```
1  // Rennwagen.h (Aufgabe 1)
2
3  class Rennwagen {
4  private:
5      int kmh;
6  public:
7      Rennwagen() { kmh = 0; }
8      void beschleunigen(int K) { kmh += K; }
9      void bremsen() {
10         kmh -= 50;
11         if (kmh < 0) kmh = 0;
12     }
13     int Tempo() { return kmh; }
14     void coutTempo() {
15         if (kmh>0) cout << "Die Geschwindigkeit betraegt jetzt " << kmh << " km/h\n";
16         else cout << "Der Rennwagen steht!\n";
17     }
18 };
```

```cpp
// Rennwagen.cpp (Aufgabe 1)
//
#include "stdafx.h"         // Die gewohnten includes...
#include <iostream>
#include <string>
using namespace std;        // namespace vorab

#include "Rennwagen.h"      // Klassenvereinbarung einbinden

int main()
{
    Rennwagen RedBull;      // Objekt statisch erzeugen
    Rennwagen* McLaren = new Rennwagen;     // Objekt dynamisch erzeugen;
    Rennwagen* Formel1[2];  // Array mit Zeigern auf Rennwagen erzeugen
    Formel1[0] = new Rennwagen; // 1. Arrayelement anlegen
    Formel1[1] = new Rennwagen; // 2. Arrayelement anlegen
    Rennwagen DTM[2] = {
        Rennwagen(),
        Rennwagen()
    };

    cout << endl; // Zugriff auf statisch erzeugte Objekte:
    RedBull.beschleunigen(200);
    RedBull.bremsen();
    RedBull.bremsen();
    RedBull.bremsen();
    RedBull.coutTempo();

    cout << endl; // Zugriff auf dynamisch erzeugte Objekte:
    McLaren->beschleunigen(230);
    McLaren->bremsen();
    McLaren->bremsen();
    McLaren->bremsen();
    cout << "McLaren hat jetzt eine Geschwindigkeit von " << McLaren->Tempo() << "
km/h\n";
    McLaren->beschleunigen(30);
    cout << "McLaren hat jetzt eine Geschwindigkeit von " << McLaren->Tempo() << "
km/h\n";

    cout << endl;  // Zugriff auf dynamisch erzeugte Array-Objekte:
    Formel1[0]->beschleunigen(180);
    Formel1[0]->bremsen();
    Formel1[0]->bremsen();
    Formel1[0]->bremsen();
    Formel1[0]->bremsen();
    cout << "Formel1-0 hat jetzt eine Geschwindigkeit von " << Formel1[0]->Tempo() <<
 " km/h\n";
    Formel1[0]->coutTempo();

    cout << endl;
    Formel1[1]->beschleunigen(180);
    Formel1[1]->bremsen();
    Formel1[1]->coutTempo();
    Formel1[1]->beschleunigen(60);
    Formel1[1]->beschleunigen(42);
    Formel1[1]->coutTempo();
```

```
54      Formel1[1]->bremsen();
55      Formel1[1]->bremsen();
56      Formel1[1]->bremsen();
57      Formel1[1]->coutTempo();
58
59      delete Formel1[0];
60      delete Formel1[1];
61
62      cout << endl; // Zugriff auf statisch erzeugte Array-Objekte
63      DTM[0].beschleunigen(220);
64      DTM[0].coutTempo();
65
66      // delete DTM[0]; geht nicht
67
68      system("pause");
69      return 0;
70  }
71
```

```
1  // Rennwagen2.h (Aufgabe 2)
2
3  const int MaxSpeed=380; // vorgezogen aus Aufgabe 4...
4
5  class Rennwagen {
6  private:
7      int kmh;
8      // die beiden neuen Eigenschaften:
9      string Name;
10     bool MotorLaeuft;
11 public:
12     // beim alten Konstruktor muss jetzt der Name nachgefragt werden
13     Rennwagen() {
14         cout << "Wie heisst der Rennwagen? ";
15         cin >> Name;
16         kmh = 0;
17         MotorLaeuft = false;
18     }
19     // der neue Konstruktor bekommt den Namen übergeben
20     Rennwagen(string n) :Name(n) { kmh = 0; MotorLaeuft = false; }
21
22     // beschleunigen jetzt schon mit Maximalwert-Berücksichtigung
23     void beschleunigen(int K) {
24         char Eingabe = 'j';
25         if (!MotorLaeuft) {
26             cout << "Der Motor des " << Name << " ist noch aus!\n";
27             cout << "Soll der Motor jetzt angelassen werden (j/n)? ";
28             cin >> Eingabe;
29         }
30         // Eingabe ist per Initialisierung ODER Benutzereingabe j - oder auch nicht:
31         // Diese Programmiermethode ist zwar sehr effizient,
32         // vermischt aber eigentlich zwei funktionell getrennte Vorgänge (Nachfrage
   und anlassen / beschleunigen)
33         if (Eingabe == 'j') {
34             MotorLaeuft = true;
35             kmh += K;
36             if (kmh > MaxSpeed) kmh = MaxSpeed;
37         }
38     }
39
40     // bremsen wie bisher:
41     void bremsen() {
42         kmh -= 50;
43         if (kmh < 0) kmh = 0;
44     }
45
46     // Tempowert ausgeben wie bisher:
47     int Tempo() { return kmh; }
48
49     // Tempowert in Satz mit Name ausgeben:
50     void coutTempo() {
51         if (kmh>0) cout << "Der " << Name << " faehrt jetzt " << kmh << " km/h\n";
52         else cout << "Der " << Name << " steht!\n";
53     }
54
55     // Neue Methoden, um den Motor zu starten / zu stoppen:
```

```
56      void MotorStarten() {
57          if (MotorLaeuft)
58              cout << "Der Motor des " << Name << " laeuft bereits...\n";
59          else {
60              MotorLaeuft = true;
61              cout << "Der Motor des " << Name << " wurde angelassen.\n";
62          }
63      }
64      void MotorStoppen() {
65          if (kmh == 0) {
66              MotorLaeuft = false;
67              cout << "Der Motor des " << Name << " ist jetzt aus!\n";
68          }
69          else cout << "Der " << Name << " faehrt noch! Sie sollten den Motor jetzt
            nicht ausschalten!\n";
70      }
71  };
```

```cpp
// Rennwagen2.cpp (Aufgabe 2)
//
#include "stdafx.h"
#include <iostream>
#include <string>
using namespace std;

#include "Rennwagen2.h"

int main()
{
    Rennwagen RedBull("Red Bull");                  // Objekt statisch erzeugen
    Rennwagen* McLaren = new Rennwagen("McLaren");  // Objekt dynamisch erzeugen;
    Rennwagen* Formel1[2];              // Array mit Zeigern auf Rennwagen erzeugen
    Formel1[0] = new Rennwagen;         // 1. Arrayelement anlegen
    Formel1[1] = new Rennwagen("Ferrari");  // 2. Arrayelement anlegen mit neuem Konstruktor

    Rennwagen DTM[2] = {                // und auch ein statisches Array anlegen:
        Rennwagen(),
        Rennwagen("Mercedes")
    };

    cout << endl;
    RedBull.beschleunigen(200);
    RedBull.bremsen();
    RedBull.bremsen();
    RedBull.bremsen();
    RedBull.coutTempo();

    cout << endl;
    McLaren->beschleunigen(230);
    McLaren->bremsen();
    McLaren->bremsen();
    McLaren->bremsen();
    // cout << "McLaren hat jetzt eine Geschwindigkeit von " << McLaren->tempo() << " km/h\n";
    // Hier den Namen direkt hardcoded zu programmieren, ist nicht mehr sinnvoll,
    // denn der Rennwagen kann ja ganz anders heißen. string Name ist aber private...!
    // Deshalb Name public machen oder Methode dazu schreiben oder coutTempo() verwenden.
    // In der Realität müsste man das jetzt klären, hier verwende ich den kürzesten Code:
    McLaren->coutTempo();
    McLaren->beschleunigen(30);
    // cout << "McLaren hat jetzt eine Geschwindigkeit von " << McLaren->Tempo() << " km/h\n";
    McLaren->coutTempo();

    cout << endl;
    Formel1[0]->beschleunigen(180);
    Formel1[0]->bremsen();
    Formel1[0]->bremsen();
    Formel1[0]->bremsen();
    Formel1[0]->bremsen();
```

```
51      // cout << "Formel1-0 hat jetzt eine Geschwindigkeit von " << Formel1[0]->Tempo() ↵
    << " km/h\n";
52      Formel1[0]->coutTempo();
53
54      cout << endl;
55      Formel1[1]->MotorStarten();
56      Formel1[1]->beschleunigen(180);
57      Formel1[1]->bremsen();
58      Formel1[1]->coutTempo();
59      Formel1[1]->beschleunigen(60);
60      Formel1[1]->beschleunigen(42);
61      Formel1[1]->coutTempo();
62      Formel1[1]->bremsen();
63      Formel1[1]->MotorStoppen();
64      Formel1[1]->bremsen();
65      Formel1[1]->bremsen();
66      Formel1[1]->coutTempo();
67
68      cout << endl;
69      DTM[0].beschleunigen(200);
70      DTM[0].coutTempo();
71
72      system("pause");
73      return 0;
74  }
75
```

```
1  // Rennwagen4.h (Aufgabe 4)
2
3  const int MaxSpeed=380; // Maximales Tempo der Rennwagen
4
5  enum Tempo {minimal=30,wenig=75,moderat=110,viel=150,maximal=180};
6  enum Bremse {voll=MaxSpeed,sacht=50,stark=100};
7
8  class Rennwagen {
9  private:
10     int kmh;
11     // die beiden neuen Eigenschaften:
12     string Name;
13     bool MotorLaeuft;
14 public:
15     // beim alten Konstruktor muss jetzt der Name nachgefragt werden
16     Rennwagen() {
17         cout << "Wie heisst der Rennwagen? ";
18         cin >> Name;
19         kmh = 0;
20         MotorLaeuft = false;
21     }
22     // der neue Konstruktor bekommt den Namen übergeben
23     Rennwagen(string n) :Name(n) { kmh = 0; MotorLaeuft = false; }
24
25     // beschleunigen mit Maximalwert-Berücksichtigung
26     // MSVisualStudio akzeptiert enums als int sowie Standardwerte in der Funktionsdefinition ohne Prototyp,
27     // das ist nicht ganz die reine Lehre aber erleichtert natürlich die Schreibarbeit:
28     // Wir müssen nur die Argumentliste im Funktionskopf anpassen!
29     void beschleunigen(int K=moderat) {
30         char Eingabe = 'j';
31         if (!MotorLaeuft) {
32             cout << "Der Motor des " << Name << " ist noch aus!\n";
33             cout << "Soll der Motor jetzt angelassen werden (j/n)? ";
34             cin >> Eingabe;
35         }
36         // Eingabe ist per Initialisierung ODER Benutzereingabe j - oder auch nicht:
37         // Diese Programmiermethode ist zwar sehr effizient,
38         // vermischt aber eigentlich zwei funktionell getrennte Vorgänge (Nachfrage und anlassen / beschleunigen)
39         if (Eingabe == 'j') {
40             MotorLaeuft = true;
41             kmh += K;
42             if (kmh > MaxSpeed) kmh = MaxSpeed;
43         }
44     }
45
46     // bremsen (fast) wie bisher: akzeptiert auch enums und Standardwerte...
47     // void bremsen(enum K=sacht) würde NUR enums und keine bel. Werte akzeptieren
48     void bremsen(int K=sacht) {
49         kmh -= K;
50         if (kmh < 0) kmh = 0;
51     }
52
53     // Tempowert ausgeben wie bisher:
```

```
54      int Tempo() { return kmh; }
55
56      // Tempowert in Satz mit Name ausgeben:
57      void coutTempo() {
58          if (kmh>0) cout << "Der " << Name << " faehrt jetzt " << kmh << " km/h\n";
59          else cout << "Der " << Name << " steht!\n";
60      }
61
62      // Neue Methoden, um den Motor zu starten / zu stoppen:
63      void MotorStarten() {
64          if (MotorLaeuft)
65              cout << "Der Motor des " << Name << " laeuft bereits...\n";
66          else {
67              MotorLaeuft = true;
68              cout << "Der Motor des " << Name << " wurde angelassen.\n";
69          }
70      }
71      void MotorStoppen() {
72          if (kmh == 0) {
73              MotorLaeuft = false;
74              cout << "Der Motor des " << Name << " ist jetzt aus!\n";
75          }
76          else cout << "Der " << Name << " faehrt noch! Sie sollten den Motor jetzt
            nicht ausschalten!\n";
77      }
78  };
```

```cpp
// Rennwagen4.cpp (Aufgabe 4)
//
#include "stdafx.h"
#include <iostream>
#include <string>
using namespace std;

#include "Rennwagen4.h"

int main()
{
    Rennwagen RedBull("Red Bull");                  // Objekt statisch erzeugen
    Rennwagen* McLaren = new Rennwagen("McLaren");  // Objekt dynamisch erzeugen;
    Rennwagen* Formel1[2];              // Array mit Zeigern auf Rennwagen erzeugen
    Formel1[0] = new Rennwagen;         // 1. Arrayelement anlegen
    Formel1[1] = new Rennwagen("Ferrari");  // 2. Arrayelement anlegen mit neuem Konstruktor

    Rennwagen DTM[2] = {                // und auch ein statisches Array anlegen:
        Rennwagen(),
        Rennwagen("Mercedes")
    };

    cout << endl;
    RedBull.beschleunigen(200);
    RedBull.bremsen(sacht);
    RedBull.bremsen();
    RedBull.bremsen(stark);
    RedBull.coutTempo();

    cout << endl;
    McLaren->beschleunigen(230);
    McLaren->bremsen();
    McLaren->bremsen(sacht);
    McLaren->bremsen();
    // cout << "McLaren hat jetzt eine Geschwindigkeit von " << McLaren->tempo() << " km/h\n";
    // Hier den Namen direkt hardcoded zu programmieren, ist nicht mehr sinnvoll,
    // denn der Rennwagen kann ja ganz anders heißen. string Name ist aber private...!
    // Deshalb Name public machen oder Methode dazu schreiben oder coutTempo() verwenden.
    // In der Realität müsste man das jetzt klären, hier verwende ich den kürzesten Code:
    McLaren->coutTempo();
    McLaren->beschleunigen(minimal);
    // cout << "McLaren hat jetzt eine Geschwindigkeit von " << McLaren->Tempo() << " km/h\n";
    McLaren->coutTempo();

    cout << endl;
    Formel1[0]->beschleunigen(200);
    Formel1[0]->bremsen(sacht);
    Formel1[0]->bremsen(30);    // Side-Effekt: nummerische Werte gehen jetzt auch hier...
    Formel1[0]->bremsen();      // ist das unerwünscht, müssten wir im Funktionskopf
```

```
    die Argumentliste anpassen
50      Formel1[0]->bremsen();
51      // cout << "Formel1-0 hat jetzt eine Geschwindigkeit von " << Formel1[0]->Tempo() ↵
         << " km/h\n";
52      Formel1[0]->coutTempo();
53
54      cout << endl;
55      Formel1[1]->MotorStarten();
56      Formel1[1]->beschleunigen(maximal);
57      Formel1[1]->bremsen();
58      Formel1[1]->coutTempo();
59      Formel1[1]->beschleunigen(maximal);
60      Formel1[1]->beschleunigen(viel);
61      Formel1[1]->coutTempo();
62      Formel1[1]->MotorStoppen();
63      Formel1[1]->beschleunigen(moderat);
64      Formel1[1]->coutTempo();
65      Formel1[1]->bremsen(voll);
66      Formel1[1]->coutTempo();
67
68      cout << endl;
69      DTM[0].beschleunigen(200);
70      DTM[0].coutTempo();
71
72      system("pause");
73      return 0;
74  }
75
```