好的,我会按照文档中的实验内容步骤来给出相应的PlantUML代码和Java代码。

1. 首先是基本的Bridge设计模式类图:

```plaintext

@startuml

abstract class Bird {

# flyBehavior: FlyBehavior

+ {abstract} display(): void

+ performFly(): void

}

interface FlyBehavior {

+ fly(): void

}

class AbleFly {

+ fly(): void

}

Bird o-- FlyBehavior

AbleFly ..|> FlyBehavior

@enduml

```

对应的Java代码:

```java

public abstract class Bird {

protected FlyBehavior flyBehavior;

public abstract void display();

public void performFly() {

flyBehavior.fly();

}

}

public interface FlyBehavior {

void fly();

}

public class AbleFly implements FlyBehavior {

public void fly() {

System.out.println("I'm flying!");

}

}

```

2. 添加UnableFly类:

```plaintext

@startuml

abstract class Bird {

# flyBehavior: FlyBehavior

+ {abstract} display(): void

+ performFly(): void

}

interface FlyBehavior {

+ fly(): void

}

class AbleFly {

+ fly(): void

}

class UnableFly {

+ fly(): void

}

Bird o-- FlyBehavior

AbleFly ..|> FlyBehavior

UnableFly ..|> FlyBehavior

@enduml

```

Java代码:

```java

public class UnableFly implements FlyBehavior {

public void fly() {

System.out.println("扑腾几下飞不起来");

}

}

```

3. 添加SwimBehavior接口及其实现类:

```plaintext

@startuml

abstract class Bird {

# flyBehavior: FlyBehavior

# swimBehavior: SwimBehavior

+ {abstract} display(): void

+ performFly(): void

+ performSwim(): void

}

interface FlyBehavior {

+ fly(): void

}

interface SwimBehavior {

+ swim(): void

}

class AbleFly {

+ fly(): void

}

class UnableFly {

+ fly(): void

}

class AbleSwim {

+ swim(): void

}

class UnableSwim {

+ swim(): void

}

Bird o-- FlyBehavior

Bird o-- SwimBehavior

AbleFly ..|> FlyBehavior

UnableFly ..|> FlyBehavior

AbleSwim ..|> SwimBehavior

UnableSwim ..|> SwimBehavior

@enduml

```

Java代码:

```java

public interface SwimBehavior {

void swim();

}

public class AbleSwim implements SwimBehavior {

public void swim() {

System.out.println("鱼翔浅底");

}

}

public class UnableSwim implements SwimBehavior {

public void swim() {

System.out.println("扑腾几下就淹死了");

}

}

// 更新Bird类

public abstract class Bird {

protected FlyBehavior flyBehavior;

protected SwimBehavior swimBehavior;

public abstract void display();

public void performFly() {

flyBehavior.fly();

}

public void performSwim() {

swimBehavior.swim();

}

}

```

4. 添加具体的鸟类:

```plaintext

@startuml

abstract class Bird {

# flyBehavior: FlyBehavior

# swimBehavior: SwimBehavior

+ {abstract} display(): void

+ performFly(): void

+ performSwim(): void

}

interface FlyBehavior {

+ fly(): void

}

interface SwimBehavior {

+ swim(): void

}

class AbleFly {

+ fly(): void

}

class UnableFly {

+ fly(): void

}

class AbleSwim {

+ swim(): void

}

class UnableSwim {

+ swim(): void

}

class Eagle {

+ display(): void

}

class Penguin {

+ display(): void

}

Bird o-- FlyBehavior

Bird o-- SwimBehavior

AbleFly ..|> FlyBehavior

UnableFly ..|> FlyBehavior

AbleSwim ..|> SwimBehavior

UnableSwim ..|> SwimBehavior

Eagle --|> Bird

Penguin --|> Bird

@enduml

```

Java代码:

```java

public class Eagle extends Bird {

public Eagle() {

flyBehavior = new AbleFly();

swimBehavior = new UnableSwim();

}

public void display() {

System.out.println("I'm an eagle");

}

}

public class Penguin extends Bird {

public Penguin() {

flyBehavior = new UnableFly();

swimBehavior = new AbleSwim();

}

public void display() {

System.out.println("I'm a penguin");

}

}

```

5. 添加Client类进行测试:

```plaintext

@startuml

class Client {

+ main(args: String[]): void

}

Client ..> Eagle

Client ..> Penguin

@enduml

```

Java代码:

```java

public class Client {

public static void main(String[] args) {

Bird eagle = new Eagle();

Bird penguin = new Penguin();

eagle.display();

eagle.performFly();

eagle.performSwim();

penguin.display();

penguin.performFly();

penguin.performSwim();

}

}

```

这就是按照实验内容步骤给出的PlantUML类图和对应的Java代码实现。