PROCEDURALE PROGRAMMING

Dozent: Dr. Andreas Jäger

Was wir bisher gelernt haben…

…ist prozedurale Programmierung

so was zeichnet prozedurale Programmierung jetzt aus.

Prozedurale Programmierung

- Basiert auf dem Aufruf von Prozeduren. (Funktionen, Routinen,...)
- Jede Prozedur kann zu jedem Programmzeitpunkt aufgerufen werden.
- Funktionale Programmierung (Paradigma) sind das Selbe.
- beinhaltet Strukturierte Programmierung
  - > Elemente zum Kontroll-fluss (if, while,...)
  - > Blöcke
  - > Strukturen

OOP

Dozent: Dr. Andreas Jäger

Object orientierte Programmierung

wie der Name impliziert:

Basiert auf dem Konzept von „Objekten“

Was könnte das bedeuten?

Objekte können:

- Daten.
- Prozeduren.
- Gültigkeitsbereiche.

sein.

Objekte als Daten

Wenn Objekte zum darstellen von Daten genutzt werden:

Daten werden als “Attribute” bezeichnet.

Objekte als Prozeduren

Wenn Objekte zum darstellen von Prozeduren genutzt werden:

Prozeduren werden als “Methoden” bezeichnet.

C++ unterstützt OOP

C++ unterstützt die Objekte orientierte Programmierung.

Im folgenden werden wir uns die Sprach Konstrukte anschauen die uns C++

zur Verfügung stellt.

C++ Klassen

```
class Bird {
 public:
  ::std::string gender;
  bool fly();
  bool walk();
  bool makeSound() { ::std::cout << this->_sound<< std::endl; }
  Bird() {}
  ~Bird() {}
 private:
  state _currentState;
  const ::std::string _sound = std::string("tchirp");};
```

New keywords

class

Schlüsselwort zum erstellen einer Klasse

public

Jeder kennt und hat Zugriff

protected

Nur die Klasse und abgeleitete Klassen haben Zugriff

private

Nur die Klasse hat Zugriff

New keywords

Konstruktor [Bird()]

Methode mit dem selben Bezeichner wie die Klasse

Existiert immer (Default-Konstruktor)

Dient zum initialisieren der Klasse (Attribute)

Kann überladen werden

this

Zeiger auf sich selbst

New keywords

Destruktor [~Bird()]

Methode mit dem selben bezeichner wie die Klasse

Existiert immer (Default-Destruktor)

Dient zum aufräumen der Klasse

Anmerkungen

Initialisierungslisten:

Erlauben das direkte initialisieren von Attributen.

Die Liste muss die selbe Reihenfolge haben wie die Deklaration der Attribute

```
class Bird
{
 public:
  int _currentState;
  ::std::string gender;

 public:
 Bird() : _currentState(3)
};
```

Übung

1. Erstellen Sie eine Klasse Bruch.

Über das Interface der Klasse muss man auf Zähler und Nenner zugreifen können

2. Generieren Sie einen Bruch in ihrer main Funktion

3. Separieren Sie Ihre Klassedeklaration und Definition
   a) Ihre Deklaration soll in eine eigene Headerdatei
   b) Ihre Definition soll in eine eigene Sourcedatei

Lösung - Header

```
// Deklaration

class Bruch {

public:

Bruch(int z, int n);


private:

int zaehler_;

int nenner_;

};
```

Lösung - Cpp

```
// Definition
Bruch::Bruch(int z, int n) :
zaehler_(z),              // Initialisierung von zaehler_
nenner_(n)                // Initialisierung von nenner_
{}
```

Defaultparameter

Sie können einem Konstruktor Defaultparameter vorgeben:

```
Bruch(int z, int n = 1); // Deklaration mit Defaultwert

/* Funktioniert genau wie bei Funktionen */
```

Mehrere Konstruktoren

Auch das Überladen des Konstruktors funktioniert wie das Überladen einer Funktion

```
Bruch::Bruch(int z) :          // Definition
zaehler_(z),
nenner_(1)
{}

Bruch::Bruch(int z, int n) : // Definition
zaehler_(z),
nenner_(n)
{}
```

Kopierkonstruktor

Spezieller Konstruktor.

Erstellt ein Objekt anhand eines bereits vorhandenen Objektes.

Der Parameter des Kopierkonstruktors ist immer eine Referenz auf ein konstantes Objekt derselben Klasse

OOP ist mehr als…

…als die Welt in Objekten abzubilden.

Was sind die beiden wichtigsten Konzepte von OOP?

OOP

Inheritance

POLYMORPHISM

INHERITANCE

Dozent: Dr. Andreas Jäger

Inheritance

- Erlaubt das erstellen von Klassen basierend auf bestehenden Klassen
- Dies erlaubt die Verwendung aller Methoden und Attribute der Klasse von der geerbt wird.

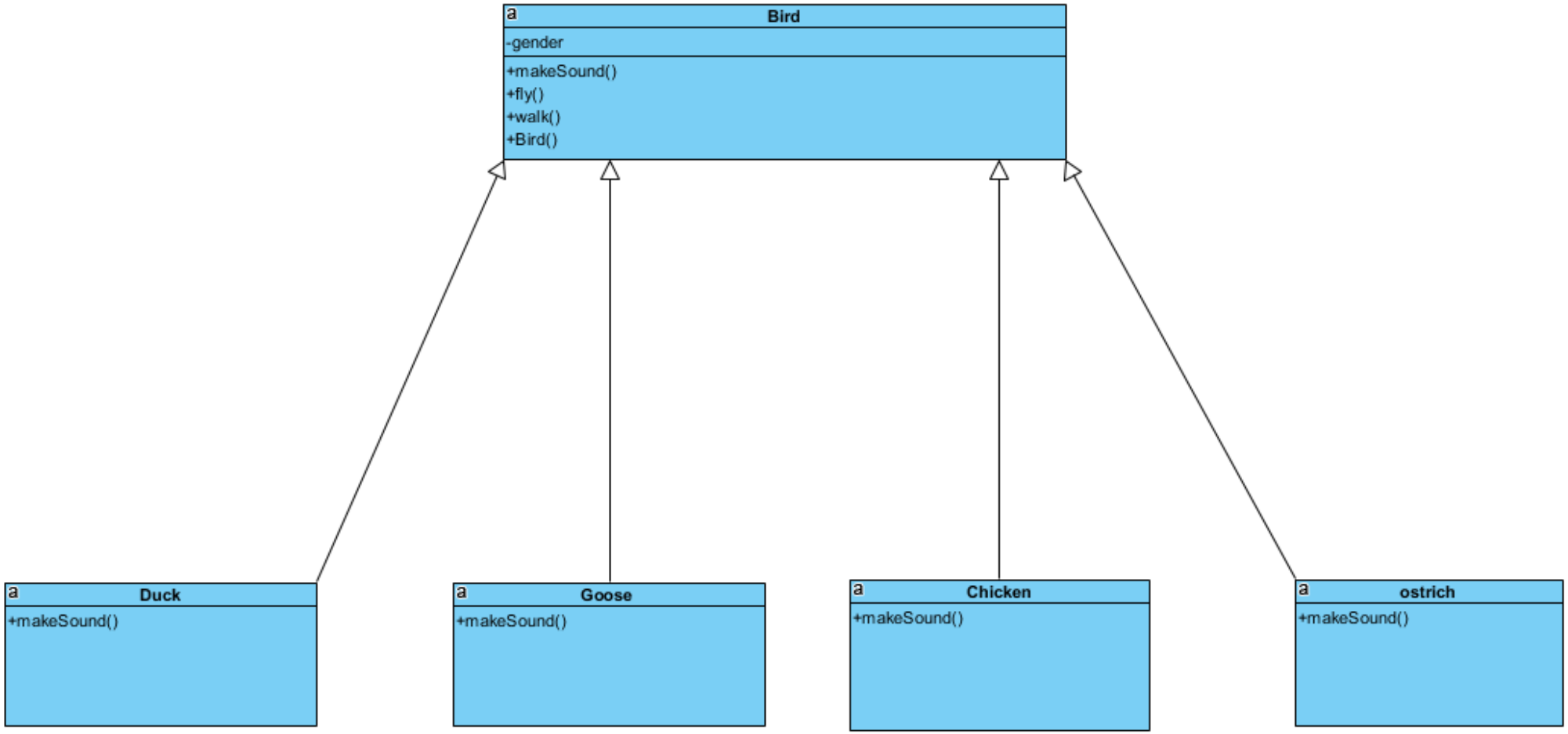
Wir unterscheiden hierbei:

Die Basisklasse

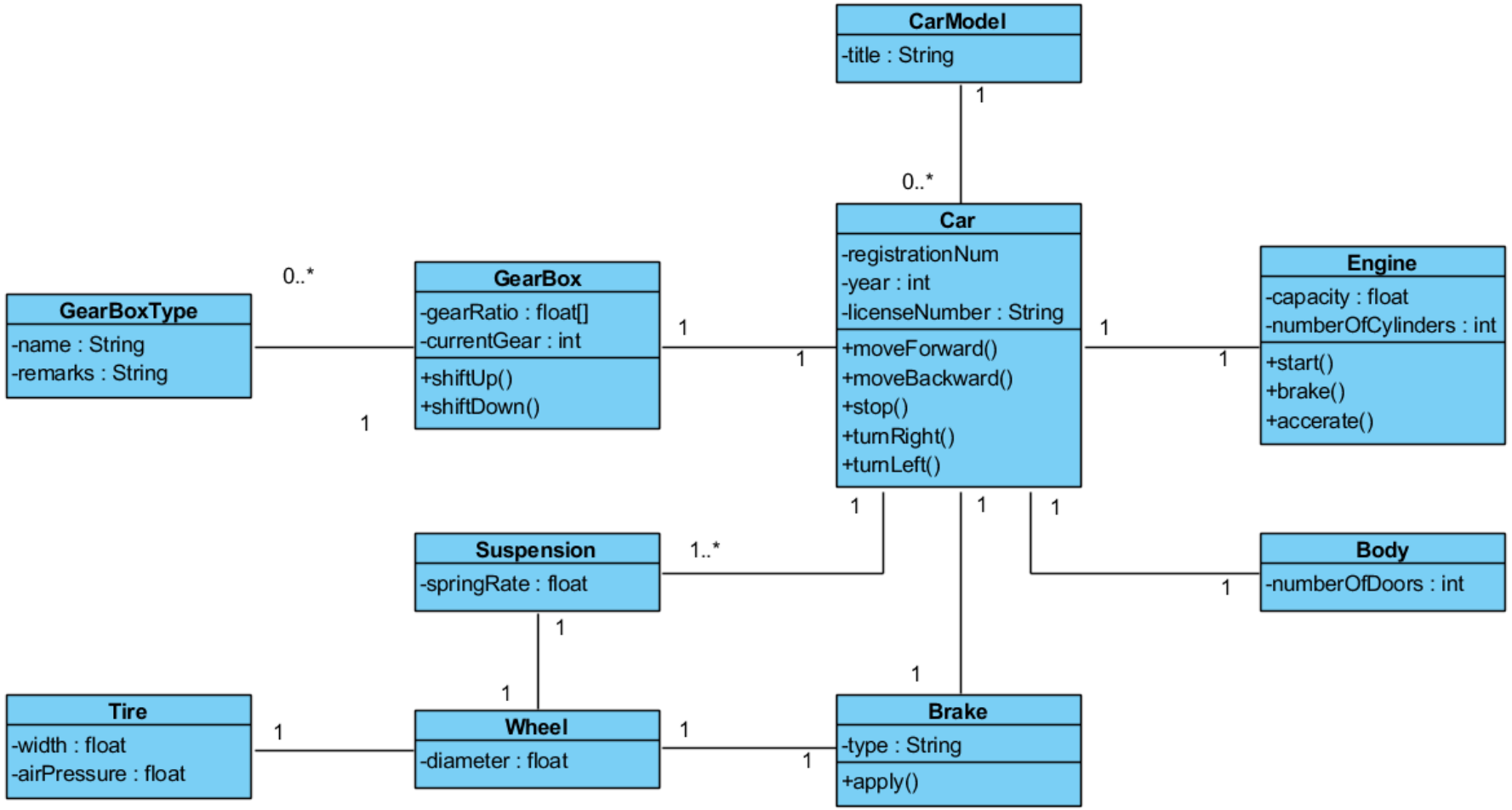
Die abgeleitet Klasse

Inheritance Example

What are the problems here?



Inheritance and Composition



C++ inheritance

```
class Duck: public Bird
{
 public:
  Duck(){}
};
```

Anmerkungen

Initialisierungslisten:

Können mehr!

Was ist wenn ein Konstruktor Parameter braucht

```
class Duck : public Bird
{
 public:
  ::std::string gender;

 public:
 Duck() : Bird(True)
};
```

POLYMORPHISM

Dozent: Dr. Andreas Jäger

Polymorphismus

Polymorphismus ist eines der wichtigsten Konzepte der OOP

Der Kern des Polymorphismus ist das Bereitstellen von einem Interface für unterschiedliche (Objekt)Typen.

Ein Interface → unterschiedliche Objekte

Polymorphismus

Anmerkung:

Polymorphie baut dabei auf Vererbung auf, ist aber nicht exklusiv für Vererbung!

Polymorphismus ist nicht beschränkt auf Klassen, sondern kann auch im Kontext von Operatoren vorkommen.

Polymorphismus

Polymorphie baut dabei auf dem Konzept der Vererbung auf.

Es macht sich zudem das Schnittstellenverhalten von Klassen zunutze.

Eine Methode (somit auch Operator) ist polymorph:
Sie hat in verschiedenen Klassen die gleiche Signatur
ABER unterschiedliche Implementierungen.

Signatur:
Name + Anzahl, Typ und Reihenfolge einer Funktion

Polymorphismus

Wir führen jetzt ein neues Schlüsselwort ein.

virtual

die Einsprung-Adresse wird erst zur Laufzeit ermittelt – dynamische Bindung.

virtual <Signatur> = 0; → pure virtual

explizit abstrakt deklariert. D.h:

Die abgeleiteten Klassen müssen eine Implementierung garantieren.

Polymorphismus

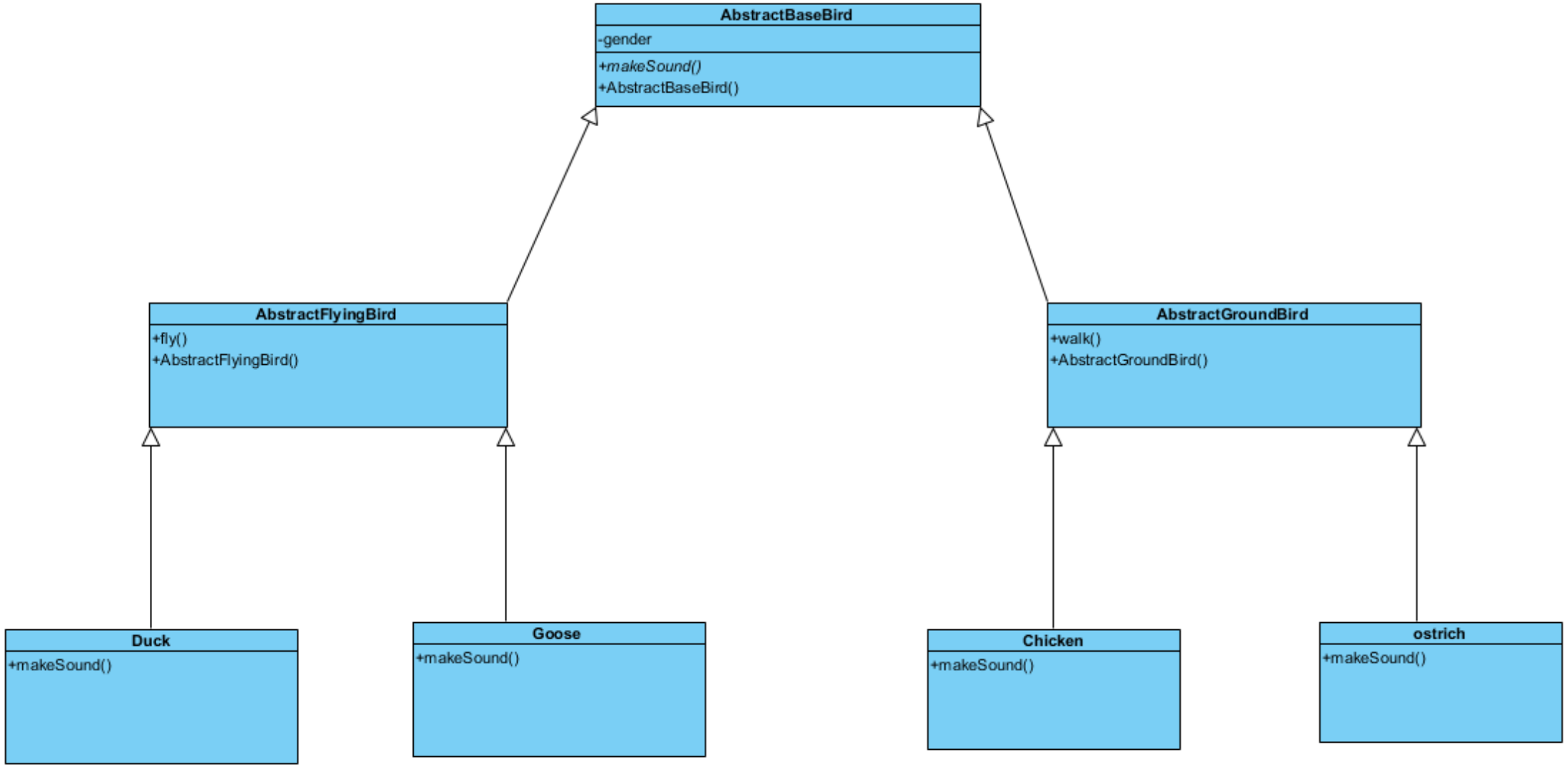
Eine Basisklasse, die virtuelle Methoden deklariert, kann somit als abstrakte Basisklasse aufgefasst werden.

Was heißt das?

Wir erzwingen von Benutzern unserer abstrakten Klasse, eine konkrete Implementierung zur Verfügung zu stellen!

(Ansonsten würde eine neue abstrakte Basisklasse erzeugt)

Polymorphism Example



C++ inheritance and polymorphism

wir können zusammenfassen:

Polymorphismus in Kontext von Vererbung baut auf der Vererbung auf.

Es erweitert dieses Konzept, um zur Laufzeit dynamisch die Korrekte Implementierung einer Schnittstelle zu ermitteln.

Example Code C++

```
#include <iostream>
class OutputGeneraterBase {
 public:
  virtual void show() { cout << "Hello base class"; }
};

class OutputGenerater : public OutputGeneraterBase {
 public:
  void show() { cout << "Hello derive class"; }
};
```

Example Code C++

```
void main() {
  OutputGeneraterBase _base;
  OutputGenerater     _derived;
  OutputGeneraterBase *pbase;
  pbase = &_base;
  pbase->show();  // call base class function

  pbase = &_derived;
  pbase->show();  // call derive class function
}
```