิวต์ weight ลง 0.5 และแสดงข้อความต่อไปนี้ ก็ต่อเมื่อค่าแอททริบิวต์ weight มากกว่าหรือเท่ากับ 5

Take Off

ถ้าไม่ใช่เมธอดนี้จะแสดงข้อความต่อไปนี้

I'm hungry.

- กำหนดให้สร้างเมธอด landing() ให้สมบูรณ์ โดยที่เมธอดดังกล่าวจะลดค่าแอททริบิวต์ weight ลง 0.5 และแสดงข้อความต่อไปนี้ ก็ต่อเมื่อค่าแอททริบิวต์ weight มากกว่าหรือเท่ากับ 5

Landing

ถ้าไม่ใช่เมธอดนี้จะแสดงข้อความต่อไปนี้

I'm hungry.

กำหนดโค้ดสำหรับทดสอบความถูกต้องของอินเทอร์เฟซที่นักศึกษาได้พัฒนาขึ้นดังนี้  
โค้ด

```
public class Main {
    public static void main(String[] args) {
        Pigeon p1 = new Pigeon(10,15,20);
        System.out.println(p1);
        p1.takeOff();
        System.out.println(p1);
        p1.fly();
        System.out.println(p1);
        p1.landing();
        System.out.println(p1);
        p1.eat("seed");
        System.out.println(p1);
        Pigeon p3 = new Pigeon();
        System.out.println(p3);
    }
}
```

ตัวอย่างผลลัพธ์

```
Pigeon 15.0 kg and 20.0 cm. There are 1 pigeons.
Take Off
Pigeon 14.5 kg and 20.0 cm. There are 1 pigeons.
Fly Fly
Pigeon 14.25 kg and 20.0 cm. There are 1 pigeons.
Landing
Pigeon 13.75 kg and 20.0 cm. There are 1 pigeons.
Pigeon 13.95 kg and 20.0 cm. There are 1 pigeons.
Pigeon 0.0 kg and 0.0 cm. There are 2 pigeons.
```

2.3. ให้นักศึกษาสร้างคลาสไม่สมบูรณ์ (Abstract Class) ต่อไปนี้ ได้แก่ Vehicle และคลาสสมบูรณ์ Plane และ Ship โดยกำหนดให้แต่ละเมธอดมีรายละเอียดดังต่อไปนี้

#### คลาส Vehicle

- กำหนดให้สร้างเมธอดประเภท setter () และ getter () แบบปกติของทุกแอททริบิวต์
- กำหนดให้สร้างเมธอด addFuel () โดยจะนำค่าพารามิเตอร์ fuel ที่รับเข้ามาไปบวกเพิ่มในแอททริบิวต์ fuel ก็ต่อเมื่อค่าพารามิเตอร์ fuel ต้องมากกว่า 0 แต่ถ้าไม่ใช่ให้แสดงข้อความต่อไปนี้

Fuel is empty.

- กำหนดให้สร้างคอนสแตนต์เมธอด Vehicle(double fuel) ซึ่งจะนำพารามิเตอร์ fuel ไปกำหนดให้แอททริบิวต์ fuel
- กำหนดให้สร้างคอนสแตนต์เมธอด Vehicle () ซึ่งจะกำหนดให้แอททริบิวต์ fuel เป็น 0.0
- กำหนดให้ประกาศเมธอด honk () เป็นเมธอดแบบไม่สมบูรณ์

#### คลาส Ship

- กำหนดให้สร้างคอนสแตนต์เมธอด Ship(double fuel) ซึ่งจะนำพารามิเตอร์ fuel ไปกำหนดให้แอททริบิวต์ fuel โดยอาศัยคอนสแตนต์เมธอดของคลาสแม่ที่สอดคล้องกัน
- กำหนดให้สร้างคอนสแตนต์เมธอด Ship () ซึ่งจะกำหนดให้แอททริบิวต์ fuel เป็น 0.0 โดยอาศัยคอนสแตนต์เมธอดของคลาสแม่ที่สอดคล้องกัน
- กำหนดให้สร้างเมธอด float () ให้สมบูรณ์ โดยที่เมธอดดังกล่าวจะลดค่าของแอททริบิวต์ fuel ลง 50 ก็ต่อเมื่อค่าของแอททริบิวต์ fuel มากกว่าหรือเท่ากับ 50 จากนั้น จะแสดงข้อความว่า

Ship moves

แต่ถ้าค่าของแอททริบิวต์ fuel น้อยกว่า 50 จะทำแค่แสดงข้อความว่า

Fuel is not enough.

- กำหนดให้สร้างเมธอด startEngine () ให้สมบูรณ์ โดยที่เมธอดดังกล่าวจะลดค่าของแอททริบิวต์ fuel ลง 10 ก็ต่อเมื่อค่าของแอททริบิวต์ fuel มากกว่าหรือเท่ากับ 10 จากนั้น ค่อยแสดงข้อความว่า

Engine starts

แต่ ถ้าไม่ค่าของแอททริบิวต์ fuel น้อยกว่า 10 จะทำแค่แสดงข้อความว่า

Fuel is not enough.

- กำหนดให้สร้างเมธอด stopEngine () ให้สมบูรณ์ โดยที่เมธอดดังกล่าวจะแสดงข้อความว่า

Engine stops

- กำหนดให้สร้างเมธอด honk () ให้สมบูรณ์ โดยที่เมธอดดังกล่าวจะแสดงข้อความว่า

Shhhhhh

- กำหนดให้สร้างเมธอด move () ให้สมบูรณ์ โดยที่เมธอดดังกล่าวจะเรียกใช้งานเมธอด float () จำนวนหนึ่งครั้ง

- กำหนดให้สร้างเมธอด `move(int distance)` ให้สมบูรณ์ โดยที่เมธอดดังกล่าวจะเรียกใช้งานเมธอด `float()` จำนวนตามค่าในตัวแปร `distance` ครั้ง เนื่องจากมีการเรียกใช้งานเมธอด `float()` ดังนั้น ถ้ากรณีค่าของแอททริบิวต์ `fuel` น้อยกว่า 50 โปรแกรมจะหยุดการทำงานพร้อมแสดงข้อความต่อไปนี้

Fuel is not enough.

### คลาส Plane

- กำหนดให้สร้างคอนสแตนต์เมธอด `Plane(double fuel, String airline, String boeing)` ซึ่งจะนำพารามิเตอร์ `fuel` ไปกำหนดให้แอททริบิวต์ `fuel` และ นำพารามิเตอร์ `airline` และ `boeing` ไปกำหนดให้แอททริบิวต์ `airline` และ `boeing` ตามลำดับ
- กำหนดให้สร้างคอนสแตนต์เมธอด `Plane()` ซึ่งจะกำหนดให้แอททริบิวต์ `fuel`, `airline` และ `boeing` เป็น 0.0, "" และ "" ตามลำดับ
- กำหนดให้สร้างเมธอด `setter()` และ `getter()` ของทุกแอททริบิวต์ ยกเว้น `MAX_FLYER` ที่ไม่มีเมธอด `setter()` และ `getter()` นอกจากนี้ `MAX_FLYER` มีค่าเท่ากับ 2
- กำหนดให้สร้างเมธอด `startEngine()` ให้สมบูรณ์ โดยที่ ถ้าค่าของแอททริบิวต์ `fuel` มากกว่าหรือเท่ากับ 20 แล้ว ค่าของแอททริบิวต์ `fuel` จะลดลง 20 และแสดงข้อความว่า

Plane's Engine starts

แต่ ถ้าค่าของแอททริบิวต์ `fuel` น้อยกว่า 20 แล้วจะแสดงข้อความว่า

Fuel is not enough.

- กำหนดให้สร้างเมธอด `stopEngine()` ให้สมบูรณ์ โดยที่เมธอดดังกล่าวจะแสดงข้อความว่า

Plane's Engine stops

- กำหนดให้สร้างเมธอด `honk()` ให้สมบูรณ์ โดยที่เมธอดดังกล่าวจะแสดงข้อความว่า

Weeeeeeee

- กำหนดให้สร้างเมธอด `fly()` ให้สมบูรณ์ โดยที่ ถ้าค่าของแอททริบิวต์ `fuel` มากกว่าหรือเท่ากับ 20 แล้ว ค่าของแอททริบิวต์ `fuel` จะลดลง 20 และแสดงข้อความว่า

Plane Fly

แต่ ถ้าค่าของแอททริบิวต์ `fuel` น้อยกว่า 20 แล้วจะแสดงข้อความว่า

Fuel is nearly empty.

- กำหนดให้สร้างเมธอด `takeOff()` ให้สมบูรณ์ ถ้าค่าของแอททริบิวต์ `fuel` มากกว่าหรือเท่ากับ 10 แล้ว ค่าของแอททริบิวต์ `fuel` จะลดลง 10 และแสดงข้อความว่า

Plane Already to Take Off

แต่ ถ้าค่าของแอททริบิวต์ `fuel` น้อยกว่า 10 แล้วจะแสดงข้อความว่า

Fuel is nearly empty.

- กำหนดให้สร้างเมธอด `landing()` ให้สมบูรณ์ ถ้าค่าของแอททริบิวต์ `fuel` มากกว่าหรือเท่ากับ 10 แล้ว ค่าของแอททริบิวต์ `fuel` จะลดลง 10 และแสดงข้อความว่า

Plane Already to Landing

แต่ ถ้าค่าของแอททริบิวต์ fuel น้อยกว่า 10 แล้วจะแสดงข้อความว่า

Fuel is nearly empty.

กำหนดโค้ดสำหรับทดสอบความถูกต้องของอินเทอร์เฟซที่นักศึกษาได้พัฒนาขึ้นดังนี้

โค้ด

```
public class Main {
    public static void main(String[] args) {
        Plane p1 = new Plane(200,"IT Airline","FX-747");
        Ship s1 = new Ship(200);

        System.out.println("=== plane ===");
        p1.startEngine();
        p1.takeOff();
        p1.fly();
        p1.fly();
        p1.honk();
        p1.landing();
        p1.stopEngine();

        System.out.println("=== ship ===");
        s1.startEngine();
        s1.move(2);
        s1.honk();
        s1.stopEngine();

    }
}
```

ตัวอย่างผลลัพธ์

```
=== plane ===
Plane's Engine starts
Plane Already to Take Off
Plane Fly
Plane Fly
Weeeeeeee
Plane Already to Landing
Plane's Engine stops
=== ship ===
Engine starts
Ship moves
Ship moves
Shhhhhh
Engine stops
```