แบบฝึกปฏิบัติการครั้งที่ 11

1. กำหนดอินเทอร์เฟซ SimpleQueue, คลาส Product, ผลการรันโปรแกรม และคลาส InterfaceTester ดังนี้

```
public interface SimpleQueue {
          void enqueue(Object o);
          void dequeue();
      }
```

```
public class Product {
    private String name;
    private double price;
    public Product(String n, double p) {
        name = n;
        price = p;
    }
    public String getName() {
        return name;
    }
    public double getPrice() {
        return price;
    }
}
```

ผลการรันโปรแกรม

```
ออเจ้าเอย is added in queue
เธอหนอเธอ is added in queue
เพียงสบตา is added in queue
บุพเพสันนิวาส is added in queue
Now playing ออเจ้าเอย
Now playing เธอหนอเธอ
Now playing เพียงสบตา
Now playing บุพเพสันนิวาส
Pen is added in queue
Pencil is added in queue
Ruler is added in queue
Eraser is added in queue
Pencil Box is added in queue
Total amount = 100.0
```

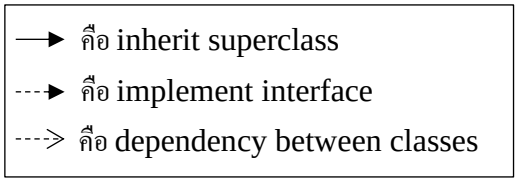
```
public class InterfaceTester {
    public static void main(String[] args) {
        MusicBox m = new MusicBox();
        m.enqueue("ออเจ้าเอย");
        m.enqueue("เธอหนอเธอ");
        m.enqueue("เพียงสบตา");
        m.enqueue("บุพเพสันนิวาส");
        m.dequeue();
        m.dequeue();
        m.dequeue();
        m.dequeue();
        SelfCheckOut s = new SelfCheckOut();
        s.enqueue(new Product("Pen", 25));
        s.enqueue(new Product("Pencil", 10));
        s.enqueue(new Product("Ruler", 20));
```

```
        s.enqueue(new Product("Eraser", 15));
        s.enqueue(new Product("Pencil Box", 30));
        s.dequeue();
        s.dequeue();
        s.dequeue();
        s.dequeue();
        s.dequeue();
        System.out.println("Total amount = " + s.getAmount());
    }
}
```

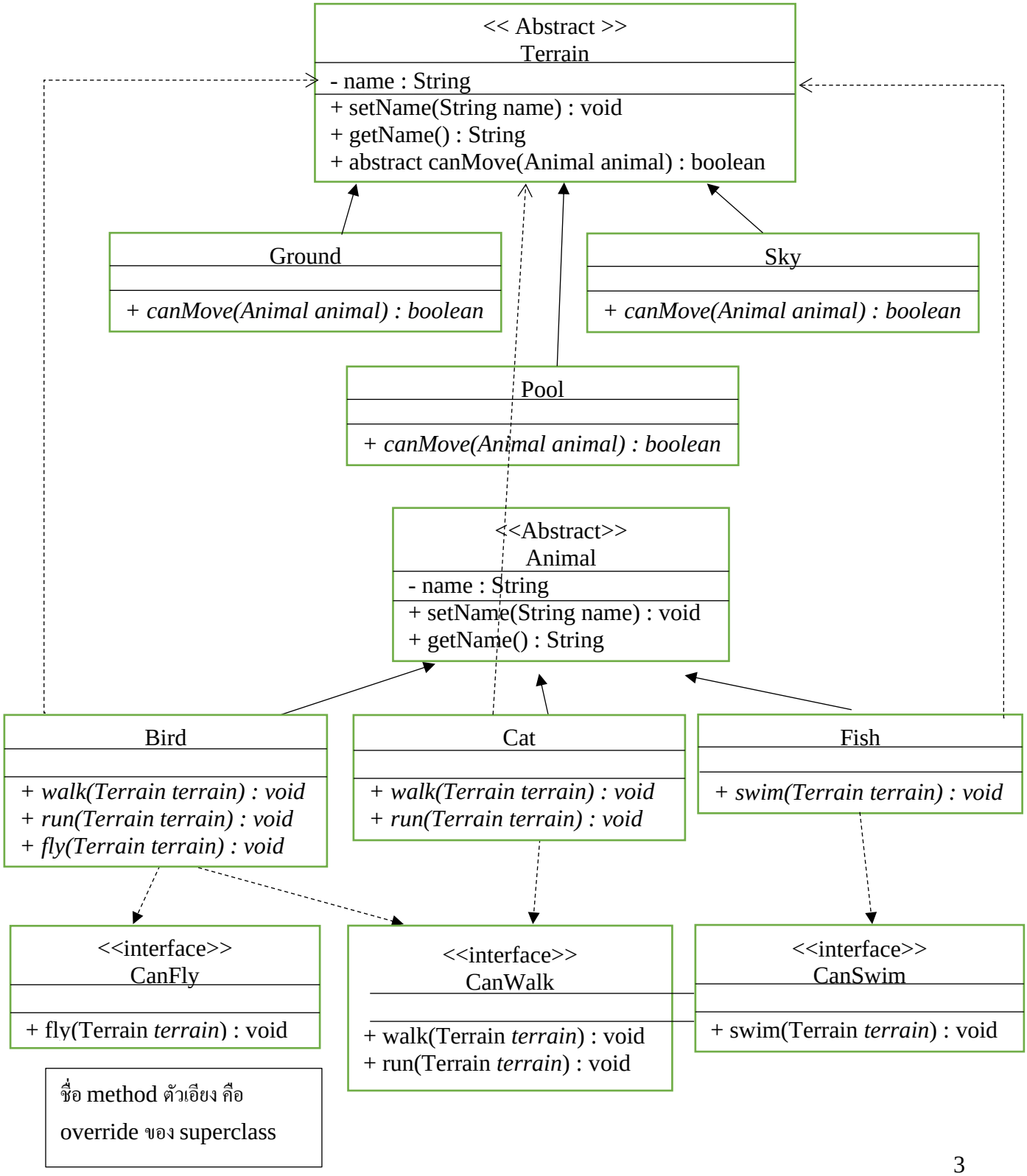
<u>อธิบายเพิ่มเติม</u> การเก็บข้อมูลใส่คิว จะใส่เรียงตามลำดับการเพิ่มข้อมูลเข้ามาในคิว และเมื่อเอาของออกจากคิวจะดึงเอาตัวแรกสุดออกมาก่อน เช่นในตัวอย่างการรัน ครูเพิ่มเพลงในคิว (เทียบกับการเก็บใน playlist) เป็น ออเจ้าเอย เธอหนอเธอ ... ตอนเอาของออกจากคิวมาเล่น ก็เอาเพลงแรกคือ ออเจ้าเอย ออกมาเล่นก่อน แล้วเพลงต่อไปก็คือ เธอหนอเธอ ... ตามลำดับ

จงสร้างคลาส MusicBox สืบทอดจาก SimpleQueue เพื่อจำลองเป็นตู้เพลง มีอาร์เรย์ลิสต์เก็บข้อมูลเพลงตามคิวที่ใส่เข้ามา โดยให้ override method enqueue() เพื่อรับชื่อเพลงเอามาใส่ในคิวเพลง (อาร์เรย์ลิสต์) ที่ต้องเล่น แล้วแสดงผลลัพธ์ว่า ชื่อเพลง is added in queue และ override method dequeue() เพื่อเอาเพลงออกจากคิว (อาร์เรย์ลิสต์) ไปเล่น และแสดงผลลัพธ์ว่า Now playing ชื่อเพลง

จงสร้างคลาส SelfCheckOut สืบทอดจาก SimpleQueue เพื่อจำลองเป็นเครื่องชำระเงินค่าสินค้าด้วยตนเอง (จินตนาการเป็นแบบที่เราวางสินค้าไปบนสายพานแล้วสายพานค่อย ๆ ลำเลียงไปผ่านตัวอ่านเพื่อคำนวณเงินที่ต้องชำระ) โดย override method enqueue() เพื่อรับสินค้ามาใส่ในคิวสินค้า (อาร์เรย์ลิสต์) ที่ต้องชำระเงิน แล้วแสดงผลลัพธ์ว่า ชื่อสินค้า is added in queue และ override method dequeue() เพื่อเอาสินค้าออกจากคิว (อาร์เรย์ลิสต์) ไปคิดเงินรวมที่ต้องชำระ



2. กำหนด inheritance hierarchy ดังนี้



<u>คำอธิบาย</u>

ภูมิประเทศ (Terrain) แบ่งออกเป็น บนดิน ท้องฟ้า และ ในน้ำ ในภูมิประเทศแต่ละชนิดจะบอกว่า สัตว์ชนิดใดสามารถเคลื่อนที่ในภูมิประเทศได้ (canMove() ซึ่ง return boolean)

สัตว์ แบ่งออกเป็น แมว,ปลา และนก โดยสัตว์แต่ละชนิดมีความสามารถแตกต่างกัน ดังต่อไปนี้ แมว สามารถเดินและวิ่ง (CanWalk), ปลา สามารถว่ายน้ำ (CanSwim), นก สามารถ เดิน วิ่ง และบิน (CanWalk, CanFly) โดยถ้ามีการระบุชนิดของภูมิประเทศมาให้จะบอกว่าสัตว์มีความสามารถนั้น ๆ หรือไม่ เช่น ถ้าบอกว่า แมวเดินบนดิน โปรแกรมจะตรวจสอบว่าในภูมิประเทศนั้น แมวเดินได้หรือไม่ ถ้าได้ แสดงข้อความว่าได้ ถ้าไม่ได้ แสดงข้อความว่าไม่ได้

ครูให้ไฟล์ที่เป็น interface คือ CanFly, CanWalk และ CanSwim ดังนี้

```
public interface CanFly {
    void fly(Terrain terrain);
}
```

```
public interface CanWalk {
    void walk(Terrain terrain);
    void run(Terrain terrain);
}
```

```
public interface CanSwim {
    void swim(Terrain terrain);
}
```

ไฟล์ที่เป็น class คือ Sky, Fish และ InterfaceProject ซึ่งคลาสทดสอบนี้ยังไม่สมบูรณ์ นิสิตจงเติมคำสั่งตามโจทย์ที่เขียนให้ในนั้น

```
public class Sky extends Terrain{
    public Sky(String name) {
        super(name);
    }
    @Override
    public boolean canMove(Animal animal) {
        if(animal instanceof CanFly)
            return true;
        return false;
    }
}
```

```
public class Fish extends Animal implements CanSwim {
    public Fish(String name) {
        super(name);
    }
    @Override
    public void swim(Terrain terrain) {
        if (terrain.canMove(this)) {
            System.out.println(this.getName() +"(" +this.getClass().getSimpleName()
                    +") can swim on " + terrain.getName() + "("+terrain.getClass().getSimpleName()+")");
        } else {
            System.out.println(this.getName() +"(" +this.getClass().getSimpleName()
                    +") can not swim on " + terrain.getName() +
                    "("+terrain.getClass().getSimpleName()+")");
        }
    }
}
```

```
import java.util.ArrayList;
public class InterfaceProject {
    public static void main(String[] args) {
        Ground ground1 = new Ground("Bangkok Ground");
        Pool pool1 = new Pool("Chula swimming pool");
        Sky sky1 = new Sky("Japan Sky");
// 1. ทดลองรันและดูผลลัพธ์
        System.out.println("**************1*************");
        Fish fish1 = new Fish("Nemo");
        fish1.swim(ground1);
        fish1.swim(pool1);
        fish1.swim(sky1);
        System.out.println("****************************");
```

```
// 2. สร้าง object จาก class Cat และใช้งาน method ใน CanWalk ดูตัวอย่างจาก Fish
//  เสร็จแล้วผลลัพธ์ต้องตรงกับผลรัน
        System.out.println("**************2*************");
//      เพิ่มโค้ดตรงนี้
        System.out.println("****************************");

// 3. สร้าง object จาก class Bird และใช้งาน method ใน CanWalk และ CanFly
//  เสร็จแล้วผลลัพธ์ต้องตรงกับผลรัน
        System.out.println("**************3*************");
//      เพิ่มโค้ดตรงนี้
        System.out.println("****************************");

// 4.   สร้าง ArrayList Terrain และ add ground1, pool1 และ sky1 ลงไปตามลำดับ
//      สร้าง ArrayList Animal และ add cat1, fish1 และ bird1 ลงไปตามลำดับ
//      เพิ่มโค้ดตรงนี้
        System.out.println("**************4*************");
//      Uncomment code ด้านล่าง ลองอ่าน code และทำความเข้าใจ ดูผลลัพธ์ที่เกิดขึ้น
//        for(Animal animal:animals) {
//            for(Terrain terrain:terrains) {
//                if (animal instanceof CanWalk) {
//                    ((CanWalk) animal).walk(terrain);
//                    ((CanWalk) animal).run(terrain);
//                }
//                if (animal instanceof CanSwim) {
//                    ((CanSwim) animal).swim(terrain);
//                }
//                if (animal instanceof CanFly) {
//                    ((CanFly) animal).fly(terrain);
//                }
```

```
//            }
//            System.out.println("****************************");
//        }
    }
}
```

เมื่อเติมโปรแกรมใน comment ข้อ 2-4 จนสมบูรณ์แล้วและ uncomment ในข้อ 4 ผลลัพธ์ของโปรแกรมคือ

```
**************1*************
Nemo(Fish) can not swim on Bangkok Ground(Ground)
Nemo(Fish) can swim on Chula swimming pool(Pool)
Nemo(Fish) can not swim on Japan Sky(Sky)
****************************
**************2*************
Garfield(Cat) can walk on Bangkok Ground(Ground)
Garfield(Cat) can not walk on Chula swimming pool(Pool)
Garfield(Cat) can not walk on Japan Sky(Sky)
Garfield(Cat) can run on Bangkok Ground(Ground)
Garfield(Cat) can not run on Chula swimming pool(Pool)
Garfield(Cat) can not run on Japan Sky(Sky)
****************************
**************3*************
Red Angry Bird(Bird) can walk on Bangkok Ground(Ground)
Red Angry Bird(Bird) can not walk on Chula swimming pool(Pool)
Red Angry Bird(Bird) can not walk on Japan Sky(Sky)
Red Angry Bird(Bird) can run on Bangkok Ground(Ground)
Red Angry Bird(Bird) can not run on Chula swimming pool(Pool)
Red Angry Bird(Bird) can not run on Japan Sky(Sky)
Red Angry Bird(Bird) can fly over Bangkok Ground(Ground)
Red Angry Bird(Bird) can fly over Chula swimming pool(Pool)
Red Angry Bird(Bird) can fly over Japan Sky(Sky)
```

```
**************************
**************4*************
Garfield(Cat) can walk on Bangkok Ground(Ground)
Garfield(Cat) can run on Bangkok Ground(Ground)
Garfield(Cat) can not walk on Chula swimming pool(Pool)
Garfield(Cat) can not run on Chula swimming pool(Pool)
Garfield(Cat) can not walk on Japan Sky(Sky)
Garfield(Cat) can not run on Japan Sky(Sky)
**************************
Nemo(Fish) can not swim on Bangkok Ground(Ground)
Nemo(Fish) can swim on Chula swimming pool(Pool)
Nemo(Fish) can not swim on Japan Sky(Sky)
**************************
Red Angry Bird(Bird) can walk on Bangkok Ground(Ground)
Red Angry Bird(Bird) can run on Bangkok Ground(Ground)
Red Angry Bird(Bird) can fly over Bangkok Ground(Ground)
Red Angry Bird(Bird) can not walk on Chula swimming pool(Pool)
Red Angry Bird(Bird) can not run on Chula swimming pool(Pool)
Red Angry Bird(Bird) can fly over Chula swimming pool(Pool)
Red Angry Bird(Bird) can not walk on Japan Sky(Sky)
Red Angry Bird(Bird) can not run on Japan Sky(Sky)
Red Angry Bird(Bird) can fly over Japan Sky(Sky)
**************************
```

จาก inheritance hierarchy จงสร้าง abstract superclass Animal และ Terrain และจงสร้าง class Ground, Pool, Bird, Cat ขอให้นิสิตดูคำสั่งในคลาสทดสอบว่าในแต่ละ class จะต้องสร้าง constructor อย่างไร