Schnittstellen
und
Abstrakte Klassen

Beispiel

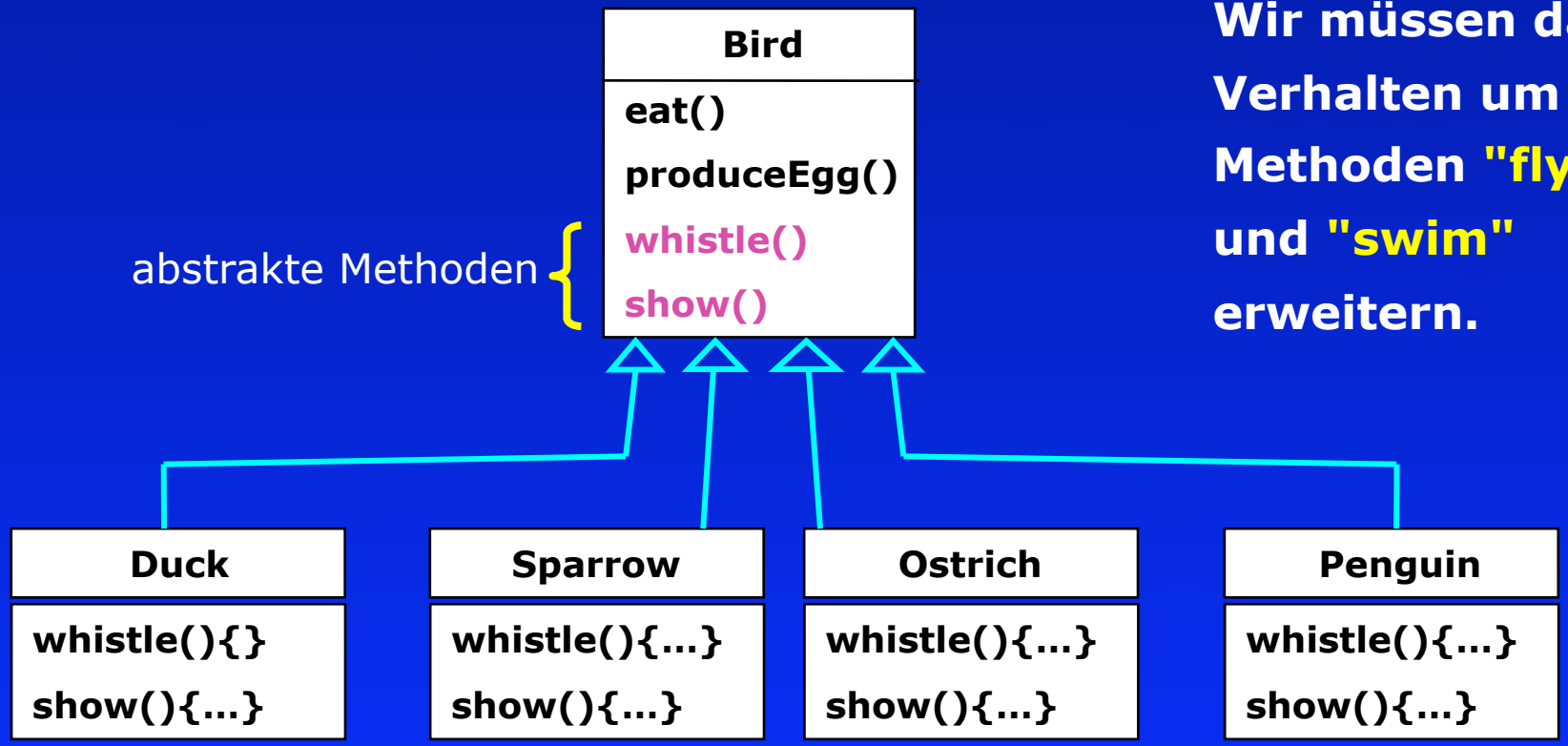
Prof. Dr. Margarita Esponda-Argüero

Abstrakte Klassen und Interfaces als Strukturierungsmittel

Beispiel:

Nehmen wir an:

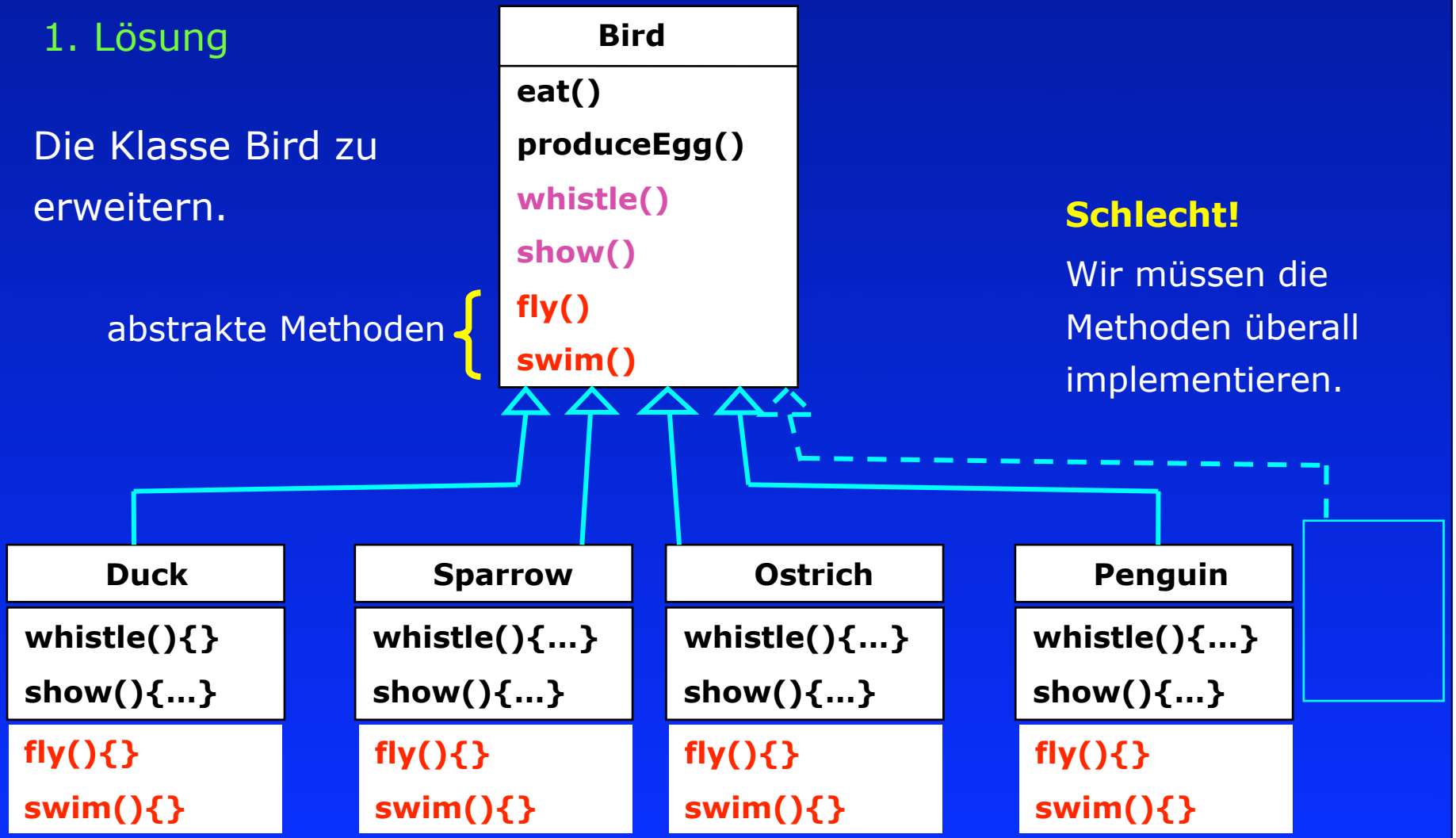
Wir müssen das Verhalten um die Methoden "fly" und "swim" erweitern.



Abstrakte Klassen und Interfaces als Strukturierungsmittel

1. Lösung

Die Klasse Bird zu erweitern.



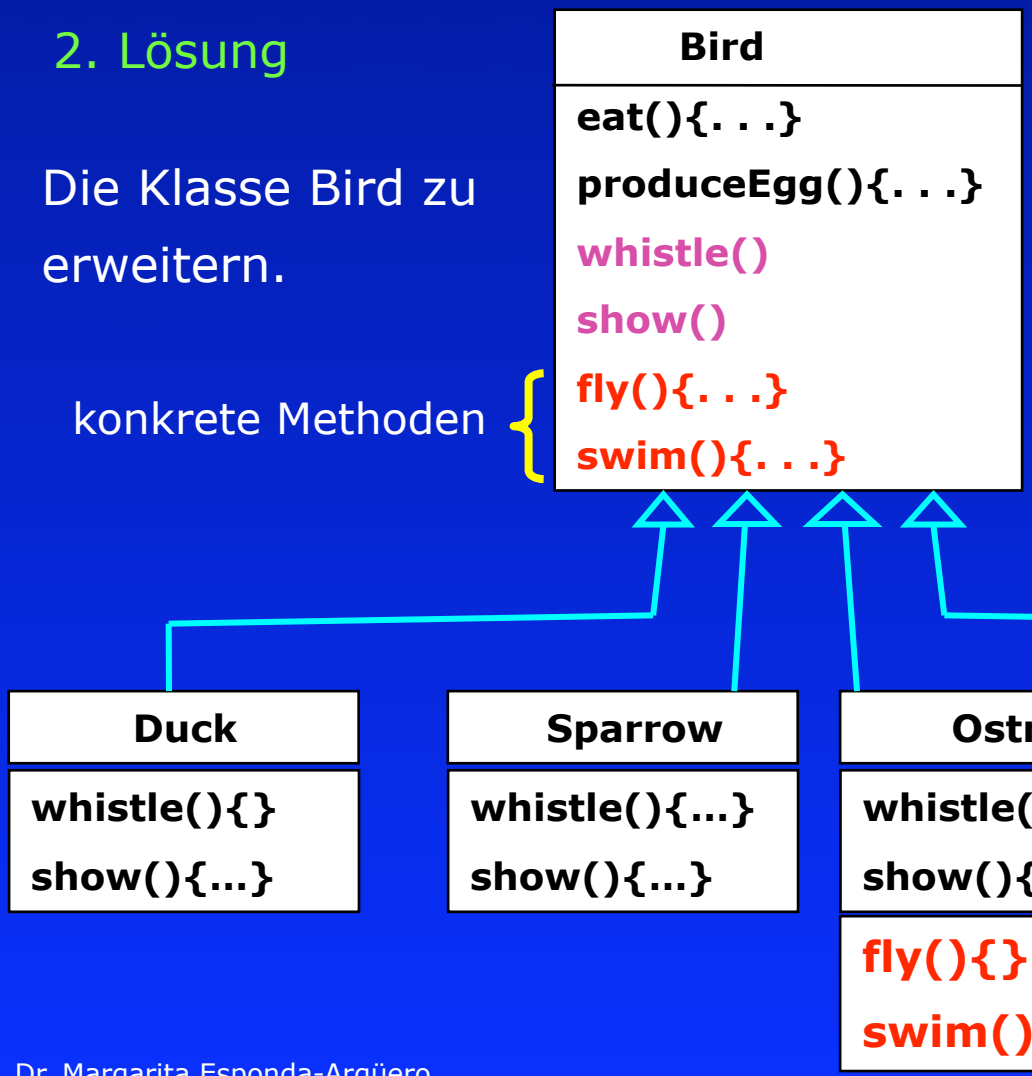
Schlecht!

Wir müssen die Methoden überall implementieren.

Abstrakte Klassen und Interfaces als Strukturierungsmittel

2. Lösung

Die Klasse Bird zu erweitern.



Probleme:

Pinguine und **Strauße** können nicht fliegen.

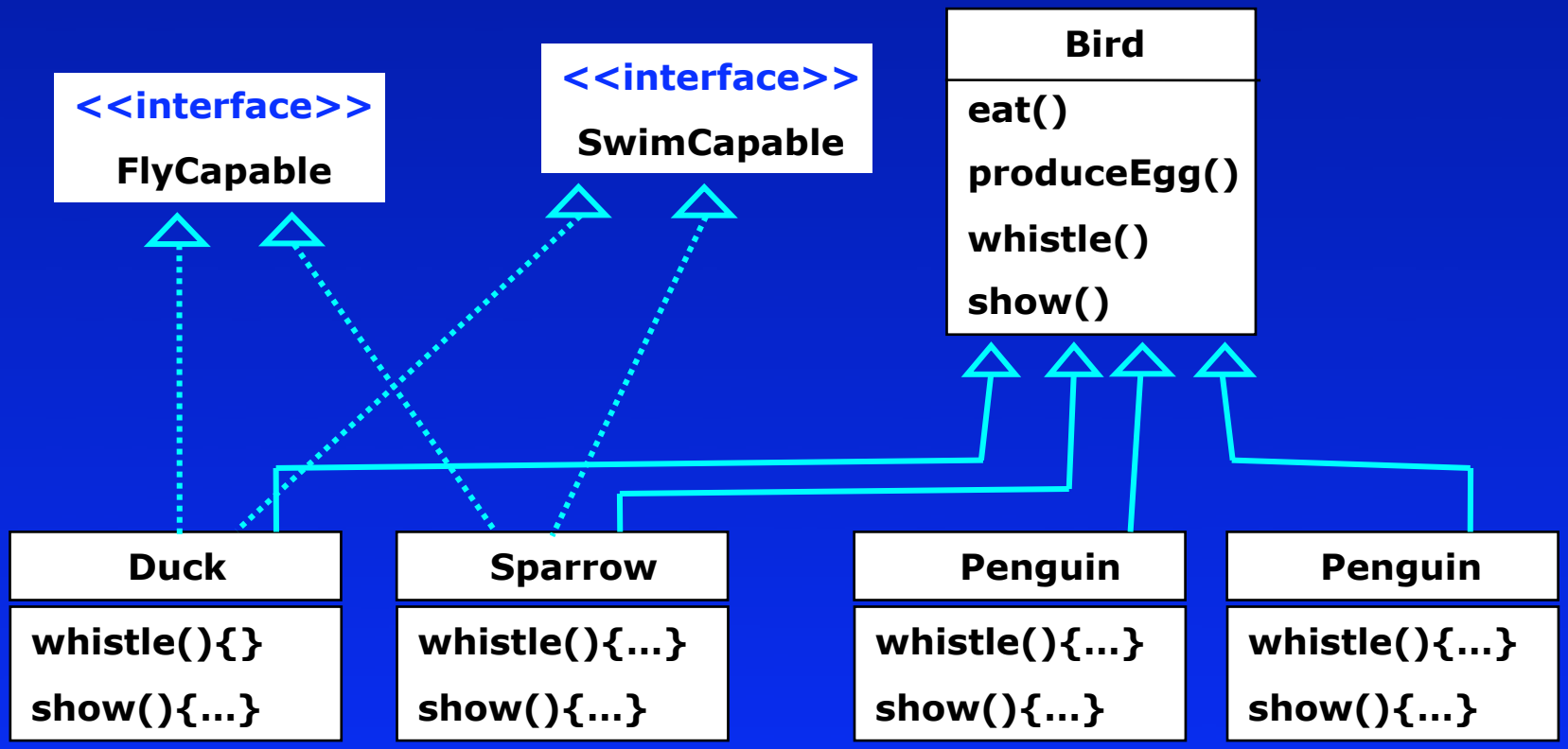
Strauße können nicht schwimmen.

Schlecht!

Abstrakte Klassen und Interfaces als Strukturierungsmittel

3. Lösung

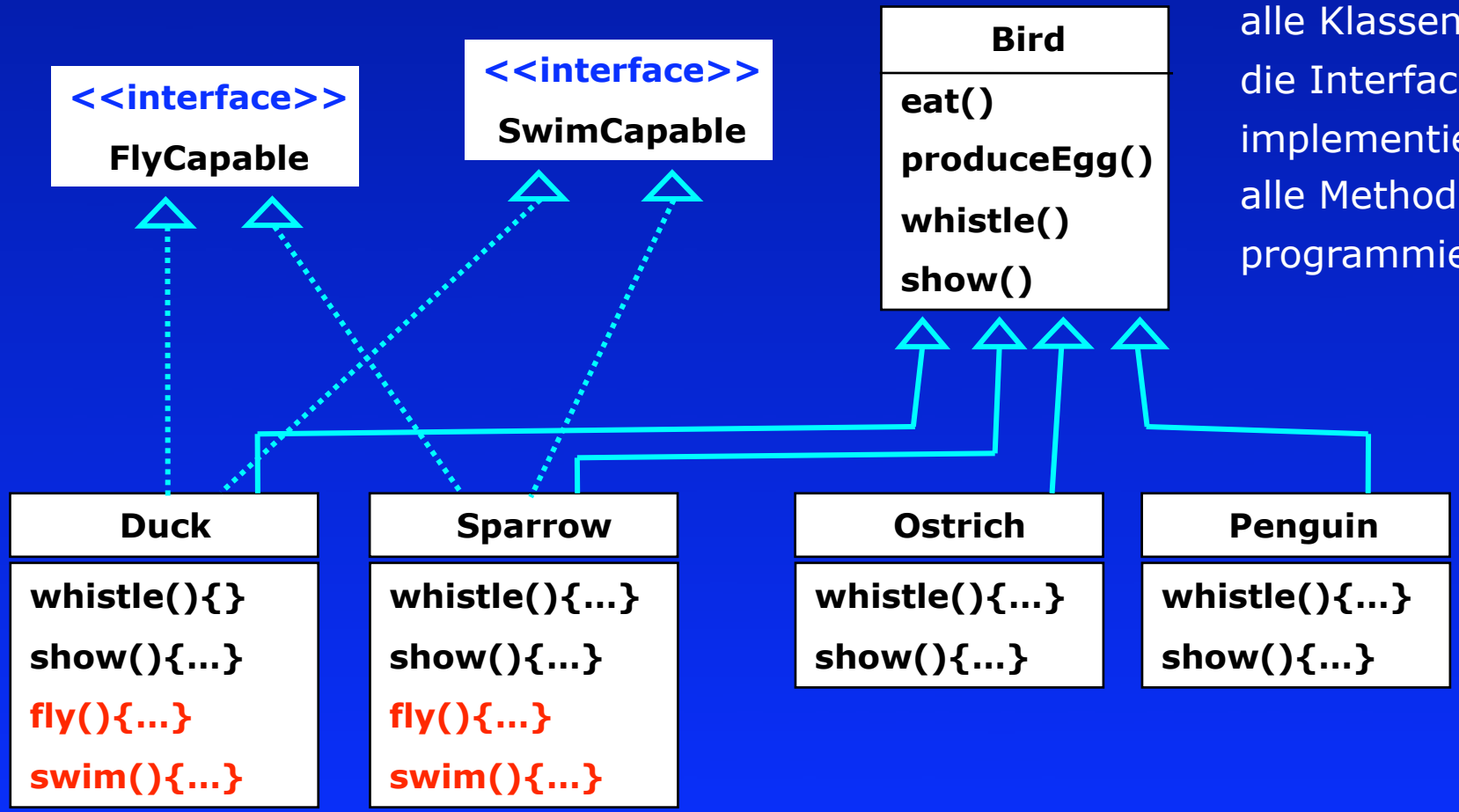
Wir definieren
zwei interfaces



Abstrakte Klassen und Interfaces als Strukturierungsmittel

3. Lösung

Wir definieren zwei Interfaces



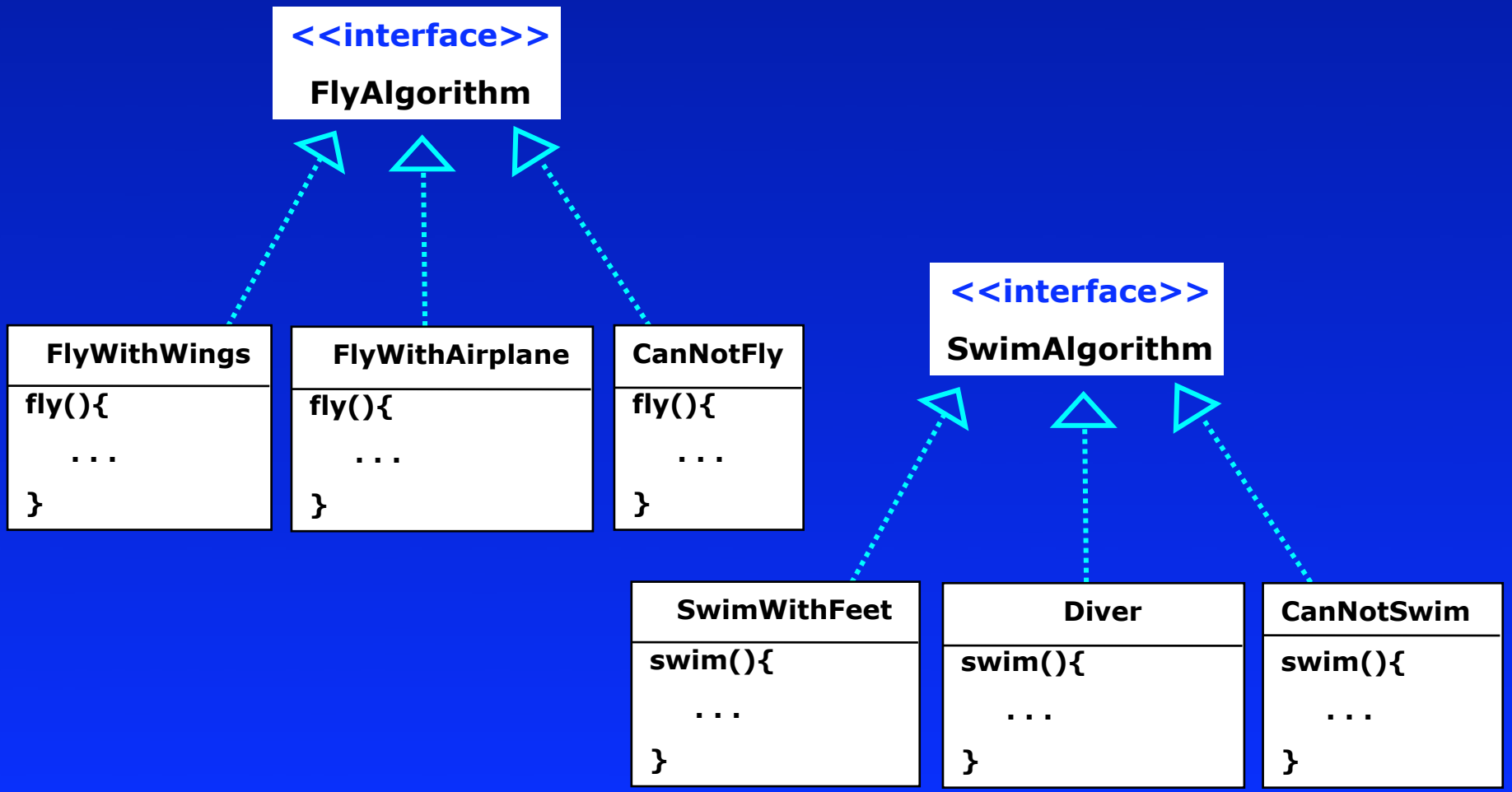
Probleme:

Wir müssen in alle Klassen, die die Interfaces implementieren alle Methoden programmieren.

Abstrakte Klassen und Interfaces als Strukturierungsmittel

4. Lösung - Wir trennen alles was sich in dem Verhalten ständig ändert.

- Wir definieren zwei Interfaces mit verschiedene Implementierungsklassen

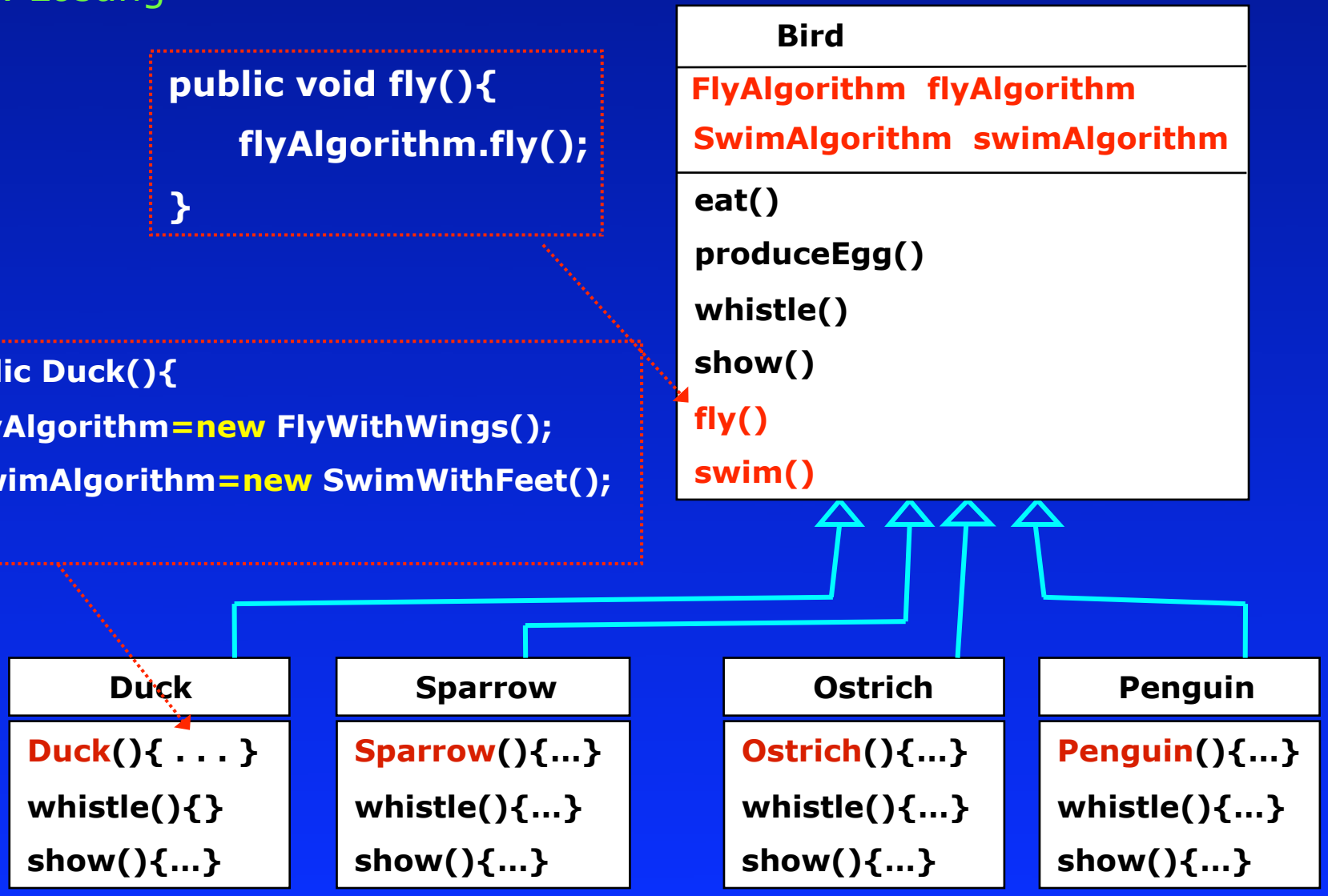


Abstrakte Klassen und Interfaces als Strukturierungsmittel

4. Lösung

```
public void fly(){
    flyAlgorithm.fly();
}
```

```
public Duck(){
  flyAlgorithm=new FlyWithWings();
  swimAlgorithm=new SwimWithFeet();
}
```

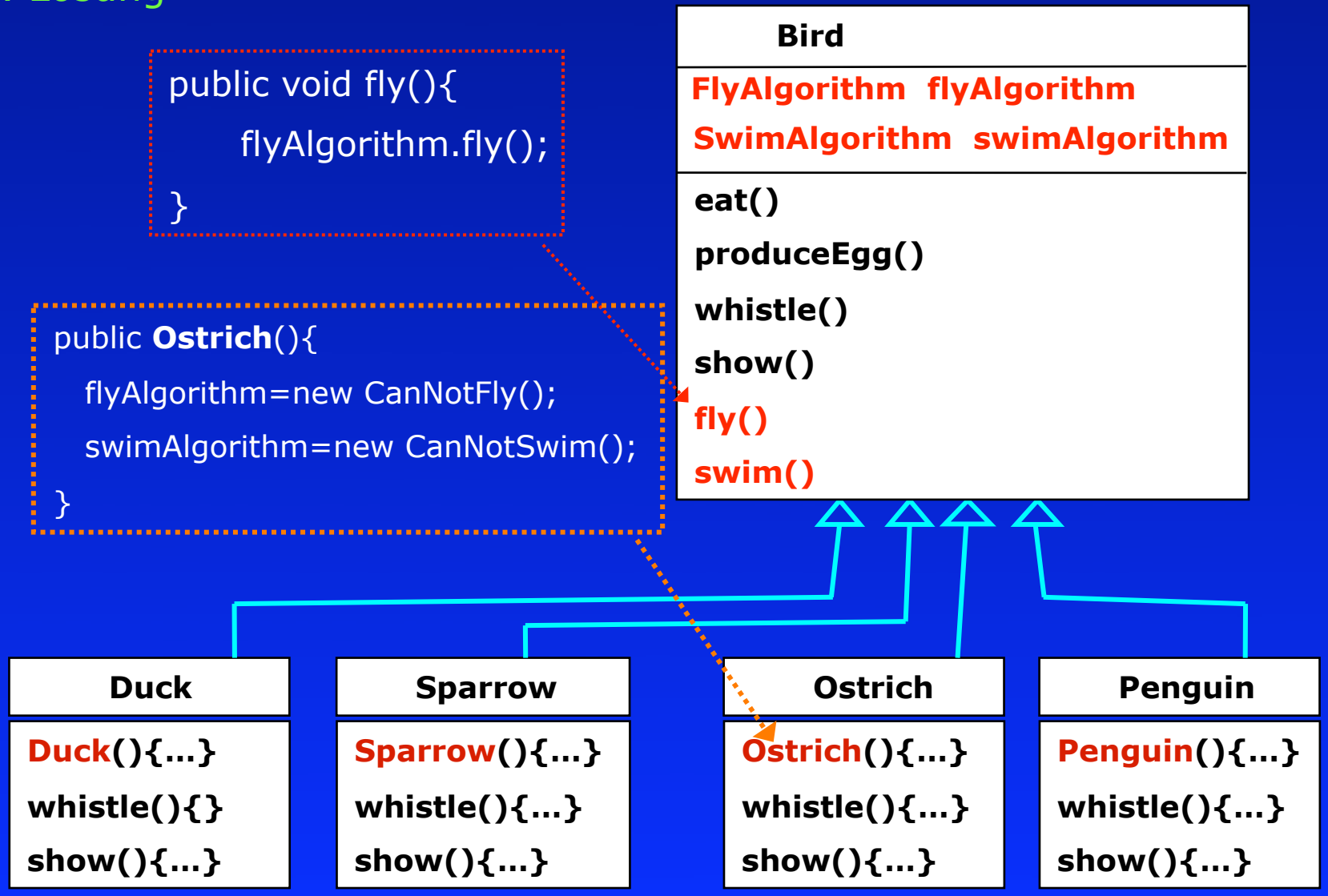


Abstrakte Klassen und Interfaces als Strukturierungsmittel

4. Lösung

```
public void fly(){
      flyAlgorithm.fly();
}
```

```
public Ostrich(){
   flyAlgorithm=new CanNotFly();
   swimAlgorithm=new CanNotSwim();
}
```

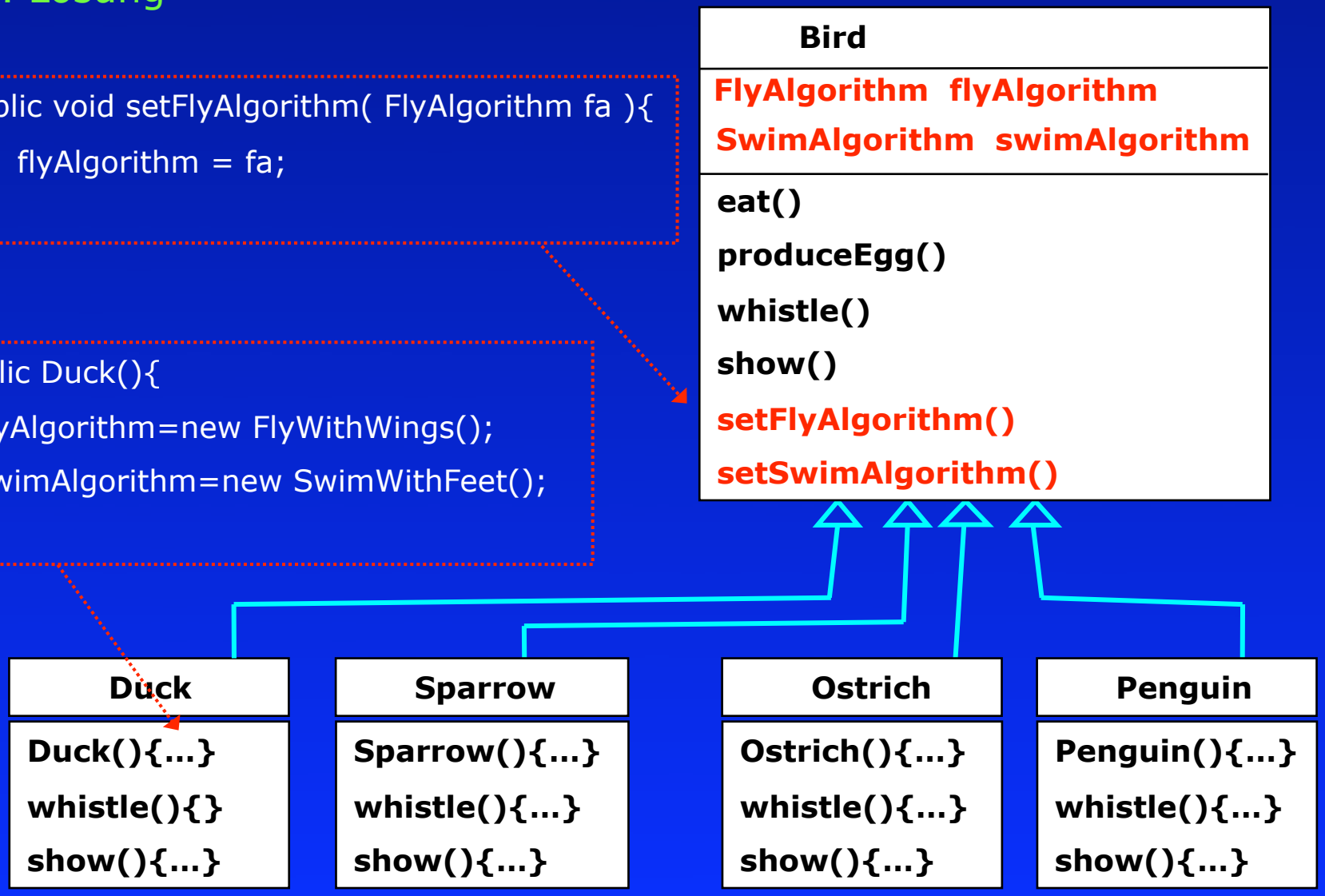


Abstrakte Klassen und Interfaces als Strukturierungsmittel

5. Lösung

```
public void setFlyAlgorithm( FlyAlgorithm fa ){
    flyAlgorithm = fa;
}
```

```
public Duck(){
  flyAlgorithm=new FlyWithWings();
  swimAlgorithm=new SwimWithFeet();
}
```



5. Lösung

Strategie-Pattern