Abgeleitete Klassen

- ***abgeleitete Klasse*** **erweitert** ***Basisklasse*** **um eigene Komponenten (Datenmember, Methoden)**
- **Vererbung erspart Duplizierung der Strukturbeschreibung sowie der Methodenimplementierung**

```
class Konto { const long int Ktonr;
              char*    KtoInhaber;
              double   HabenZins, KtoStand;
  public:     Konto (long int Nr, char* Inh, double HZ, double = 0.0);
              void Info () const;
              void Einzahlung (double); // weitere Funktionen ...
};

class GiroKto: public Konto { double KreditLimit, SollZins;
  public:     GiroKto (long int, char*, double, double, double, double);
              void GiroInfo () const; // weitere Funktionen ...
  }
class Sparbuch: public Konto { date Faelligkeit; // date sei definiert
  public:     Sparbuch (long int, char*, double, double, date);
              void SparInfo () const; // ....
  }
```

- **Konstruktoren, Destruktoren und operator=(…) der Basisklassen werden nicht vererbt**

- jedoch Aufbau eines Objekts beginnend bei Basisklasse durch impliziten Aufruf des Default-Konstruktors (sofern vorhanden)

- oder expliziter Aufruf von Basisklassen-Konstruktoren:

```
Sparbuch::Sparbuch(long int Nr, char *Inh, double HZ, double KS, date
d):Konto(Nr, Inh, HZ, KS), Faelligkeit(d){}
```

- **Gleiche Komponenten-/Funktionsbezeichnungen in abgeleiter Klasse führen zum Überschreiben (override), Redefinieren**

- Methoden werden auch bei abweichender Parameterliste überschrieben

- Zugriff auf überschriebene Namen der Basisklassen über Geltungsbereichsoperator :: (explizite Klassenangabe)

```
class X {
  public:    int i;
             void f ();
}
class Y: public X {
  public:    int i;
             void f (char);
  }
```

```
Y y;

++y.i;
++y.X::i;

y.f("z");
y.f();          // Fehler!
y.X::f();       // o.k.
```