```cpp
/*---------------------------------------------
Aufgabe5.cpp (Simulation Parkhaus)
(Examen IKA 8/10 - Sommer 2012)
---------------------------------------------*/
#include "stdafx.h"
#include <iostream>
#include <string>
using namespace std;

// Die Definition eines Stellplatzes:
struct ChipCoin {
    int EinfahrzeitStunde;
    int EinfahrzeitMinute;
    bool ausgegeben;
};

// Die Klassenvereinbarung Parkhaus
class Parkhaus {
    const string Bezeichnung;
    const int AnzahlStellplaetze;
    int AnzahlParkvorgaenge;
    int AktuelleBelegung;
    float Einnahmen;
    ChipCoin* Coin;
    const int FreiParkMinuten;
    const float GebuehrHalbeStunde;
public:
    Parkhaus(string B, int A);
    ~Parkhaus();
    void einfahren(int hh, int mm);
    void ausfahren(int Platz, int hh, int mm);
    void printInfo();
};

// Konstruktor mit Konstanteninitialisierung über Elementliste:
Parkhaus::Parkhaus(string B, int A) :
         Bezeichnung(B),AnzahlStellplaetze(A),FreiParkMinuten(30),GebuehrHalbeStunde
(.5) {
    AnzahlParkvorgaenge = 0;
    AktuelleBelegung = 0;
    Einnahmen = .0;
    Coin = new ChipCoin[A];
    for (int i=0; i<A; i++) Coin[i].ausgegeben = false;
}

// Dekonstruktor, um dyn. Array frei zu geben:
Parkhaus::~Parkhaus() {
    delete[] Coin;
}

// Die weiteren Methoden:
void Parkhaus::einfahren(int hh, int mm) {
    if (AktuelleBelegung >= AnzahlStellplaetze) {
        cout << "Das Parkhaus ist voll belegt. Bitte warten Sie.\n";
    } else {
        // Suche nächsten freien Stellplatz:
```

```
56          int Index = 0;
57          while(Coin[Index].ausgegeben) {Index++;}
58          // gefunden:
59          Coin[Index].ausgegeben = true;
60          Coin[Index].EinfahrzeitStunde = hh;
61          Coin[Index].EinfahrzeitMinute = mm;
62          AktuelleBelegung ++;
63          AnzahlParkvorgaenge ++;
64          cout << "Sie erhalten den ChipCoin mit der Nr. " << Index << endl;
65      }
66  }
67
68  void Parkhaus::ausfahren(int Nr, int hh, int mm) {
69      // Ist der Platz belegt?
70      if (Coin[Nr].ausgegeben) {
71          // Gebühr berechnen:
72          // Dazu Parzeit feststellen:
73          int Parkzeit = 60*hh - 60*Coin[Nr].EinfahrzeitStunde + mm - Coin
            [Nr].EinfahrzeitMinute;
74          cout << "Fuer Ihre Parkdauer von " << Parkzeit << " Minuten werden ";
75          // Jetzt Gebühren berechnen: Innerhalb der FreiParkZeit?
76          if (Parkzeit <= FreiParkMinuten)
77              cout << "keine Gebuehren verlangt!\n";
78          else {
79              // Darüber hinaus gehende Parkzeit durch halbe Stunde teilen
80              cout << ((Parkzeit - FreiParkMinuten)/30 + 1)*GebuehrHalbeStunde << "
                Euro berechnet.\n";
81              Einnahmen += ((Parkzeit - FreiParkMinuten)/30 + 1)*GebuehrHalbeStunde;
82          }
83          // Stellplatz freigeben:
84          Coin[Nr].ausgegeben = false;
85          AktuelleBelegung --;
86      }
87      else
88          cout << "Der von Ihnen angegebene ChipCoin ist nicht gueltig...\n";
89  }
90
91  void Parkhaus::printInfo() {
92      cout << "\nDas Parkhaus " << Bezeichnung << " hat " << AnzahlStellplaetze << "
        Stellplaetze, ";
93      cout << "davon sind " << AktuelleBelegung << " zur Zeit belegt.\n";
94      cout << "Insgesamt wurden mit " << AnzahlParkvorgaenge << " Parkvorgaengen
        Einnahmen von " << Einnahmen << " Euro erzielt.\n\n";
95  }
96
97  // Das Hauptprogramm:
98  int main()
99  {
100     Parkhaus P("Post",100);
101
102     for (int i=0; i<60; i++)
103     {P.einfahren(8,i);}
104     P.printInfo();
105     system("pause");
106
107     for (int i=0; i<60; i+=10) P.ausfahren(i,9,10);
```

```
108     P.printInfo();
109     system("pause");
110 
111     for (int i=0; i<50; i++) P.einfahren(9,45);
112     P.printInfo();
113     system("pause");
114 
115     for (int i=10; i<21; i++) P.ausfahren(i,10,15);
116     P.printInfo();
117     system("pause");
118 
119     for (int i=0; i<100; i+=5) P.ausfahren(i,12,00);
120     P.printInfo();
121     system("pause");
122 
123     return 0;
124 }
125 
```