```
 1 //============================================================================
 2 // Project   : C++-Kurs @ Sultex Limited
 3 // Modul     : C++
 4 // Title     : Übung 2 "Klasse": Ausgangslage
 5 // Author    : 'Ihr Name'
 6 // Tab-Width : 2
 7 /*///==========================================================================
 8
 9 * Description:
10 Klasse implementieren aufgrund applikatorischer Vorgaben.
11 Mit Strings als char-Array arbeiten.
12 Bearbeitung von Arrays aus Objekten.
13 Allozierung auf Stack und Heap vergleichen.
14 Performanceuntersuchung mit inline-Methoden (-> Kompiler-Option: -O2)
15
16 * History    :
17 03.10.05: Initial Version.
18
19 /*///==========================================================================
20 //       1         2         3         4         5         6         7         8
21 //345678901234567890123456789012345678901234567890123456789012345678901234567890
22 //============================================================================
23 #ifndef KLASSE_H
24 #define KLASSE_H 1
25
26 #include <cstring>
27 #include <iostream>
28
29 #include "StatisticStopWatch.h"
30
31 using namespace std;
32
33
34
35 // Klassen-Definitionen:
36
37 // Person:
38 // Eine Person hat einen Namen (20 Char's) und eine Nummer.
39 class Person {
40
41   public:
42     int           getNr();
43     const char*   getName();
44
45   private:
46
47 };
48
49
50 // Funktions-Prototypen:
51
52 // Ausgabe eines Arrays von Personen-Objekten auf die Konsole.
53 //   pPers: Pointer auf erstes Personen-Objekt im Array.
54 //   pLen:  Laenge des Personen-Arrays.
55 void printPersArr(Person* pPers, int pLen);
56
57
58 #endif /*KLASSE_H*/
```



```
 1 #include "Klasse.h"
 2
 3 // Methoden-Implementationen:
 4
 5 //Person::Person() {
 6 // ...
 7 //}
 8
 9
10
11
12 // Funktions-Implementationen:
13
14 void printPersArr(Person* pPers, int pLen) {
15
16 }
17
```

```
#include "Klasse.h"

int main() {

  const int MAX_ARR = 5;

  Person persArr[MAX_ARR];
/*
  for (int i = 0; i < MAX_ARR; i++) {
    cout << persArr[i].getNr() << ": " << persArr[i].getName() << endl;
  }
  persArr[0].setName("Miller");
  persArr[0].setNr(1);
  persArr[1].setName("Bond");
  persArr[1].setNr(007);
  for (int i = 0; i < MAX_ARR; i++) {
    cout << persArr[i].getNr() << ": " << persArr[i].getName() << endl;
  }

  // Ausgabe des Personen-Arrays wie oben mit for-Schleifen, jetzt aber mit
  // Funktion 'printPersArr()':
  printPersArr(persArr, MAX_ARR);



  // Untersuchung wieviel Zeit Objekt-Allozierungen auf Stack und Heap benoetigen:

  StatisticStopWatch stopWatch;
  const int MAX_LOOP = 5000000;


  // Allozierung auf dem Stack:
  stopWatch.reset();
  stopWatch.start();
  for (int i = 0; i < MAX_LOOP; i++) {
    Person pers(i, "John");
  }
  stopWatch.stop();
  stopWatch.printTime("Stack");


  // Allozierung auf dem Heap:
  stopWatch.reset();
  stopWatch.start();
  for (int i = 0; i < MAX_LOOP; i++) {
    Person* pers = new Person(i, "John");
  }
  stopWatch.stop();
  stopWatch.printTime("Heap: new()");

```



```

  // Tests mit 'inline-Methoden':
  int max_loop = 100 * MAX_LOOP;
  stopWatch.reset();
  stopWatch.start();
  Person pers;
  for (int i = 0; i < max_loop; i++) {
    pers.getNr();
  }
  stopWatch.stop();
  stopWatch.printTime("getNr()");
  stopWatch.reset();

  stopWatch.reset();
  stopWatch.start();
  for (int i = 0; i < max_loop; i++) {
    pers.getNrInline();
  }
  stopWatch.stop();
  stopWatch.printTime("getNrInline()");
  stopWatch.reset();
*/
}




/* Session-Log (Intel Pentium M / 1700 MHz; g++ 3.4.4):

$ make clean
rm -f Klasse.o KlasseTest.o StatisticStopWatch.o appl.exe

$ make all
g++  -g -c Klasse.cpp
g++  -g -c KlasseTest.cpp
g++  -g -c StatisticStopWatch.cpp
g++ -g -o appl.exe Klasse.o KlasseTest.o StatisticStopWatch.o

$ appl
-1:
-1:
-1:
-1:
-1:
1: Miller
7: Bond
-1:
-1:
-1:
1: Miller
7: Bond
-1:
-1:
-1:
Stack                =        218 ms
Heap: new()          =       7657 ms
getNr()              =       4140 ms
getNrInline()        =       1031 ms

$

*/

```

```
  1 //======================================================================
  2 //        * Letsch Informatik *       www.LetsInfo.ch       CH-8636 Wald
  3 //          Beratung, Ausbildung und Realisation in Software-Engineering
  4 //======================================================================
  5 // Project   : Nachdiplomstudium Software-Engineering NDS-SE
  6 // Modul     : C++
  7 // Title     : Support-Klasse  "Statistic-Stopwatch" : StatisticStopWatch.h
  8 // Author    : Thomas Letsch
  9 // Tab-Width : 2
 10 /*///====================================================================
 11 * Description: Klasse für Zeitmessungen mit Statistic-Funktion.
 12               Zu 'bedienen' wie eine Stop-Uhr.
 13 * History    : 26.08.94: Initial Version
 14               13.12.02: Kosmetik
 15               02.01.04: printTime() hinzugefuegt.
 16 * Version    : $Revision: 1.5 $  $Date: 2005/09/29 20:14:49 $
 17 /*///====================================================================
 18 //       1         2         3         4         5         6         7         8
 19 //345678901234567890123456789012345678901234567890123456789012345678901234567890
 20 //======================================================================
 21
 22 #ifndef STATISTICSTOPWATCH_H
 23 #define STATISTICSTOPWATCH_H 1
 24
 25 #include <iostream>
 26 #include <sys/time.h>
 27 #include <climits>
 28 #include <cstdlib>
 29 #include <cstdio>
 30
 31 using namespace std;
 32
 33 class StatisticStopWatch {
 34   public:
 35
 36     StatisticStopWatch();
 37
 38     int  start();        // Start a Measurement
 39     int  stop();         // Stop  a Measurement
 40     void reset();        // Reset StopWatch to Zero
 41
 42     // Print Time-Statistic to stdout (in Milliseconds).
 43     // pDescription: Prefix which is printed first.
 44     void printTimes(const char* pDescription);
 45
 46     // Print Time to stdout (in Milliseconds).
 47     // This Methode shall only be used with just one Measurement.
 48     // pDescription: Prefix which is printed first.
 49     // pDivisor:     The actual measured Time will be diveded by pDivisor.
 50     // return:       'false' if mCounter != 0, otherwise 'true'.
 51     bool printTime(const char* pDescription, int pDivisor = 1);
 52
 53   private:
 54     void timeStatistic();
 55
 56     int     mCounter;       // Number of Measurements
 57     long    mLast;          // Last messured Time
 58
 59     long    mMin;           // Minimal messured Time
 60     long    mMax;           // Maximal messured Time
 61     double  mAvg;           // Average of all messured Times
 62
 63     struct  timeval mStart; // Begin-Time of one Measurement
 64     struct  timeval mStop;  // End-Time   of one Measurement
 65     bool    mMeasuring;     // True while measuring
 66     struct  timezone mTzp;  // not used
 67
 68 }; // class StatisticStopWatch
 69 #endif
```



```
  1
  2 # Makefile für "Klasse"                          30.09.2005
  3
  4 BIN    = appl.exe
  5 OBJS   = Klasse.o KlasseTest.o StatisticStopWatch.o
  6
  7 CC     = g++
  8 CFLAGS = -g
  9
 10 all: $(BIN)
 11
 12 $(BIN): $(OBJS)
 13   $(CC) $(CFLAGS) -o $(BIN) $(OBJS)
 14
 15 clean:
 16   rm -f $(OBJS) $(BIN)
 17
 18 %.o: %.cpp %.h
 19   $(CC) $(CFLAGS) -c $<
 20
 21 %.o: %.cpp
 22   $(CC) $(CFLAGS) -c $<
```