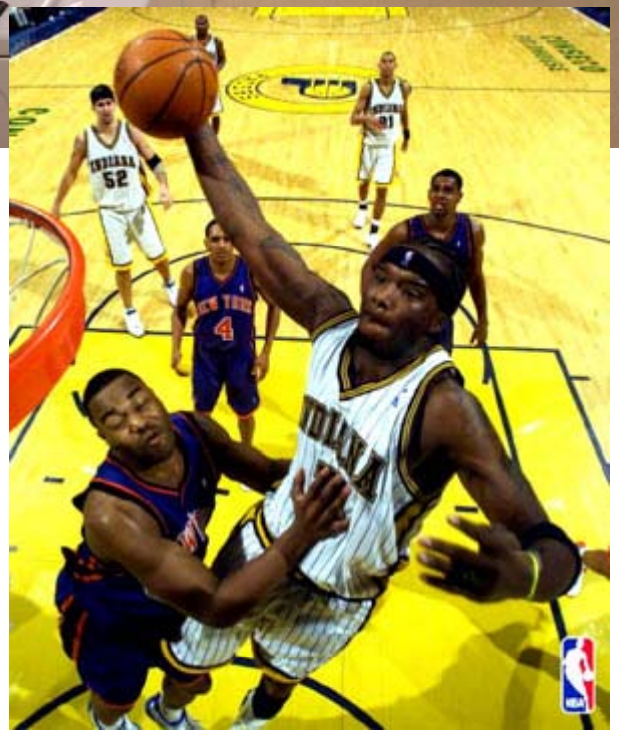
# ECE 462
# Object-Oriented Programming using C++ and Java

# Parallelism

Yung-Hsiang Lu
yunglu@purdue.edu

# Parallelism

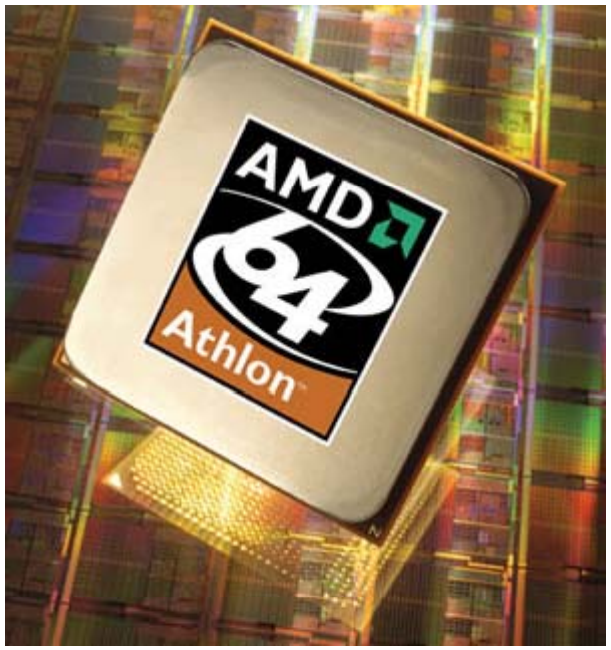
- parallel: several activities occurring simultaneously
- Parallelism is natural phenomena is our daily lives
  - Many students take notes in a class.
  - Multiple cars wait for a traffic light.
  - Several lines of people order fast food (and eat).
  - A student uses several washers in a laundry room.
  - Several players try to catch a basketball.
  - A hundred customers order books on-line.



# Sequential Programming Model

- One statement executes first, then the next statement.
- Control structures (if, for, while, function call ...) may change the sequence of execution.
- Within a small segment (inside if, for, while, function ...), statements are executed sequentially, one by one.
- Sequential programming model was appropriate when a computer had only one processor.

# Parallel Computer on Your Desktop

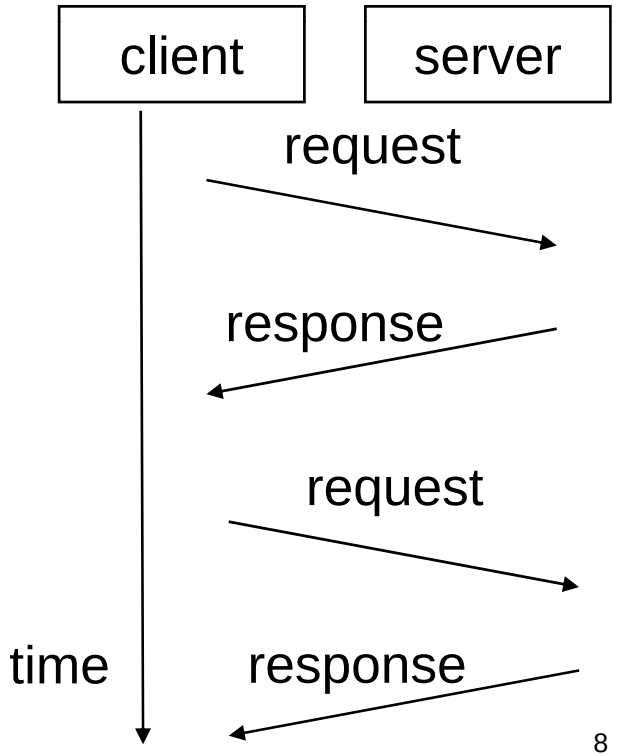



# Parallel Programming Models

- Multi-threading is one of the most popular, but not the only available programming model.
- Other models include
  - client-server
  - producer-consumer
  - pipeline
  - multithread
  - ...

# Client Server

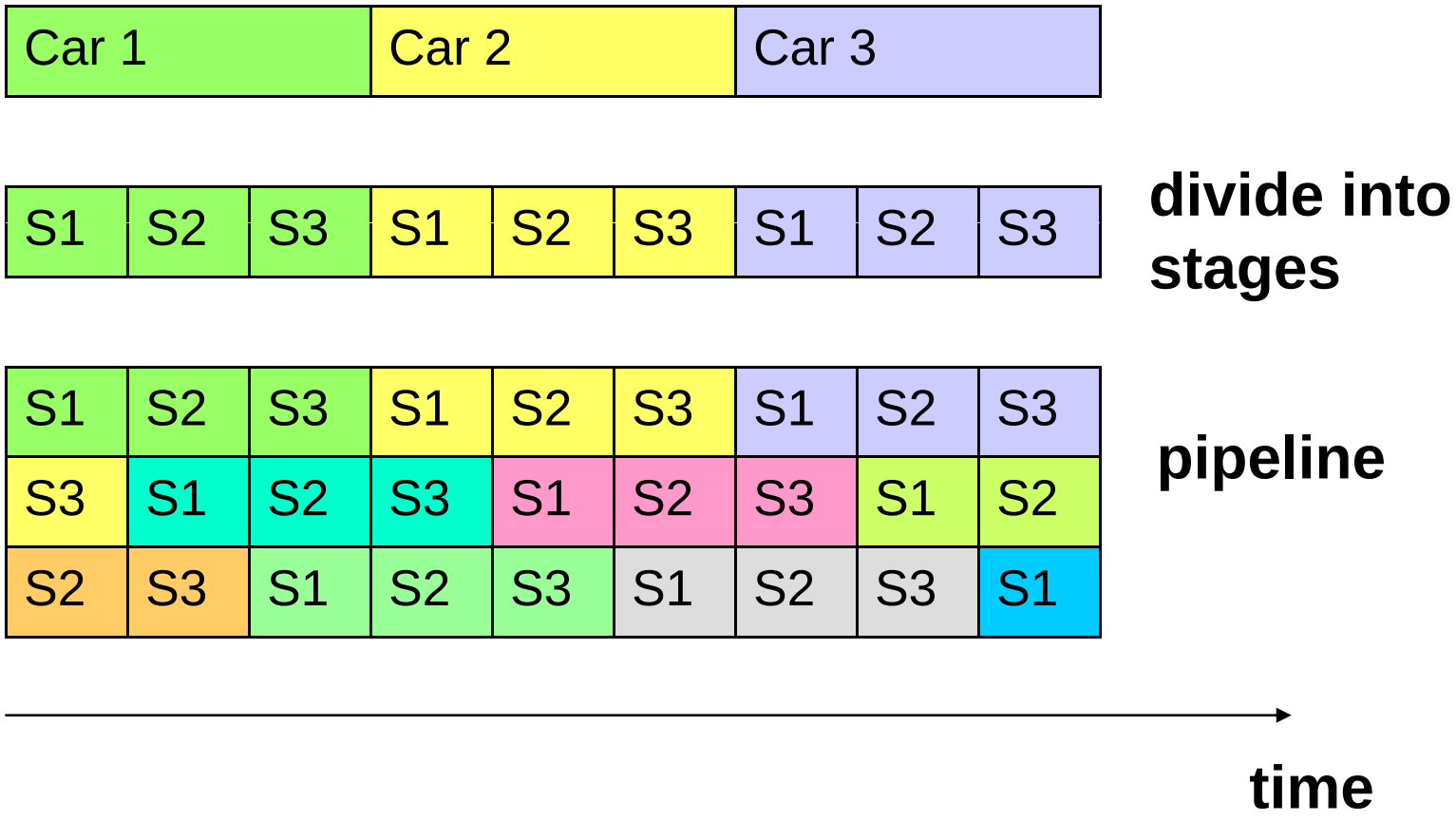
- web browser (client) — web server
- client sends request, server responds
- The client and the server may (in fact, very likely) be mulit-thread individually.



# Pipeline


- instruction-level parallelism
- factory assembly line
- buffet or BBQ line

# Latency and Throughput



# Requirements for Parallelism

- multiple processing units
- multiple data
- independence among the units and data
- (usually) some forms of agreements among the units and the data

# Multi-Thread

# Why is Multiple Thread Important?

- One of the most popular parallel programming models.
- Most machines (through libraries and languages) support multithreads.
- Multiple threads allow a program to respond to different requests quickly.
- You have been using multi-thread programs for years: web browser and most GUI programs.
- It demonstrates many important concepts.

# What is a “thread”?

- A thread is (almost) an independent program except
  - different threads created in the same program can access shared variables / objects
  - smaller overhead compared with processes in context switch
  - once a thread starts running, it runs independently until completion or synchronization
- context switch = the processor changes which thread to execute
- OOP is a natural approach for parallelism. Objects are **independent** and **active**.

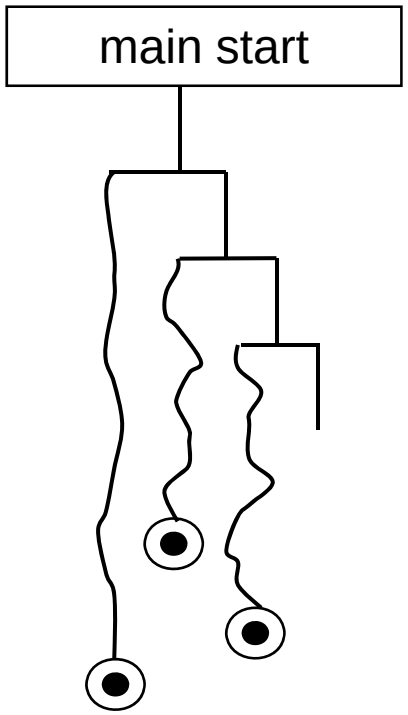
# ECE 462
# Object-Oriented Programming using C++ and Java

# Threads

Yung-Hsiang Lu
yunglu@purdue.edu

# Java Thread

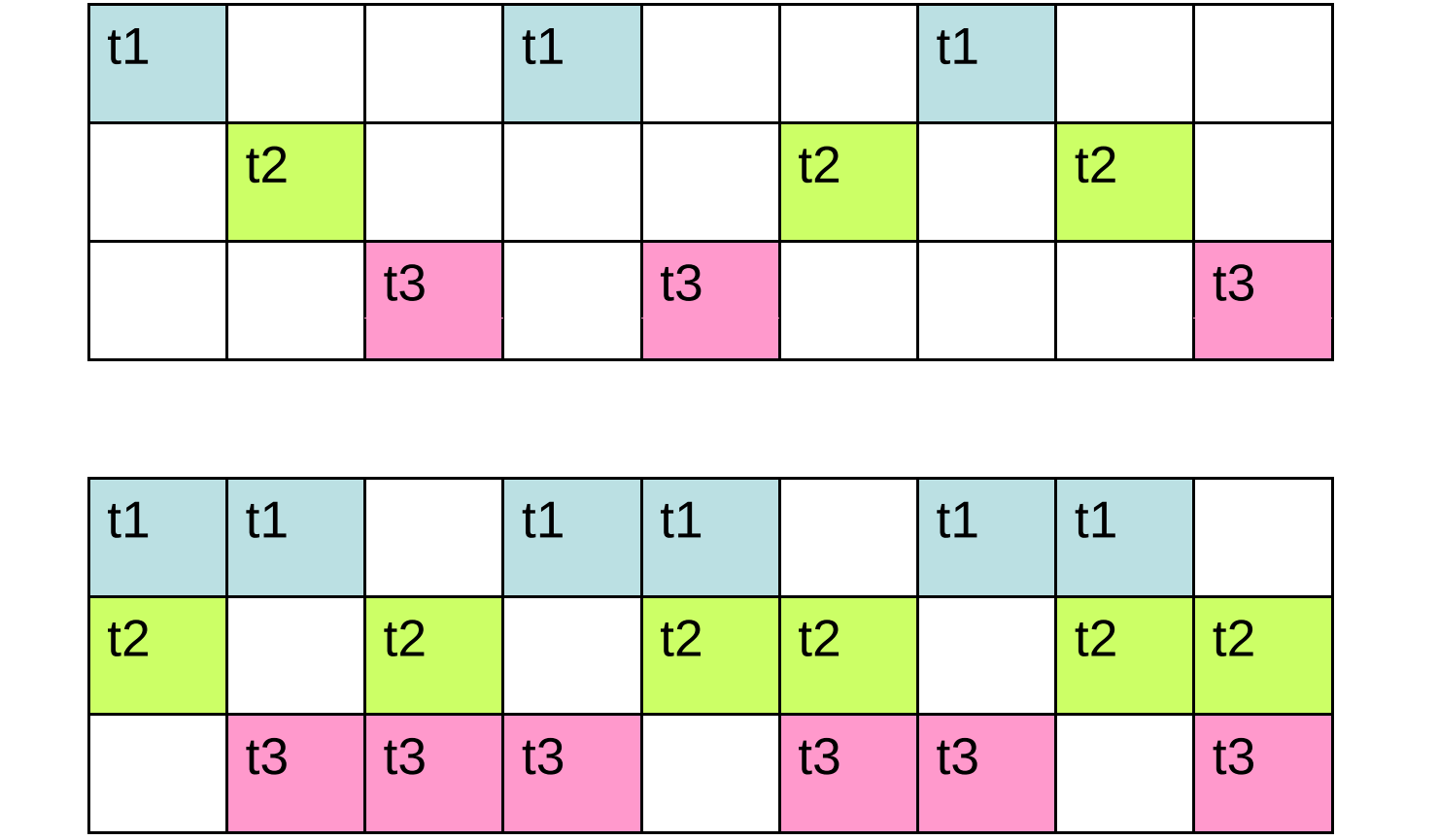
```
class YourClass extends Thread {
    public void run () {
        // what does this thread do?
    }
}
...
public static void main(String [] args) {
    YourClass t1 = new YourClass("...");
    t1.start();
}
```



# Qt Thread

```
#include <QtCore>
class YourClass extends QThread {
    public void run () {
        // what does this thread do?
    }
};
...
int main(int argc, char * argv[]) {
    YourClass t1 ("...");
    t1.start();
}
```

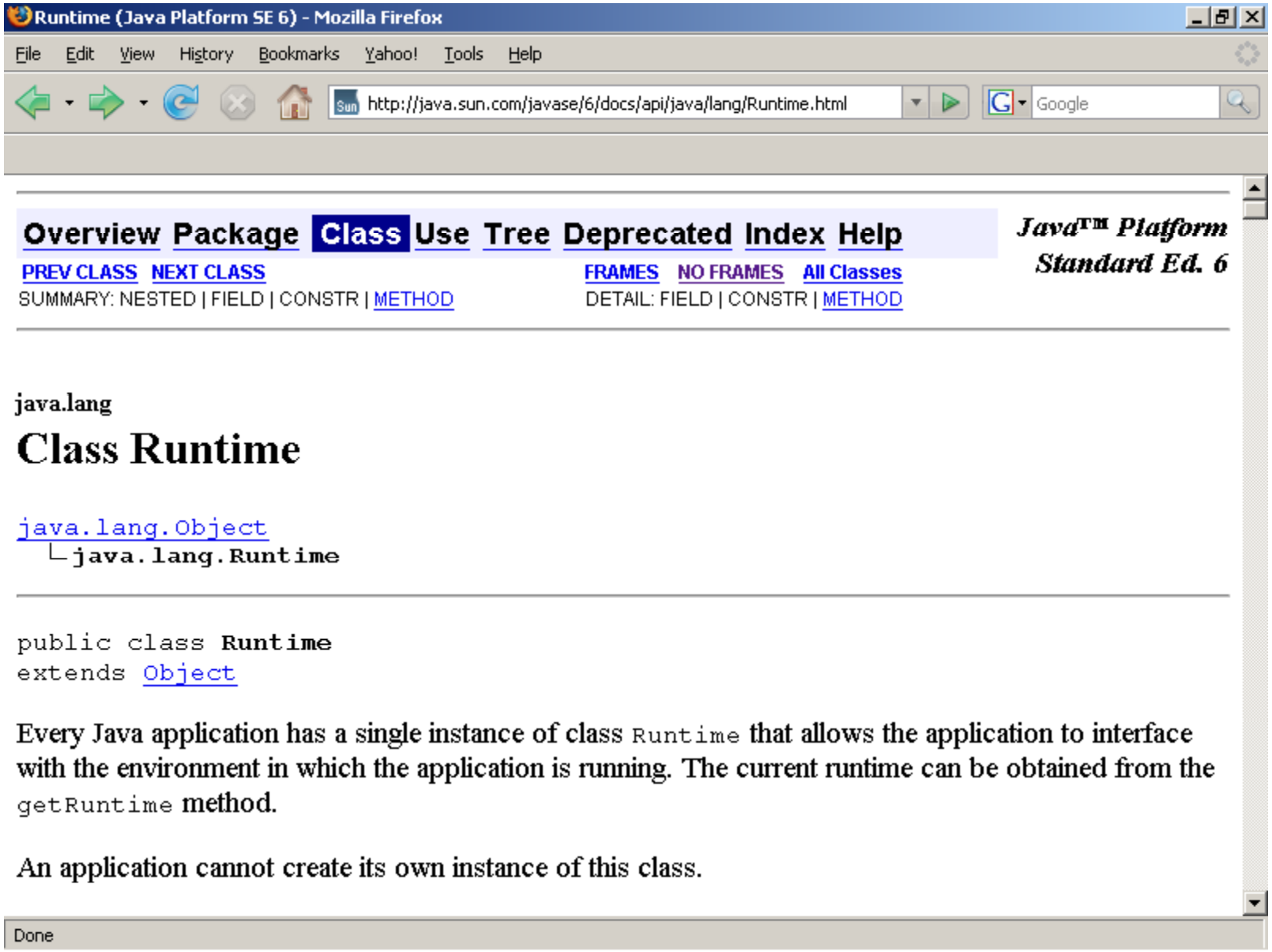
# Thread Execution Time

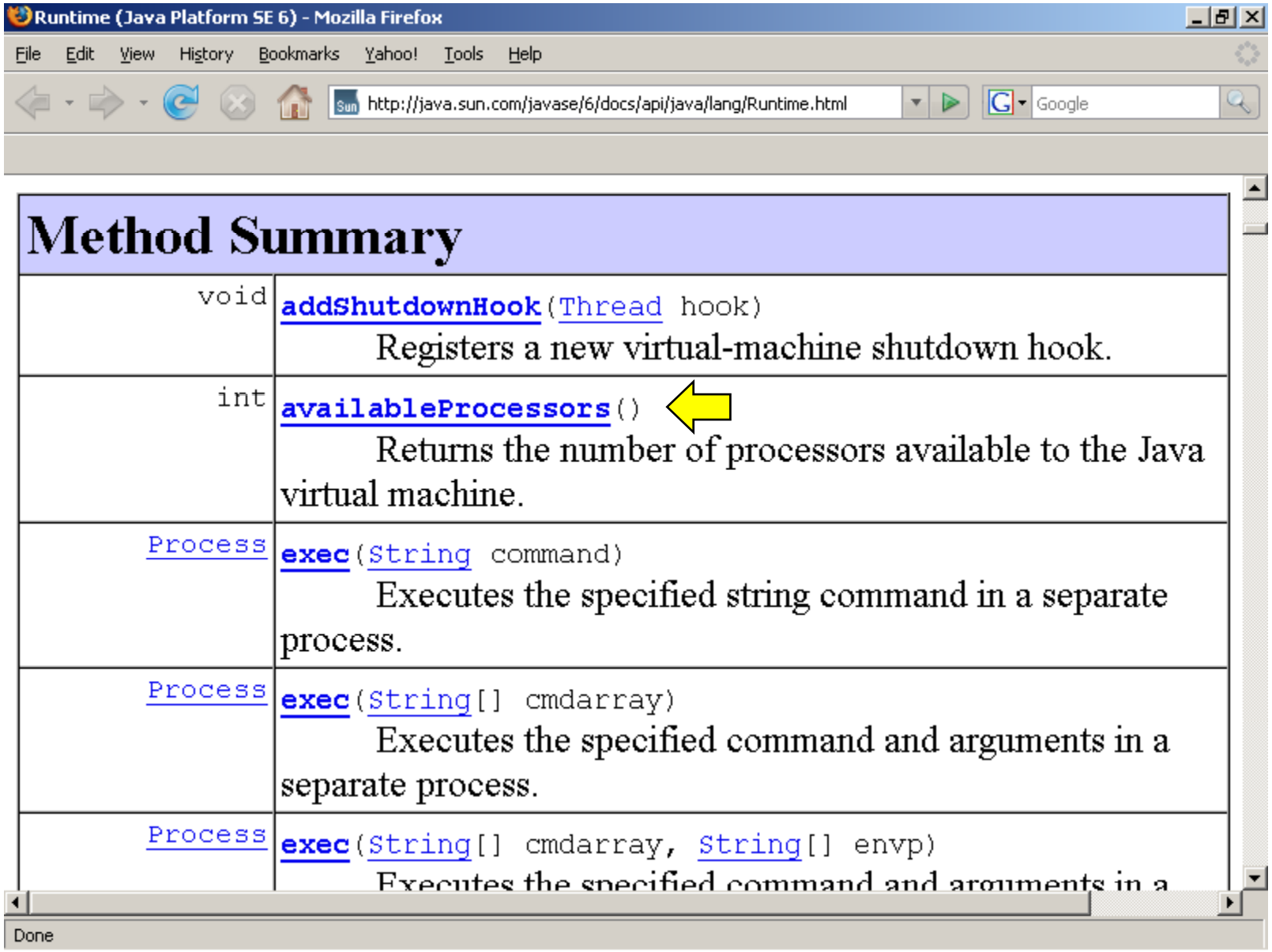


# Many Threads, Few Processors

- advantages of many threads, even on single processor
  - impression of continuous progress in all threads
  - improve utilization of different components
  - handle user inputs more quickly
- disadvantage of many threads
  - add slight work to program structure
  - incur overhead in thread creation
  - cause complex interleaving the execution and **possibly wrong results** (if you do not think "in parallel")

# Java Thread Examples


Runtime (Java Platform SE 6) - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://java.sun.com/javase/6/docs/api/java/lang/Runtime.html
Google
Overview Package Class Use Tree Deprecated Index Help
Java™ Platform Standard Ed. 6
PREV CLASS NEXT CLASS
FRAMES NO FRAMES All Classes
SUMMARY: NESTED | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD
java.lang
Class Runtime
java.lang.Object
└java.lang.Runtime
public class Runtime
extends Object
Every Java application has a single instance of class Runtime that allows the application to interface with the environment in which the application is running. The current runtime can be obtained from the getRuntime method.
An application cannot create its own instance of this class.
Done


Runtime (Java Platform SE 6) - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://java.sun.com/javase/6/docs/api/java/lang/Runtime.html
Google
Method Summary
void
addShutdownHook(Thread hook)
Registers a new virtual-machine shutdown hook.
int
availableProcessors()
Returns the number of processors available to the Java virtual machine.
Process
exec(String command)
Executes the specified string command in a separate process.
Process
exec(String[] cmdarray)
Executes the specified command and arguments in a separate process.
Process
exec(String[] cmdarray, String[] envp)
Done

# Matrix Addition

| | | | |
|---|---|---|---|
| B11 | B12 | B13 | B14 |
| B21 | B22 | B23 | B24 |
| B31 | B32 | B33 | B34 |
| B41 | B42 | B43 | B44 |

| | | | |
|---|---|---|---|
| A11 | A12 | A13 | A14 |
| A21 | A22 | A23 | A24 |
| A31 | A32 | A33 | A34 |
| A41 | A42 | A43 | A44 |

| | | | |
|---|---|---|---|
| C11 | C12 | C13 | C14 |
| C21 | C22 | C23 | C24 |
| C31 | C32 | C33 | C34 |
| C41 | C42 | C43 | C44 |

# Divide Matrix Addition

| thread 1 |
| --- |
| thread 2 |

| B11 | B12 | B13 | B14 |
| --- | --- | --- | --- |
| B21 | B22 | B23 | B24 |
| B31 | B32 | B33 | B34 |
| B41 | B42 | B43 | B44 |

| A11 | A12 | A13 | A14 |
| --- | --- | --- | --- |
| A21 | A22 | A23 | A24 |
| A31 | A32 | A33 | A34 |
| A41 | A42 | A43 | A44 |

| C11 | C12 | C13 | C14 |
| --- | --- | --- | --- |
| C21 | C22 | C23 | C24 |
| C31 | C32 | C33 | C34 |
| C41 | C42 | C43 | C44 |

# Matrix Multiplication

| | | | |
|---|---|---|---|
| B11 | B12 | B13 | B14 |
| B21 | B22 | B23 | B24 |
| B31 | B32 | B33 | B34 |
| B41 | B42 | B43 | B44 |

| | | | |
|---|---|---|---|
| A11 | A12 | A13 | A14 |
| A21 | A22 | A23 | A24 |
| A31 | A32 | A33 | A34 |
| A41 | A42 | A43 | A44 |

| | | | |
|---|---|---|---|
| C11 | C12 | C13 | C14 |
| C21 | C22 | C23 | C24 |
| C31 | C32 | C33 | C34 |
| C41 | C42 | C43 | C44 |

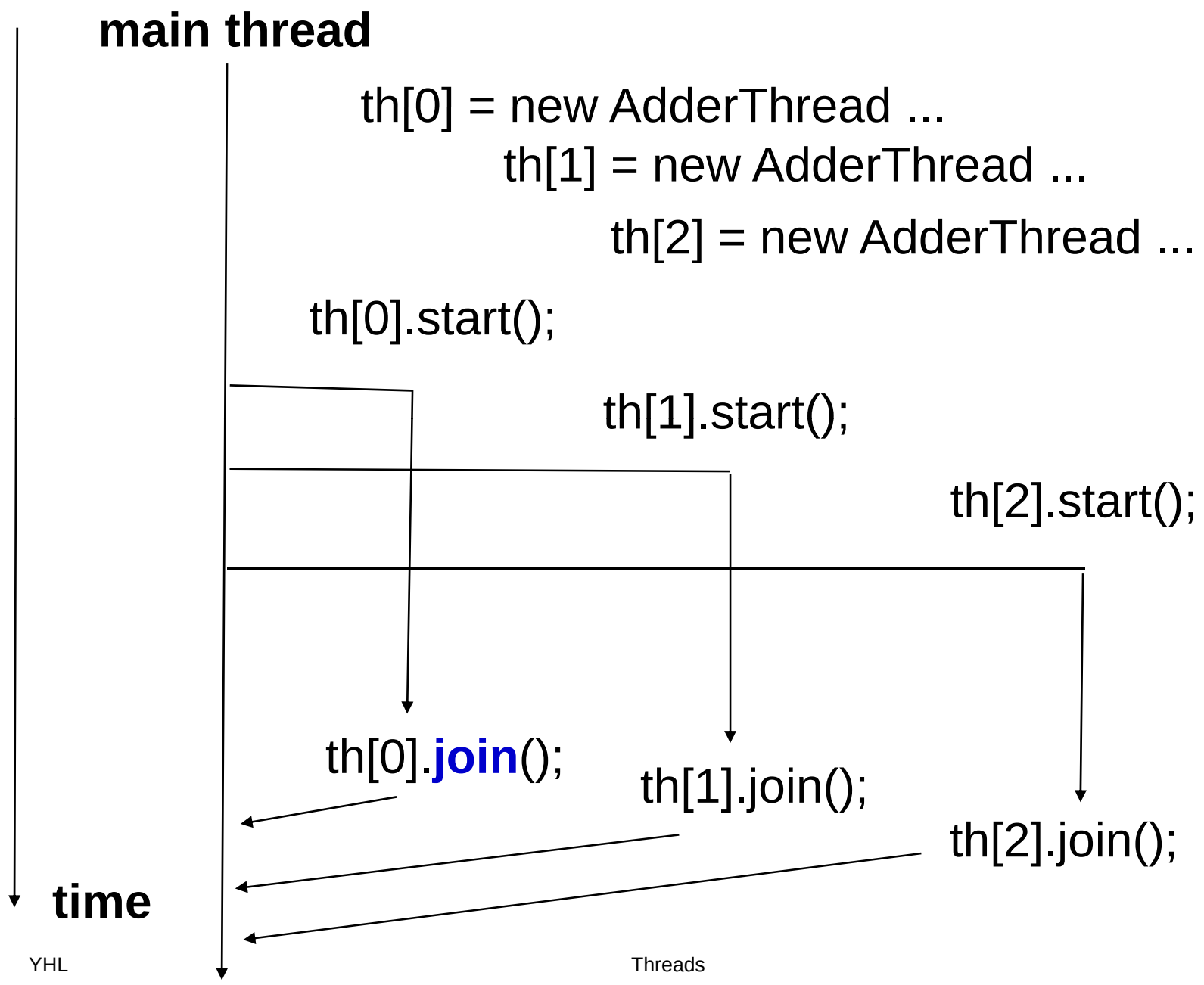
# Divide Matrix Multiplication

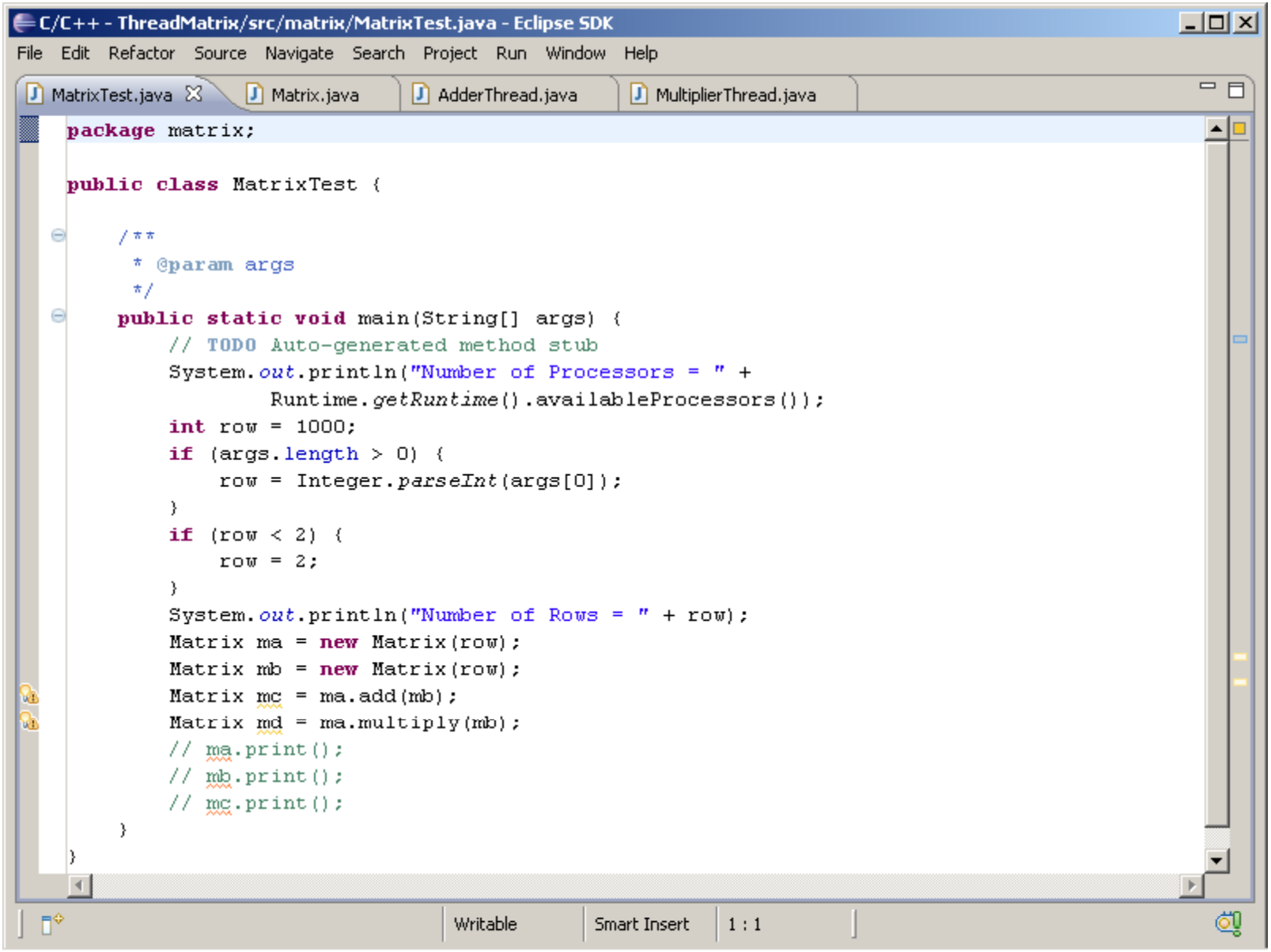
| thread 1 |
| --- |
| thread 2 |

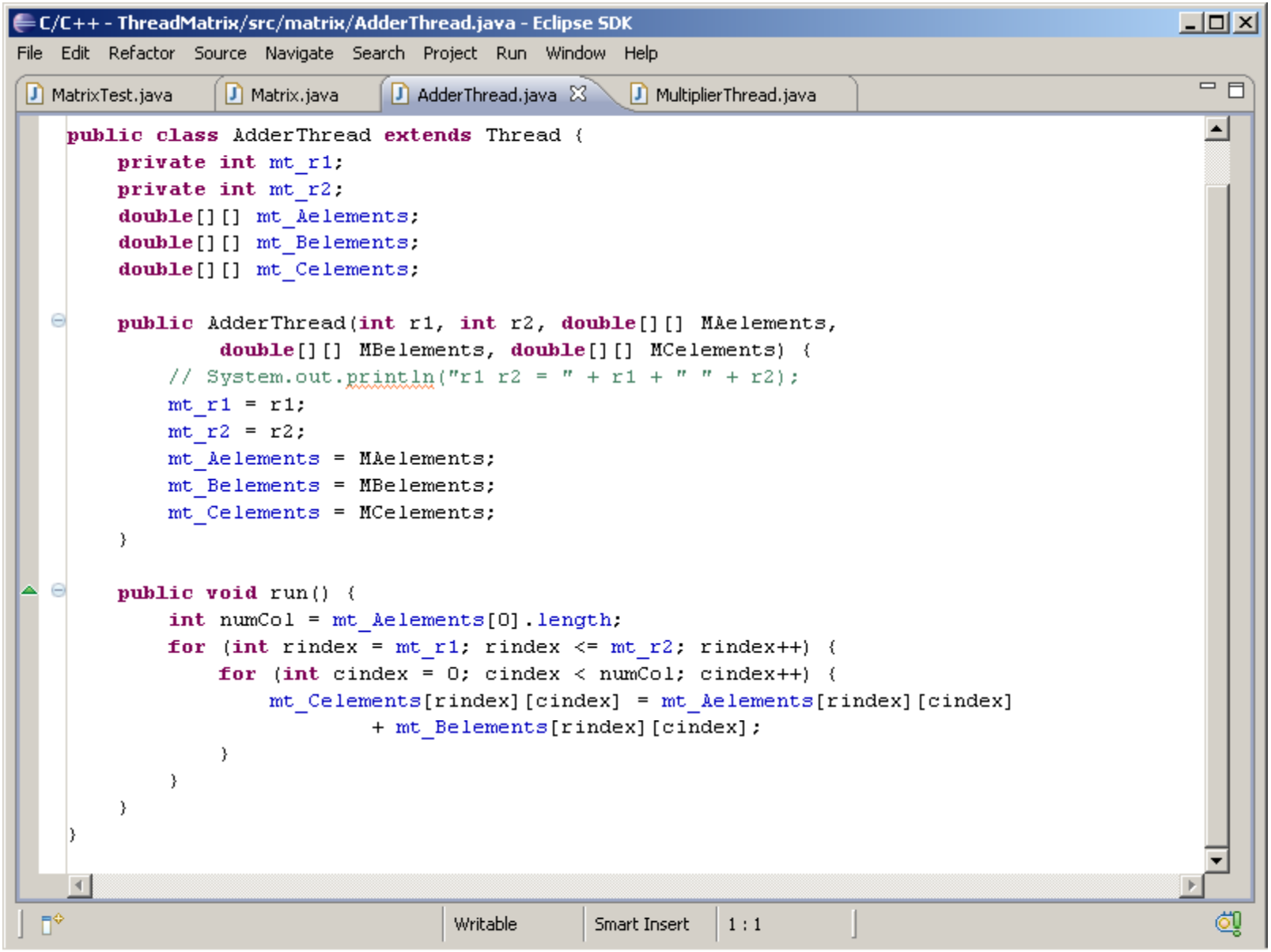
| B11 | B12 | B13 | B14 |
| --- | --- | --- | --- |
| B21 | B22 | B23 | B24 |
| B31 | B32 | B33 | B34 |
| B41 | B42 | B43 | B44 |

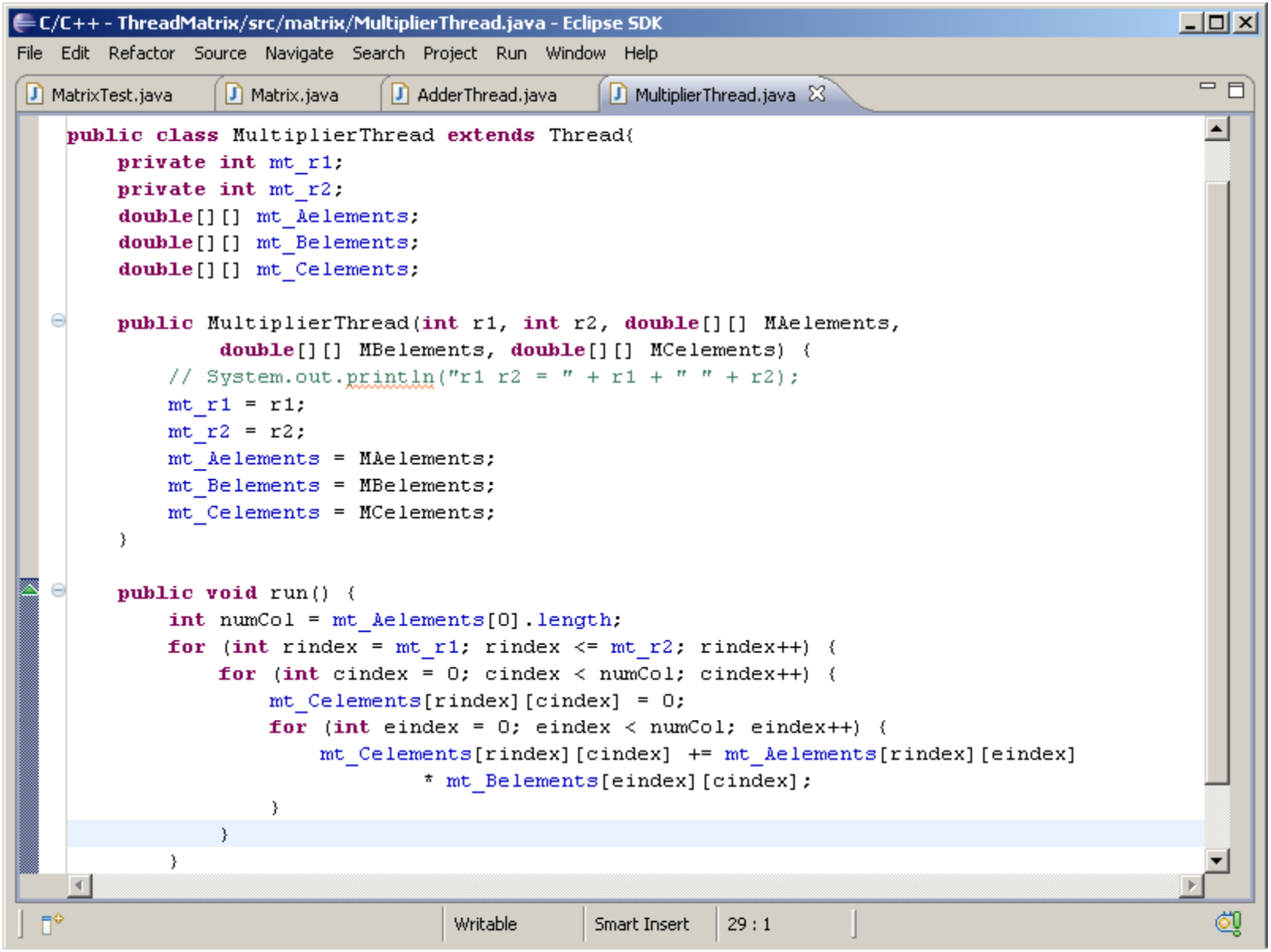
| A11 | A12 | A13 | A14 |
| --- | --- | --- | --- |
| A21 | A22 | A23 | A24 |
| A31 | A32 | A33 | A34 |
| A41 | A42 | A43 | A44 |

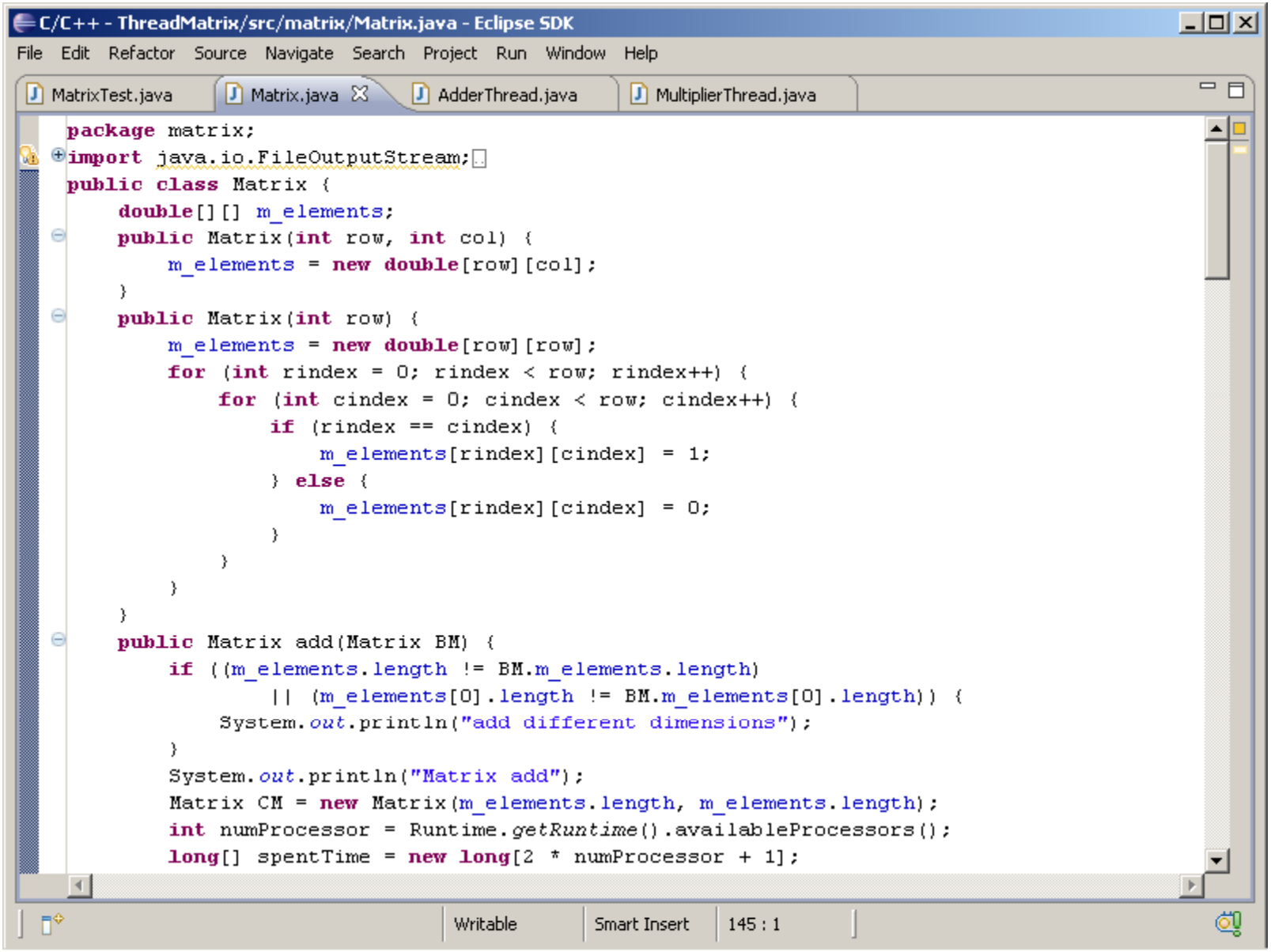
| C11 | C12 | C13 | C14 |
| --- | --- | --- | --- |
| C21 | C22 | C23 | C24 |
| C31 | C32 | C33 | C34 |
| C41 | C42 | C43 | C44 |

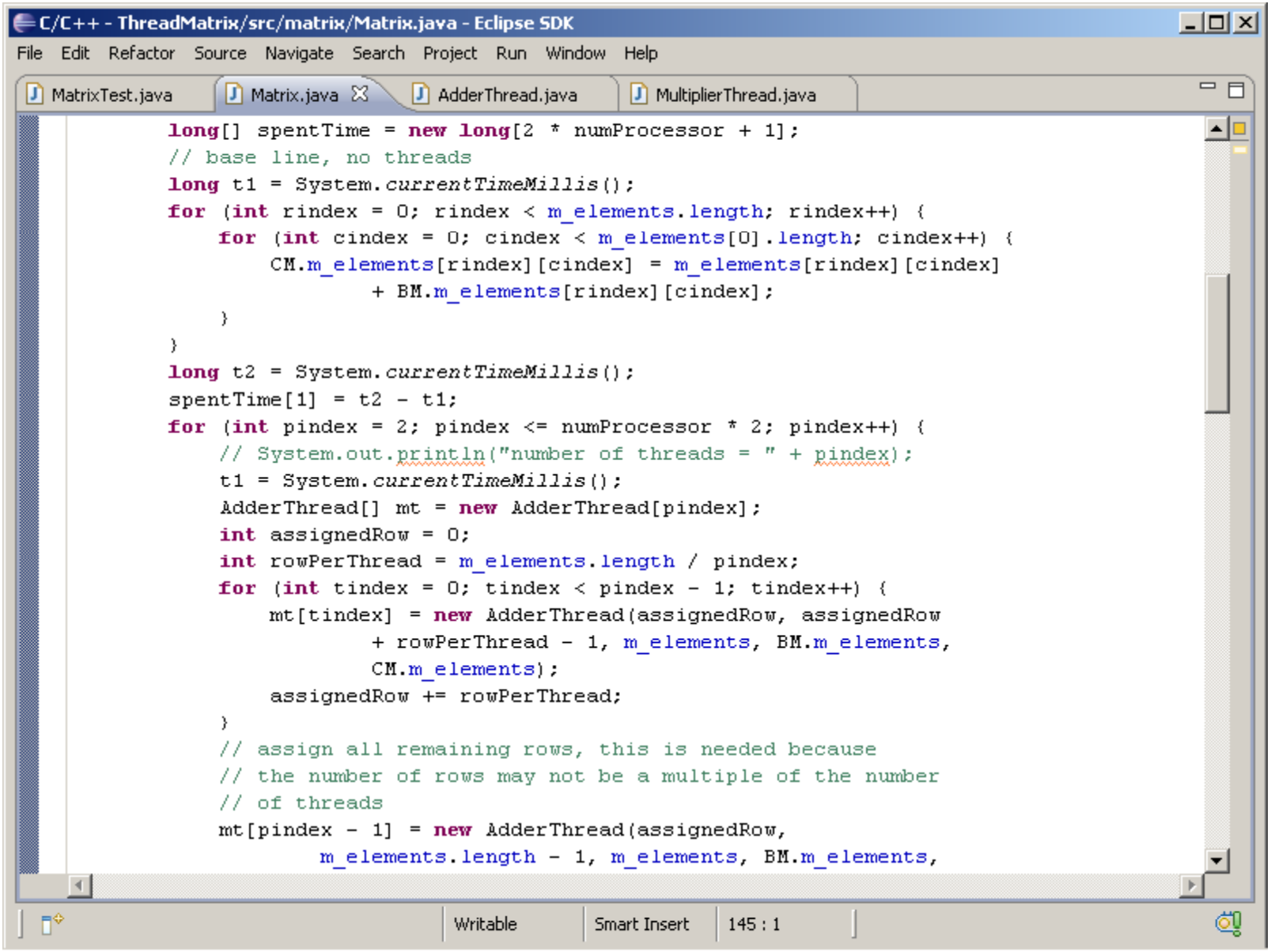

main thread
th[0] = new AdderThread ...
th[1] = new AdderThread ...
th[2] = new AdderThread ...
th[0].start();
th[1].start();
th[2].start();
th[0].join();
th[1].join();
th[2].join();
time

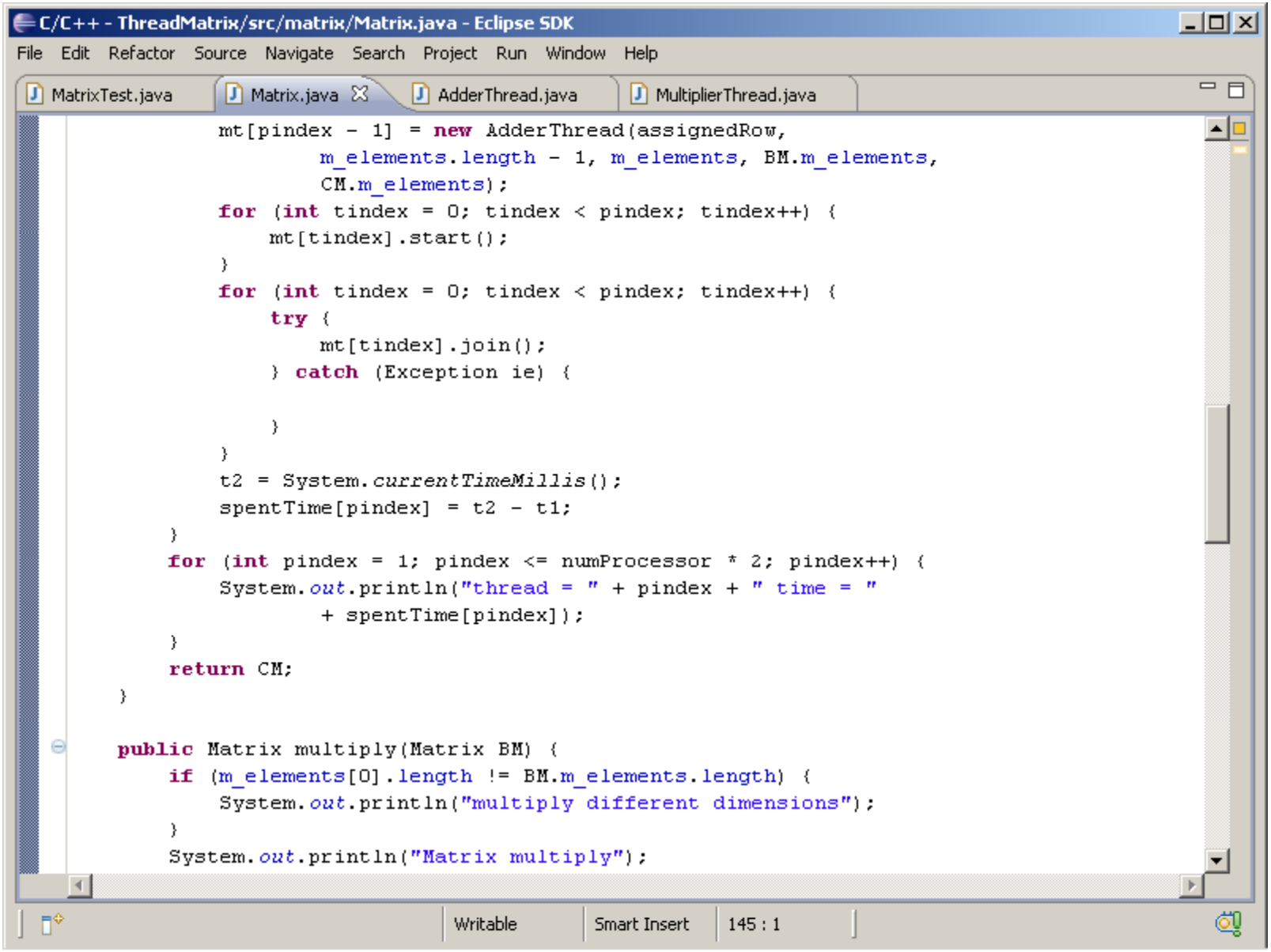

C/C++ - ThreadMatrix/src/matrix/MatrixTest.java - Eclipse SDK
File Edit Refactor Source Navigate Search Project Run Window Help
MatrixTest.java
Matrix.java
AdderThread.java
MultiplierThread.java
package matrix;
public class MatrixTest {
/**
* @param args
*/
public static void main(String[] args) {
// TODO Auto-generated method stub
System.out.println("Number of Processors = " +
Runtime.getRuntime().availableProcessors());
int row = 1000;
if (args.length > 0) {
row = Integer.parseInt(args[0]);
}
if (row < 2) {
row = 2;
}
System.out.println("Number of Rows = " + row);
Matrix ma = new Matrix(row);
Matrix mb = new Matrix(row);
Matrix mc = ma.add(mb);
Matrix md = ma.multiply(mb);
// ma.print();
// mb.print();
// mc.print();
}
}
Writable
Smart Insert
1 : 1

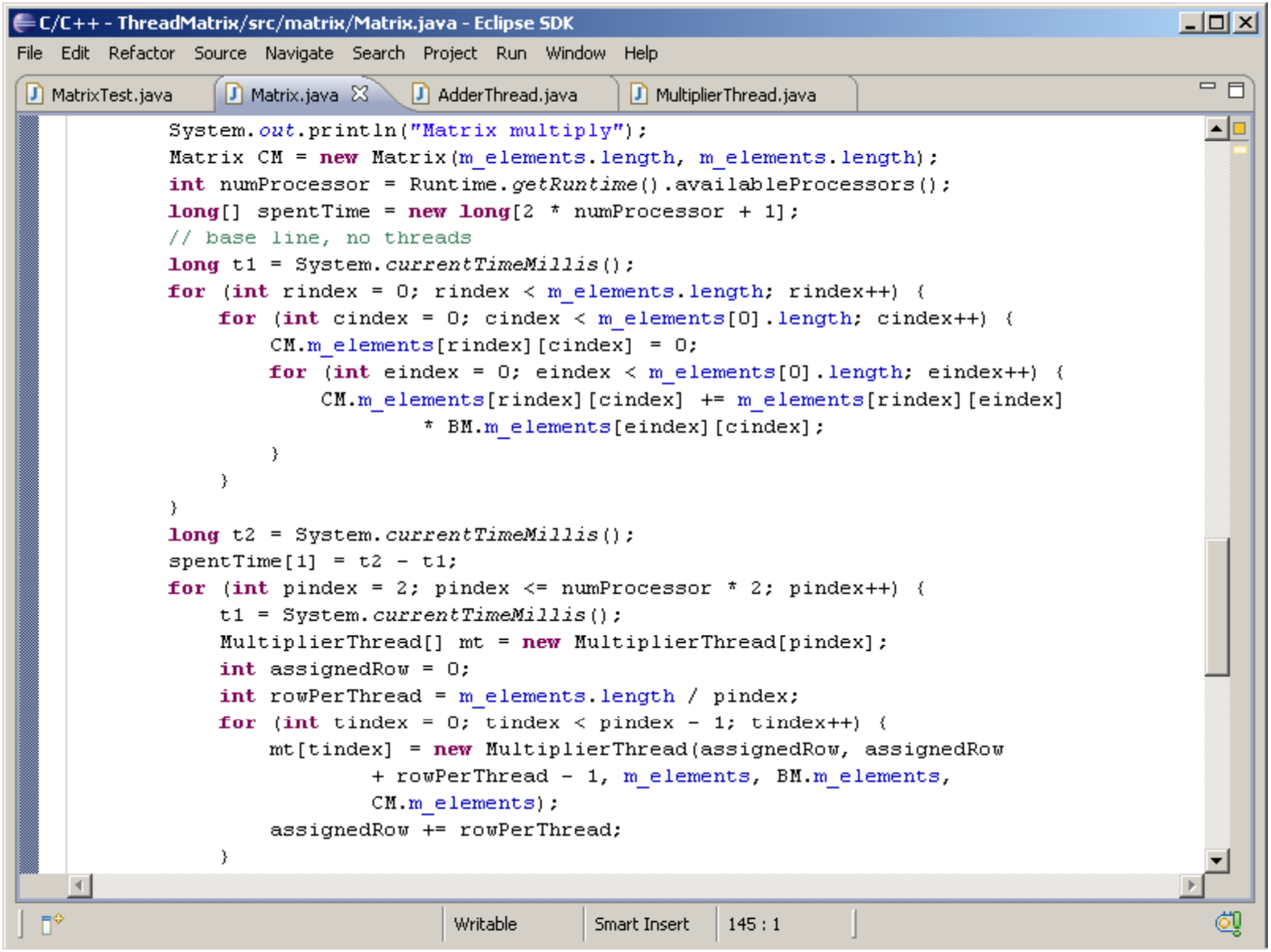

C/C++ - ThreadMatrix/src/matrix/AdderThread.java - Eclipse SDK
File Edit Refactor Source Navigate Search Project Run Window Help
MatrixTest.java
Matrix.java
AdderThread.java
MultiplierThread.java
public class AdderThread extends Thread {
private int mt_r1;
private int mt_r2;
double[][] mt_Aelements;
double[][] mt_Belements;
double[][] mt_Celements;
public AdderThread(int r1, int r2, double[][] MAelements,
double[][] MBelements, double[][] MCelements) {
// System.out.println("r1 r2 = " + r1 + " " + r2);
mt_r1 = r1;
mt_r2 = r2;
mt_Aelements = MAelements;
mt_Belements = MBelements;
mt_Celements = MCelements;
}
public void run() {
int numCol = mt_Aelements[0].length;
for (int rindex = mt_r1; rindex <= mt_r2; rindex++) {
for (int cindex = 0; cindex < numCol; cindex++) {
mt_Celements[rindex][cindex] = mt_Aelements[rindex][cindex]
+ mt_Belements[rindex][cindex];
}
}
}
}
Writable
Smart Insert
1 : 1

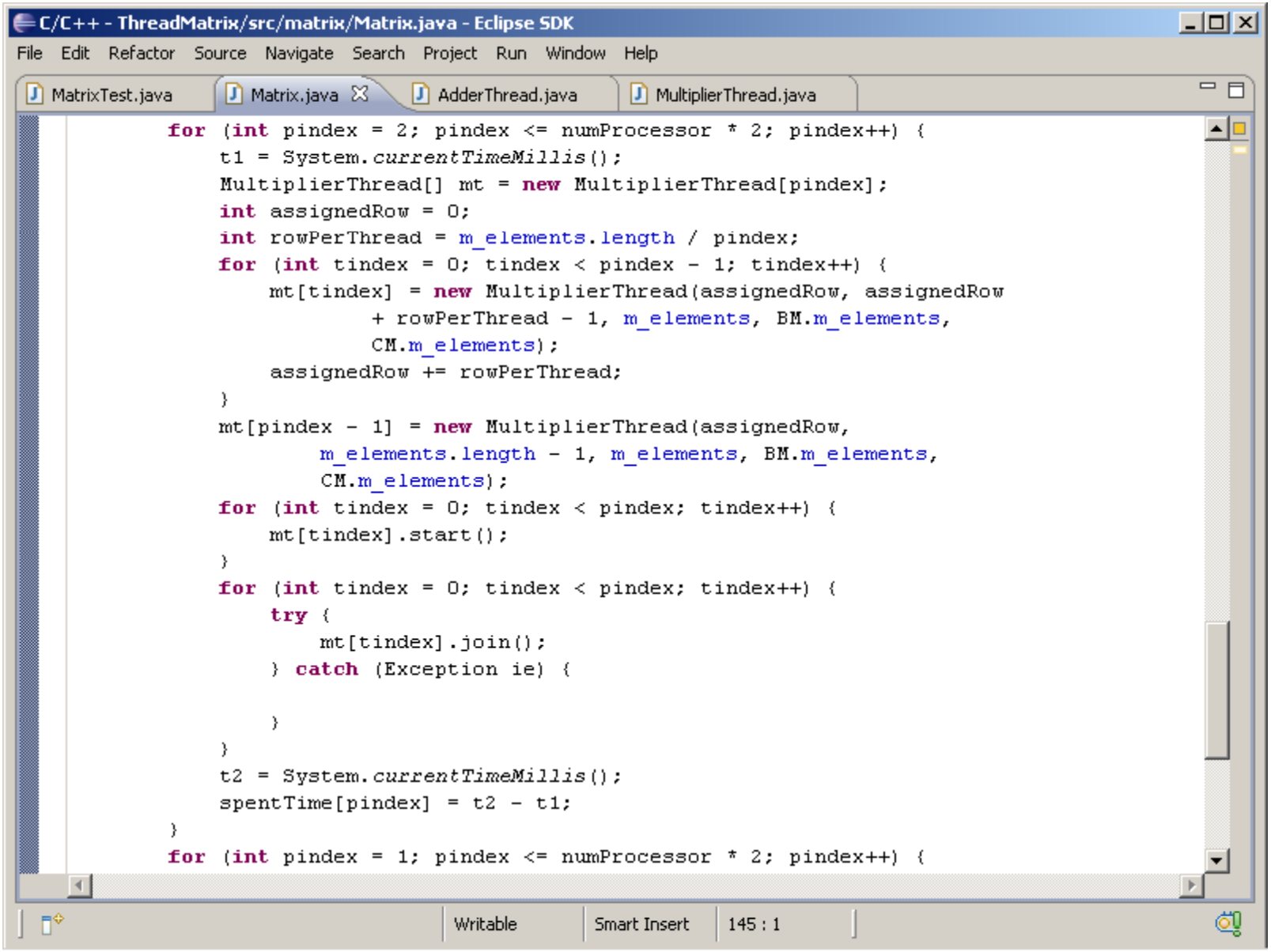

C/C++ - ThreadMatrix/src/matrix/MultiplierThread.java - Eclipse SDK
File Edit Refactor Source Navigate Search Project Run Window Help
MatrixTest.java
Matrix.java
AdderThread.java
MultiplierThread.java
public class MultiplierThread extends Thread{
private int mt_r1;
private int mt_r2;
double[][] mt_Aelements;
double[][] mt_Belements;
double[][] mt_Celements;
public MultiplierThread(int r1, int r2, double[][] MAelements,
double[][] MBelements, double[][] MCelements) {
// System.out.println("r1 r2 = " + r1 + " " + r2);
mt_r1 = r1;
mt_r2 = r2;
mt_Aelements = MAelements;
mt_Belements = MBelements;
mt_Celements = MCelements;
}
public void run() {
int numCol = mt_Aelements[0].length;
for (int rindex = mt_r1; rindex <= mt_r2; rindex++) {
for (int cindex = 0; cindex < numCol; cindex++) {
mt_Celements[rindex][cindex] = 0;
for (int eindex = 0; eindex < numCol; eindex++) {
mt_Celements[rindex][cindex] += mt_Aelements[rindex][eindex]
* mt_Belements[eindex][cindex];
}
}
}
Writable
Smart Insert
29 : 1

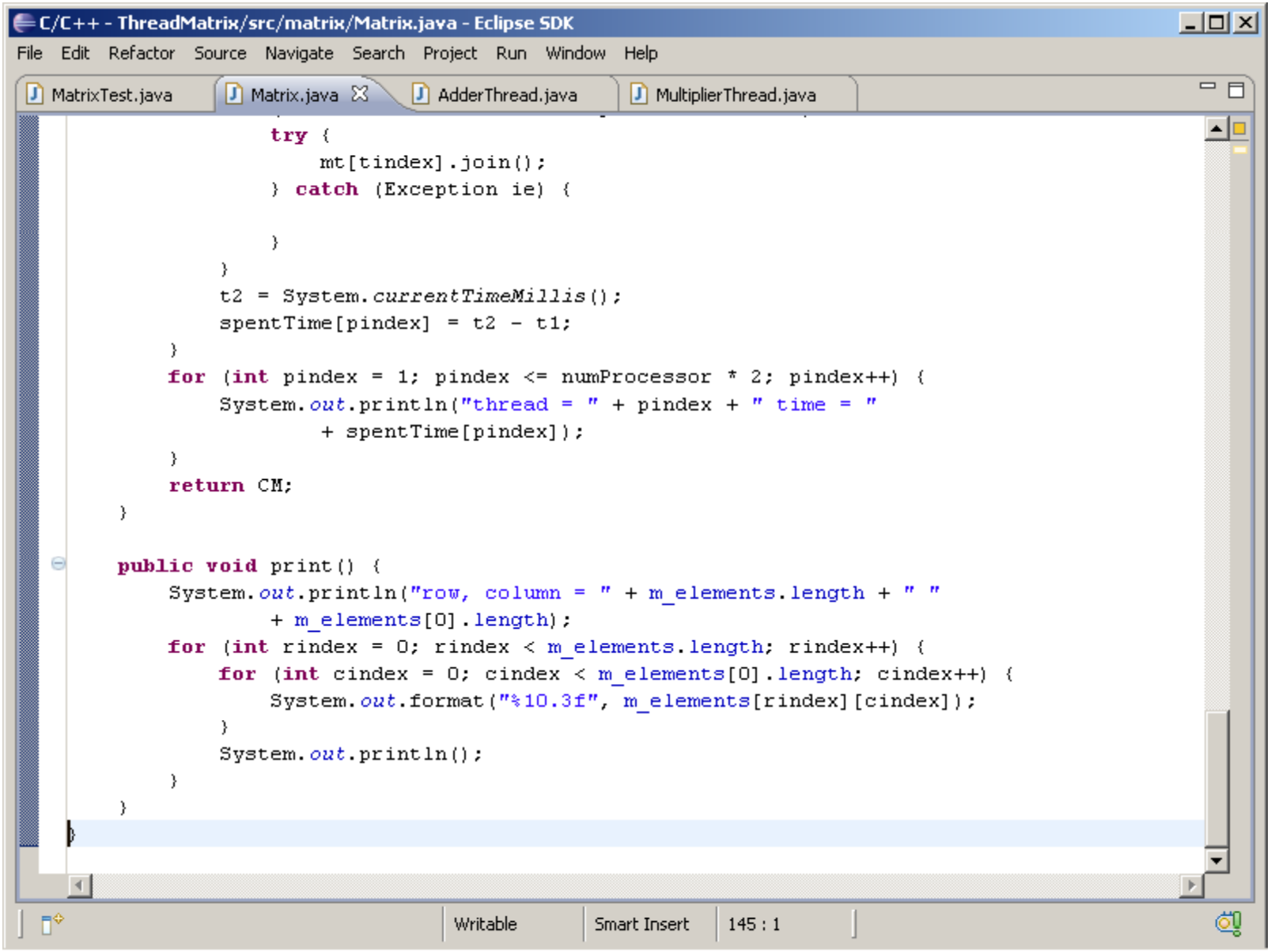

C/C++ - ThreadMatrix/src/matrix/Matrix.java - Eclipse SDK
File Edit Refactor Source Navigate Search Project Run Window Help
MatrixTest.java
Matrix.java
AdderThread.java
MultiplierThread.java
package matrix;
import java.io.FileOutputStream;
public class Matrix {
double[][] m_elements;
public Matrix(int row, int col) {
m_elements = new double[row][col];
}
public Matrix(int row) {
m_elements = new double[row][row];
for (int rindex = 0; rindex < row; rindex++) {
for (int cindex = 0; cindex < row; cindex++) {
if (rindex == cindex) {
m_elements[rindex][cindex] = 1;
} else {
m_elements[rindex][cindex] = 0;
}
}
}
}
public Matrix add(Matrix BM) {
if ((m_elements.length != BM.m_elements.length)
|| (m_elements[0].length != BM.m_elements[0].length)) {
System.out.println("add different dimensions");
}
System.out.println("Matrix add");
Matrix CM = new Matrix(m_elements.length, m_elements.length);
int numProcessor = Runtime.getRuntime().availableProcessors();
long[] spentTime = new long[2 * numProcessor + 1];
Writable
Smart Insert
145 : 1

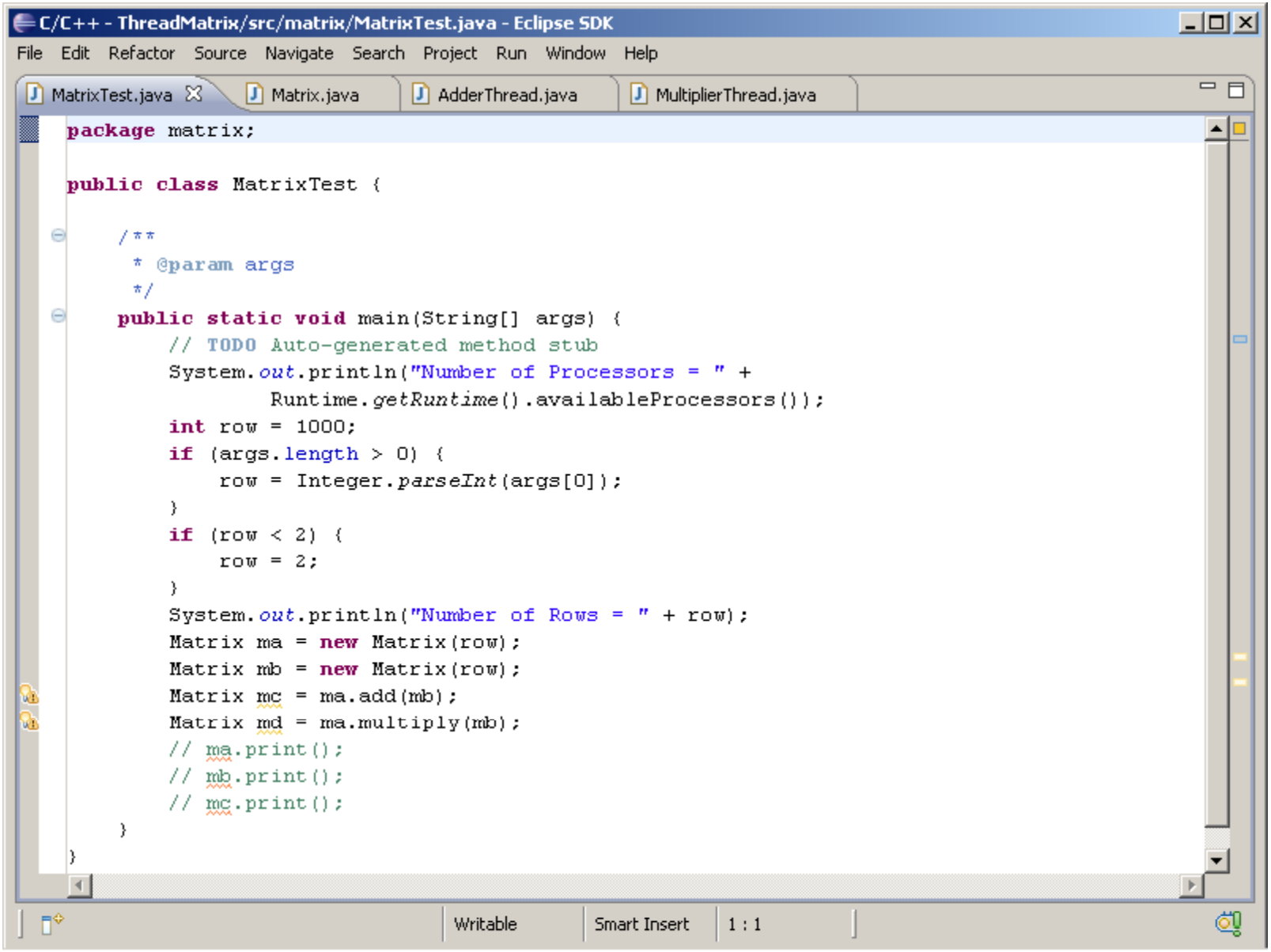

C/C++ - ThreadMatrix/src/matrix/Matrix.java - Eclipse SDK
File Edit Refactor Source Navigate Search Project Run Window Help
MatrixTest.java
Matrix.java
AdderThread.java
MultiplierThread.java
long[] spentTime = new long[2 * numProcessor + 1];
// base line, no threads
long t1 = System.currentTimeMillis();
for (int rindex = 0; rindex < m_elements.length; rindex++) {
for (int cindex = 0; cindex < m_elements[0].length; cindex++) {
CM.m_elements[rindex][cindex] = m_elements[rindex][cindex]
+ BM.m_elements[rindex][cindex];
}
}
long t2 = System.currentTimeMillis();
spentTime[1] = t2 - t1;
for (int pindex = 2; pindex <= numProcessor * 2; pindex++) {
// System.out.println("number of threads = " + pindex);
t1 = System.currentTimeMillis();
AdderThread[] mt = new AdderThread[pindex];
int assignedRow = 0;
int rowPerThread = m_elements.length / pindex;
for (int tindex = 0; tindex < pindex - 1; tindex++) {
mt[tindex] = new AdderThread(assignedRow, assignedRow
+ rowPerThread - 1, m_elements, BM.m_elements,
CM.m_elements);
assignedRow += rowPerThread;
}
// assign all remaining rows, this is needed because
// the number of rows may not be a multiple of the number
// of threads
mt[pindex - 1] = new AdderThread(assignedRow,
m_elements.length - 1, m_elements, BM.m_elements,
Writable
Smart Insert
145 : 1

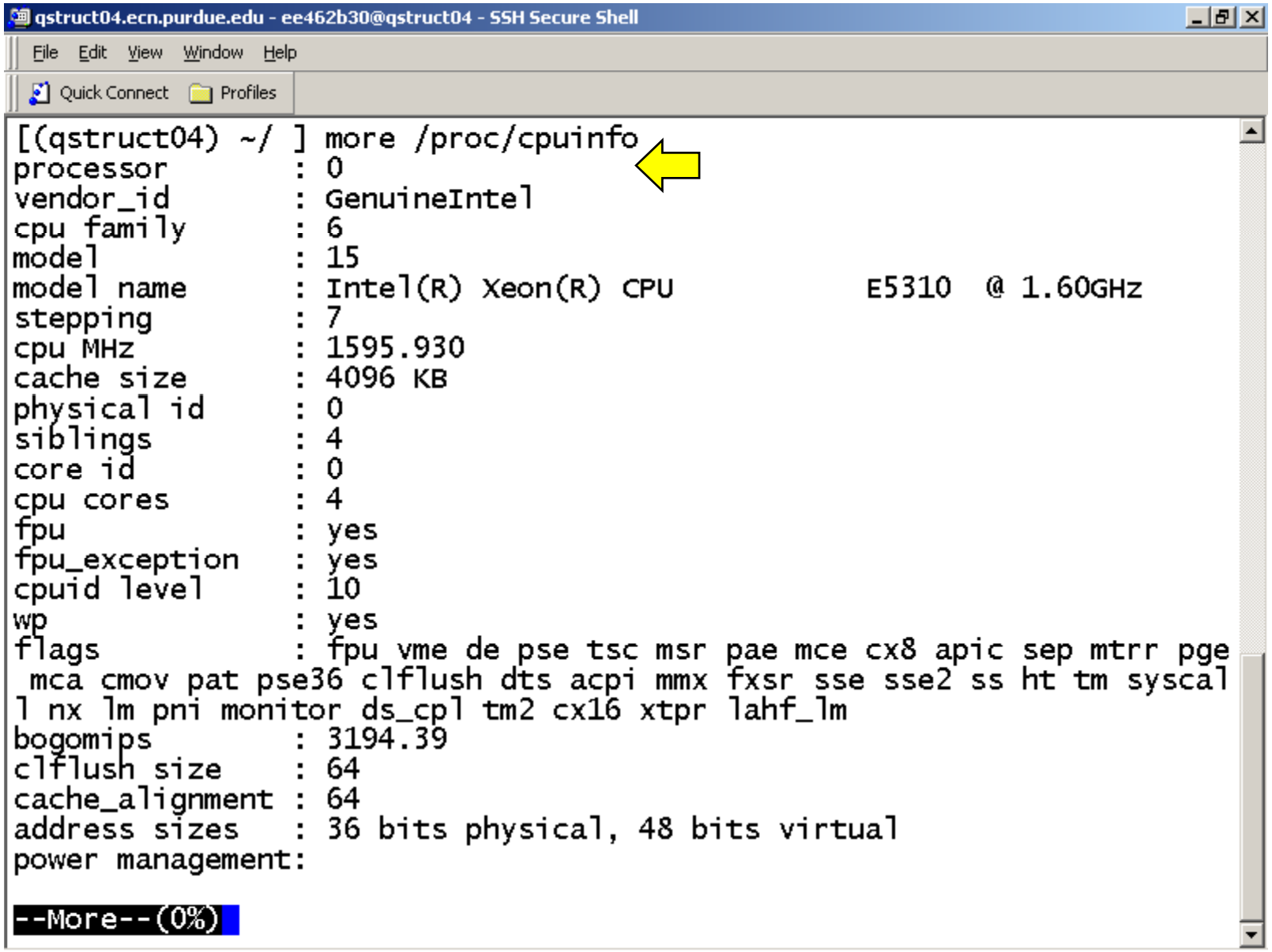

C/C++ - ThreadMatrix/src/matrix/Matrix.java - Eclipse SDK
File Edit Refactor Source Navigate Search Project Run Window Help
MatrixTest.java
Matrix.java
AdderThread.java
MultiplierThread.java
mt[pindex - 1] = new AdderThread(assignedRow,
m_elements.length - 1, m_elements, BM.m_elements,
CM.m_elements);
for (int tindex = 0; tindex < pindex; tindex++) {
mt[tindex].start();
}
for (int tindex = 0; tindex < pindex; tindex++) {
try {
mt[tindex].join();
} catch (Exception ie) {
}
}
t2 = System.currentTimeMillis();
spentTime[pindex] = t2 - t1;
}
for (int pindex = 1; pindex <= numProcessor * 2; pindex++) {
System.out.println("thread = " + pindex + " time = "
+ spentTime[pindex]);
}
return CM;
}
public Matrix multiply(Matrix BM) {
if (m_elements[0].length != BM.m_elements.length) {
System.out.println("multiply different dimensions");
}
System.out.println("Matrix multiply");
Writable
Smart Insert
145 : 1

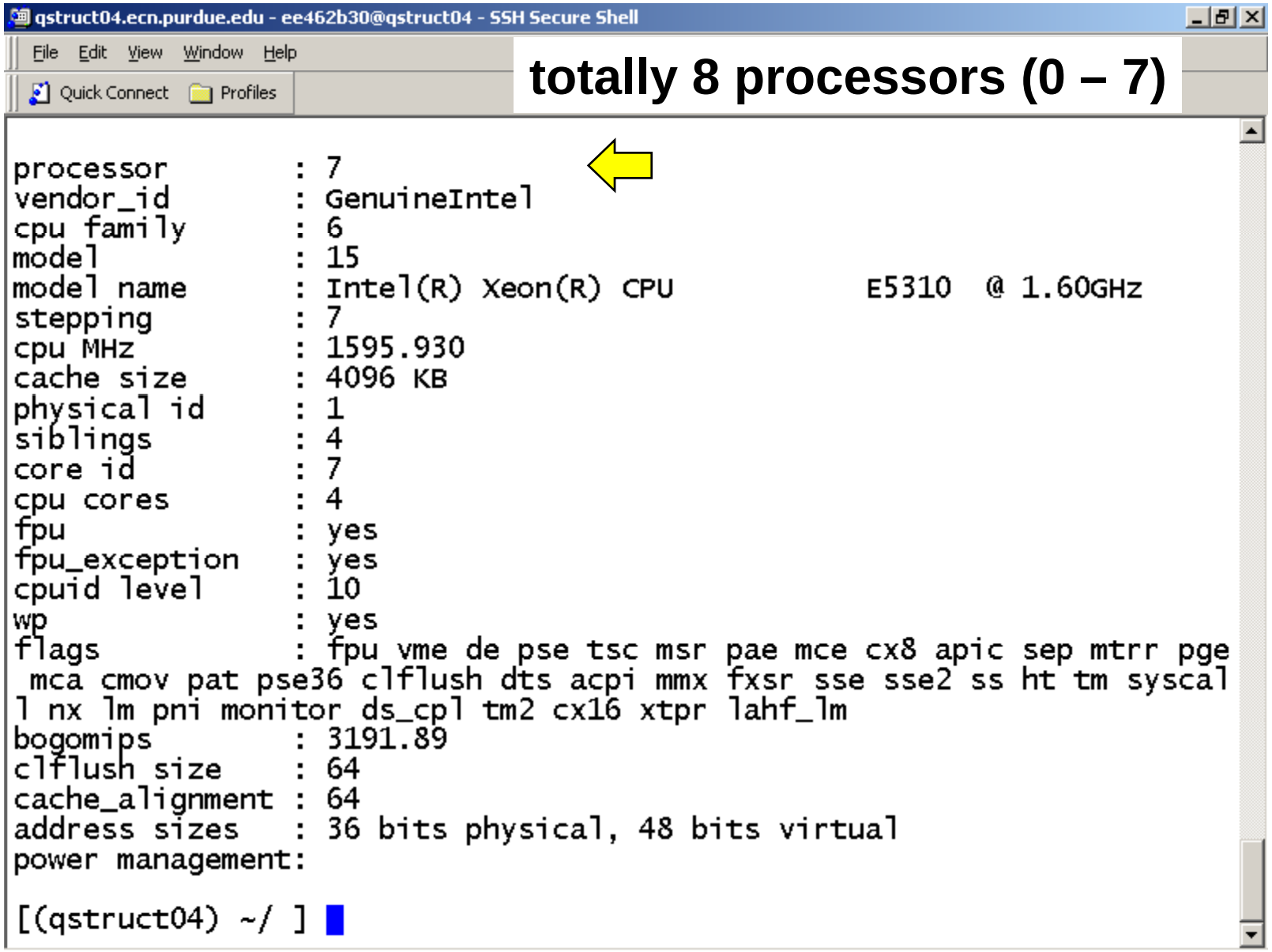

C/C++ - ThreadMatrix/src/matrix/Matrix.java - Eclipse SDK
File Edit Refactor Source Navigate Search Project Run Window Help
MatrixTest.java
Matrix.java
AdderThread.java
MultiplierThread.java
System.out.println("Matrix multiply");
Matrix CM = new Matrix(m_elements.length, m_elements.length);
int numProcessor = Runtime.getRuntime().availableProcessors();
long[] spentTime = new long[2 * numProcessor + 1];
// base line, no threads
long t1 = System.currentTimeMillis();
for (int rindex = 0; rindex < m_elements.length; rindex++) {
for (int cindex = 0; cindex < m_elements[0].length; cindex++) {
CM.m_elements[rindex][cindex] = 0;
for (int eindex = 0; eindex < m_elements[0].length; eindex++) {
CM.m_elements[rindex][cindex] += m_elements[rindex][eindex]
* BM.m_elements[eindex][cindex];
}
}
}
long t2 = System.currentTimeMillis();
spentTime[1] = t2 - t1;
for (int pindex = 2; pindex <= numProcessor * 2; pindex++) {
t1 = System.currentTimeMillis();
MultiplierThread[] mt = new MultiplierThread[pindex];
int assignedRow = 0;
int rowPerThread = m_elements.length / pindex;
for (int tindex = 0; tindex < pindex - 1; tindex++) {
mt[tindex] = new MultiplierThread(assignedRow, assignedRow
+ rowPerThread - 1, m_elements, BM.m_elements,
CM.m_elements);
assignedRow += rowPerThread;
}
Writable
Smart Insert
145 : 1

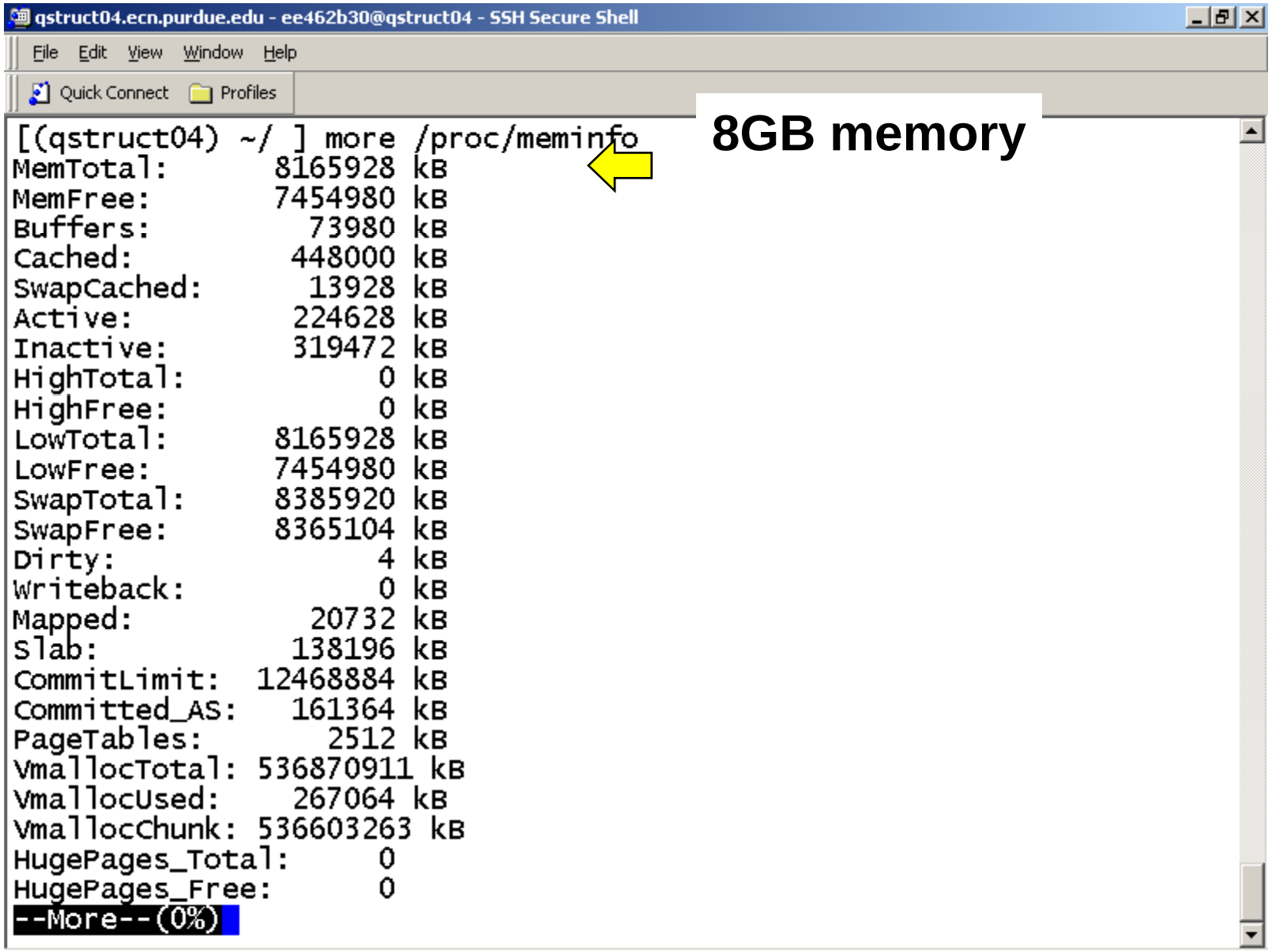

C/C++ - ThreadMatrix/src/matrix/Matrix.java - Eclipse SDK
File Edit Refactor Source Navigate Search Project Run Window Help
MatrixTest.java
Matrix.java
AdderThread.java
MultiplierThread.java
for (int pindex = 2; pindex <= numProcessor * 2; pindex++) {
t1 = System.currentTimeMillis();
MultiplierThread[] mt = new MultiplierThread[pindex];
int assignedRow = 0;
int rowPerThread = m_elements.length / pindex;
for (int tindex = 0; tindex < pindex - 1; tindex++) {
mt[tindex] = new MultiplierThread(assignedRow, assignedRow
+ rowPerThread - 1, m_elements, BM.m_elements,
CM.m_elements);
assignedRow += rowPerThread;
}
mt[pindex - 1] = new MultiplierThread(assignedRow,
m_elements.length - 1, m_elements, BM.m_elements,
CM.m_elements);
for (int tindex = 0; tindex < pindex; tindex++) {
mt[tindex].start();
}
for (int tindex = 0; tindex < pindex; tindex++) {
try {
mt[tindex].join();
} catch (Exception ie) {
}
}
t2 = System.currentTimeMillis();
spentTime[pindex] = t2 - t1;
}
for (int pindex = 1; pindex <= numProcessor * 2; pindex++) {
Writable
Smart Insert
145 : 1

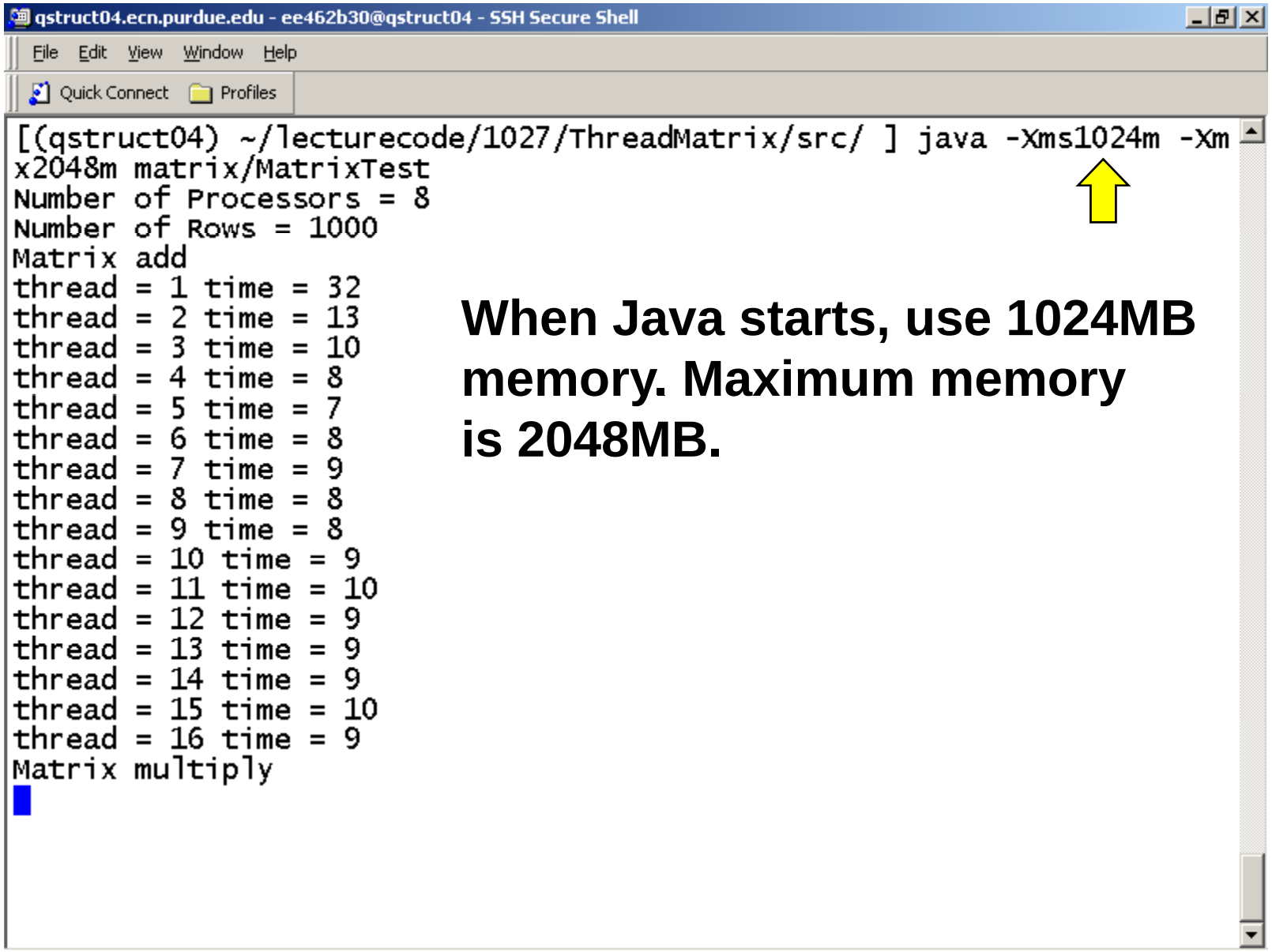

C/C++ - ThreadMatrix/src/matrix/Matrix.java - Eclipse SDK
File Edit Refactor Source Navigate Search Project Run Window Help
MatrixTest.java
Matrix.java
AdderThread.java
MultiplierThread.java
try {
mt[tindex].join();
} catch (Exception ie) {
}
}
t2 = System.currentTimeMillis();
spentTime[pindex] = t2 - t1;
}
for (int pindex = 1; pindex <= numProcessor * 2; pindex++) {
System.out.println("thread = " + pindex + " time = "
+ spentTime[pindex]);
}
return CM;
}
public void print() {
System.out.println("row, column = " + m_elements.length + " "
+ m_elements[0].length);
for (int rindex = 0; rindex < m_elements.length; rindex++) {
for (int cindex = 0; cindex < m_elements[0].length; cindex++) {
System.out.format("%10.3f", m_elements[rindex][cindex]);
}
System.out.println();
}
}
Writable
Smart Insert
145 : 1


C/C++ - ThreadMatrix/src/matrix/MatrixTest.java - Eclipse SDK
File Edit Refactor Source Navigate Search Project Run Window Help
MatrixTest.java
Matrix.java
AdderThread.java
MultiplierThread.java
package matrix;
public class MatrixTest {
/**
* @param args
*/
public static void main(String[] args) {
// TODO Auto-generated method stub
System.out.println("Number of Processors = " +
Runtime.getRuntime().availableProcessors());
int row = 1000;
if (args.length > 0) {
row = Integer.parseInt(args[0]);
}
if (row < 2) {
row = 2;
}
System.out.println("Number of Rows = " + row);
Matrix ma = new Matrix(row);
Matrix mb = new Matrix(row);
Matrix mc = ma.add(mb);
Matrix md = ma.multiply(mb);
// ma.print();
// mb.print();
// mc.print();
}
}
Writable
Smart Insert
1 : 1

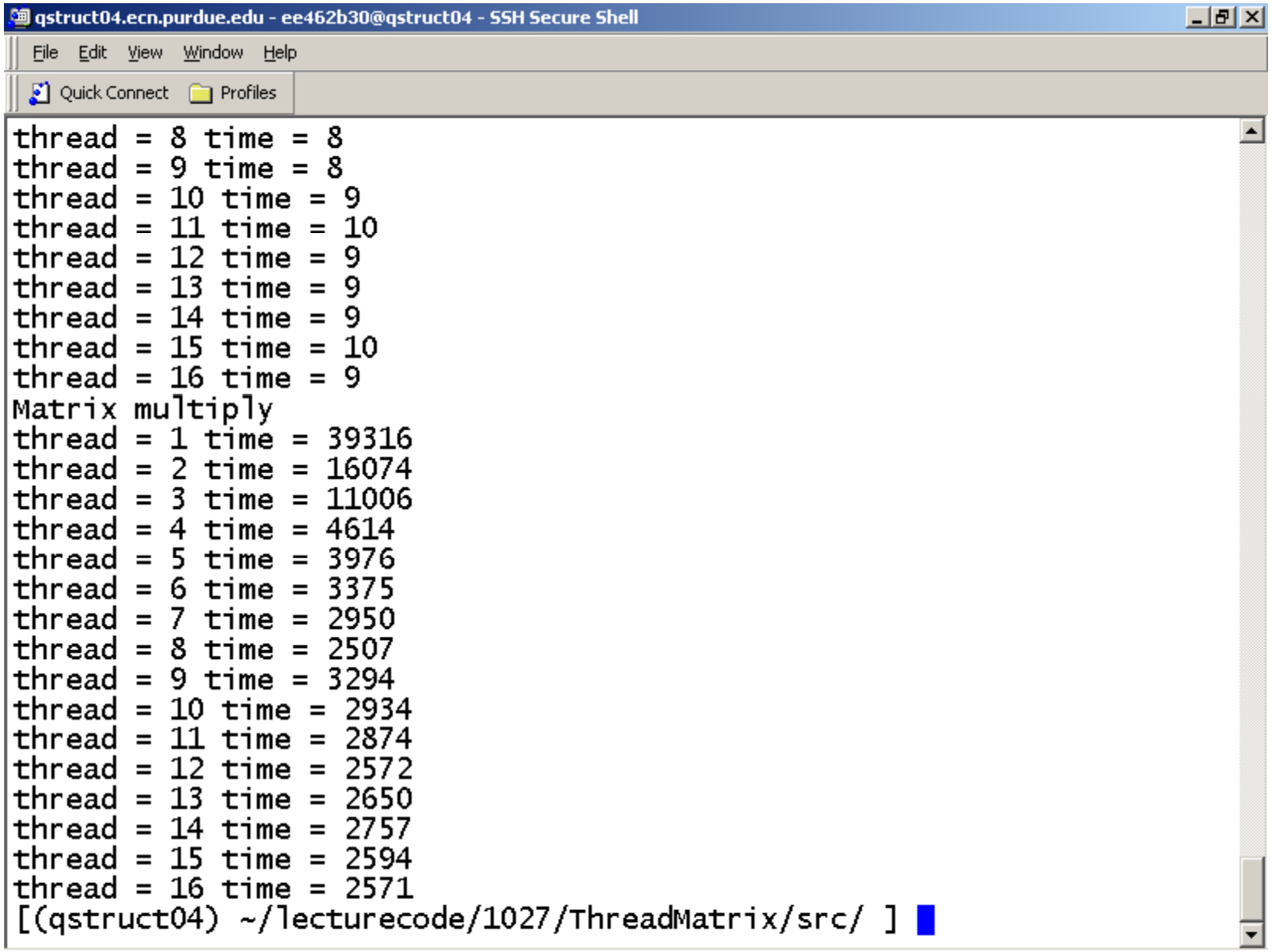

qstruct04.ecn.purdue.edu - ee462b30@qstruct04 - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
[(qstruct04) ~/ ] more /proc/cpuinfo
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 15
model name : Intel(R) Xeon(R) CPU E5310 @ 1.60GHz
stepping : 7
cpu MHz : 1595.930
cache size : 4096 KB
physical id : 0
siblings : 4
core id : 0
cpu cores : 4
fpu : yes
fpu_exception : yes
cpuid level : 10
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm syscal
l nx lm pni monitor ds_cpl tm2 cx16 xtpr lahf_lm
bogomips : 3194.39
clflush size : 64
cache_alignment : 64
address sizes : 36 bits physical, 48 bits virtual
power management:
--More--(0%)


qstruct04.ecn.purdue.edu - ee462b30@qstruct04 - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
totally 8 processors (0 – 7)
processor : 7
vendor_id : GenuineIntel
cpu family : 6
model : 15
model name : Intel(R) Xeon(R) CPU E5310 @ 1.60GHz
stepping : 7
cpu MHz : 1595.930
cache size : 4096 KB
physical id : 1
siblings : 4
core id : 7
cpu cores : 4
fpu : yes
fpu_exception : yes
cpuid level : 10
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm syscal
l nx lm pni monitor ds_cpl tm2 cx16 xtpr lahf_lm
bogomips : 3191.89
clflush size : 64
cache_alignment : 64
address sizes : 36 bits physical, 48 bits virtual
power management:
[(qstruct04) ~/ ]

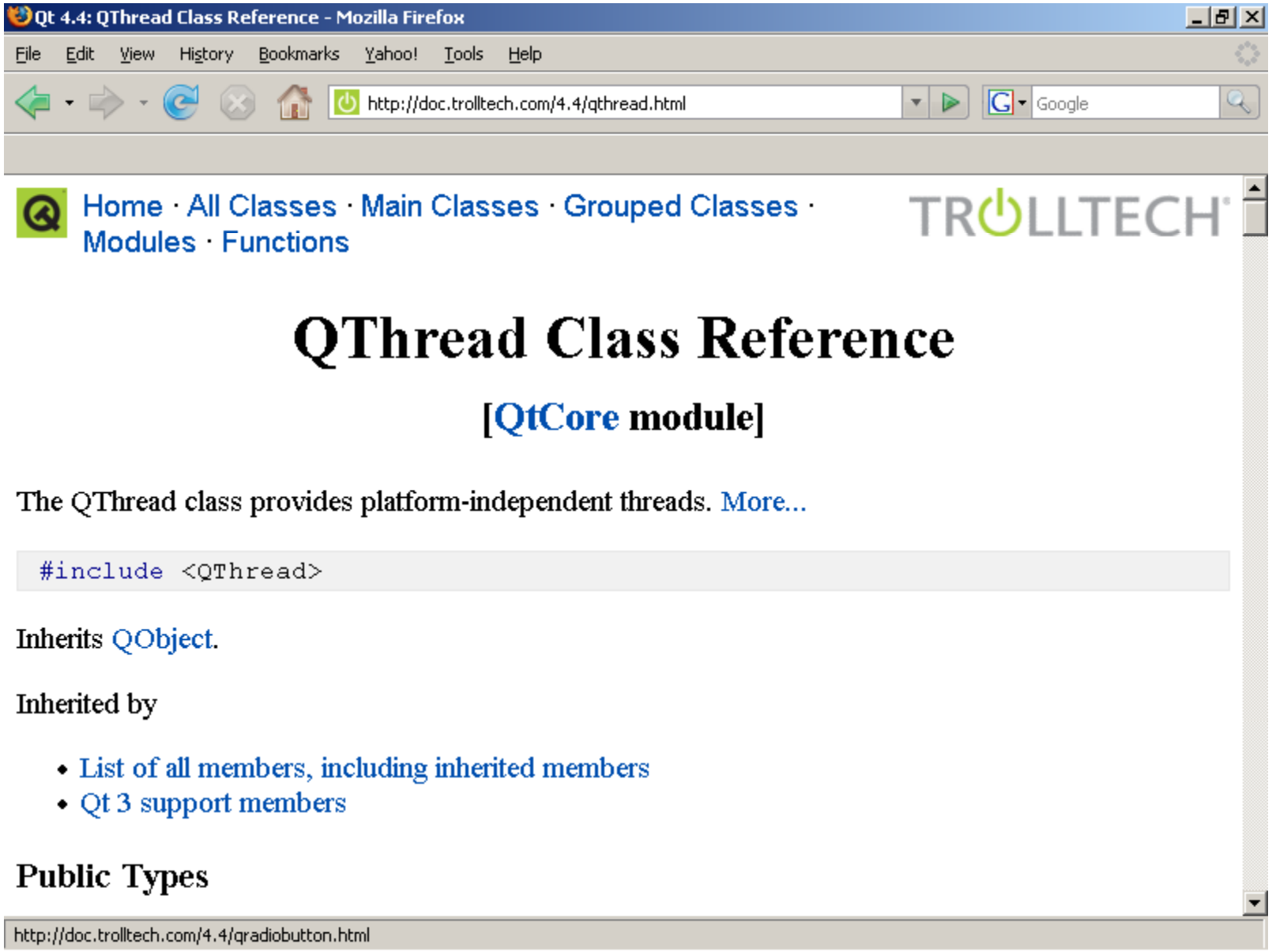

qstruct04.ecn.purdue.edu - ee462b30@qstruct04 - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
[(qstruct04) ~/ ] more /proc/meminfo
MemTotal: 8165928 kB
MemFree: 7454980 kB
Buffers: 73980 kB
Cached: 448000 kB
SwapCached: 13928 kB
Active: 224628 kB
Inactive: 319472 kB
HighTotal: 0 kB
HighFree: 0 kB
LowTotal: 8165928 kB
LowFree: 7454980 kB
SwapTotal: 8385920 kB
SwapFree: 8365104 kB
Dirty: 4 kB
Writeback: 0 kB
Mapped: 20732 kB
Slab: 138196 kB
CommitLimit: 12468884 kB
Committed_AS: 161364 kB
PageTables: 2512 kB
VmallocTotal: 536870911 kB
VmallocUsed: 267064 kB
VmallocChunk: 536603263 kB
HugePages_Total: 0
HugePages_Free: 0
--More--(0%)
8GB memory

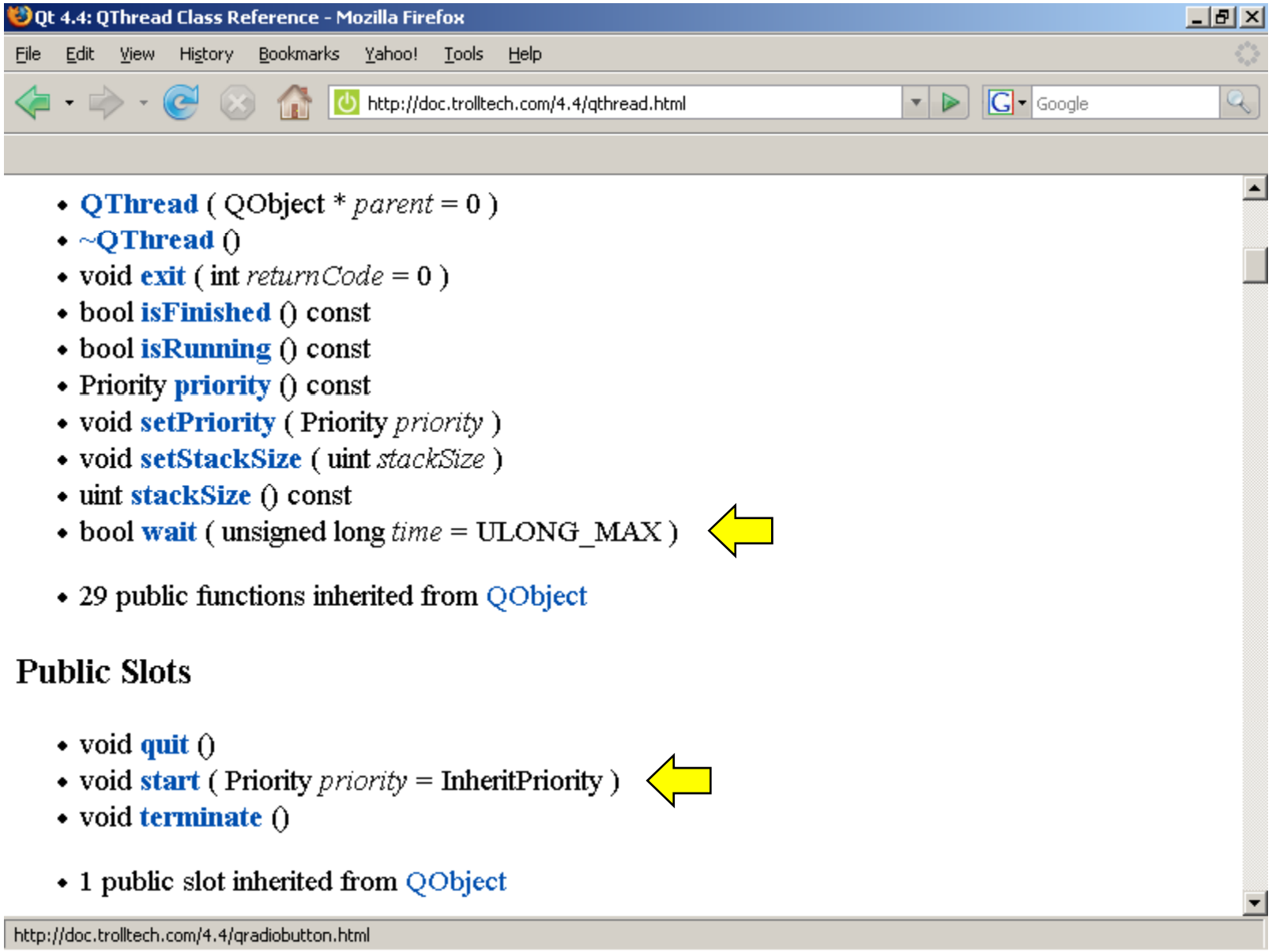


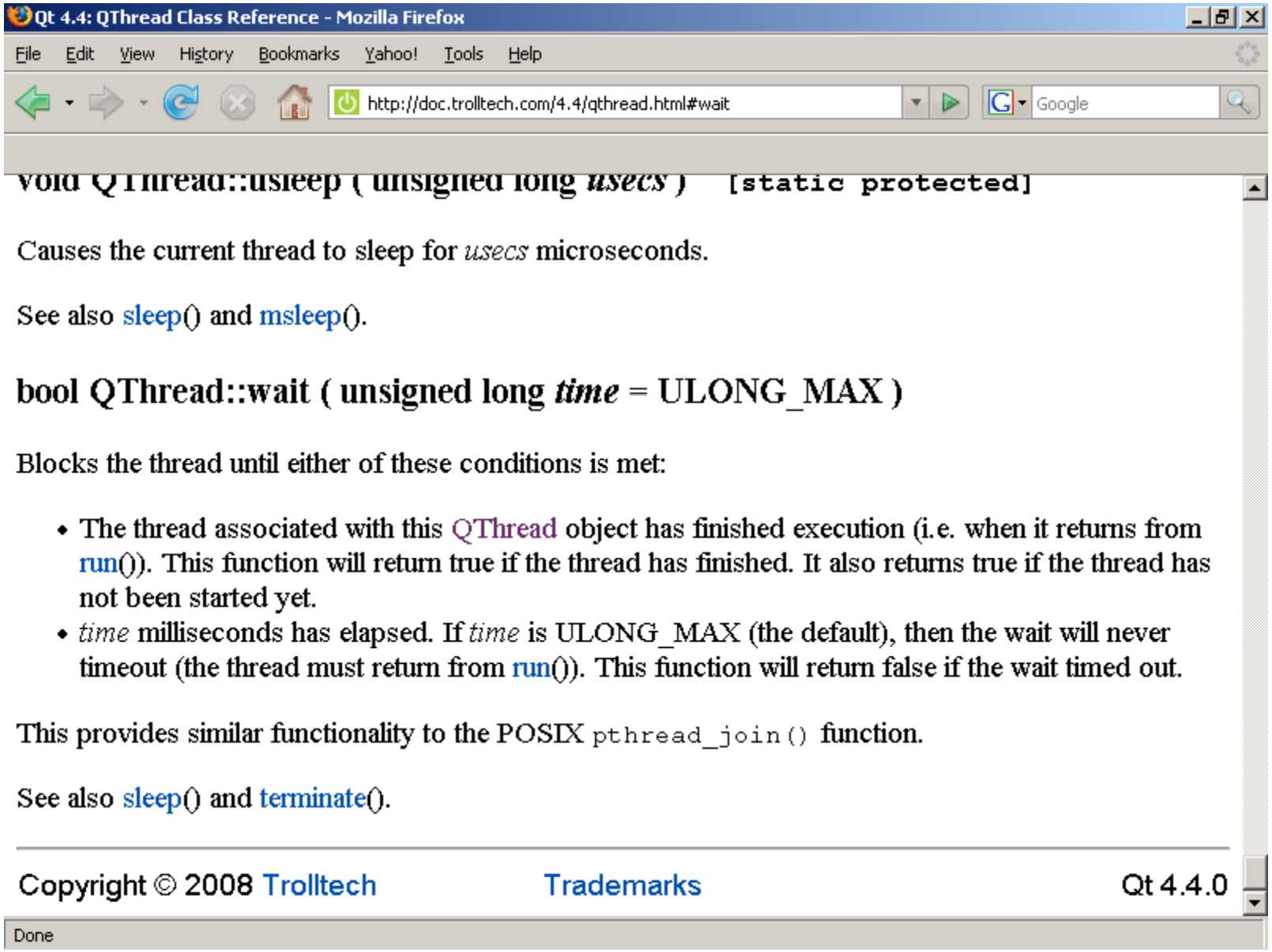

**When Java starts, use 1024MB memory. Maximum memory is 2048MB.**

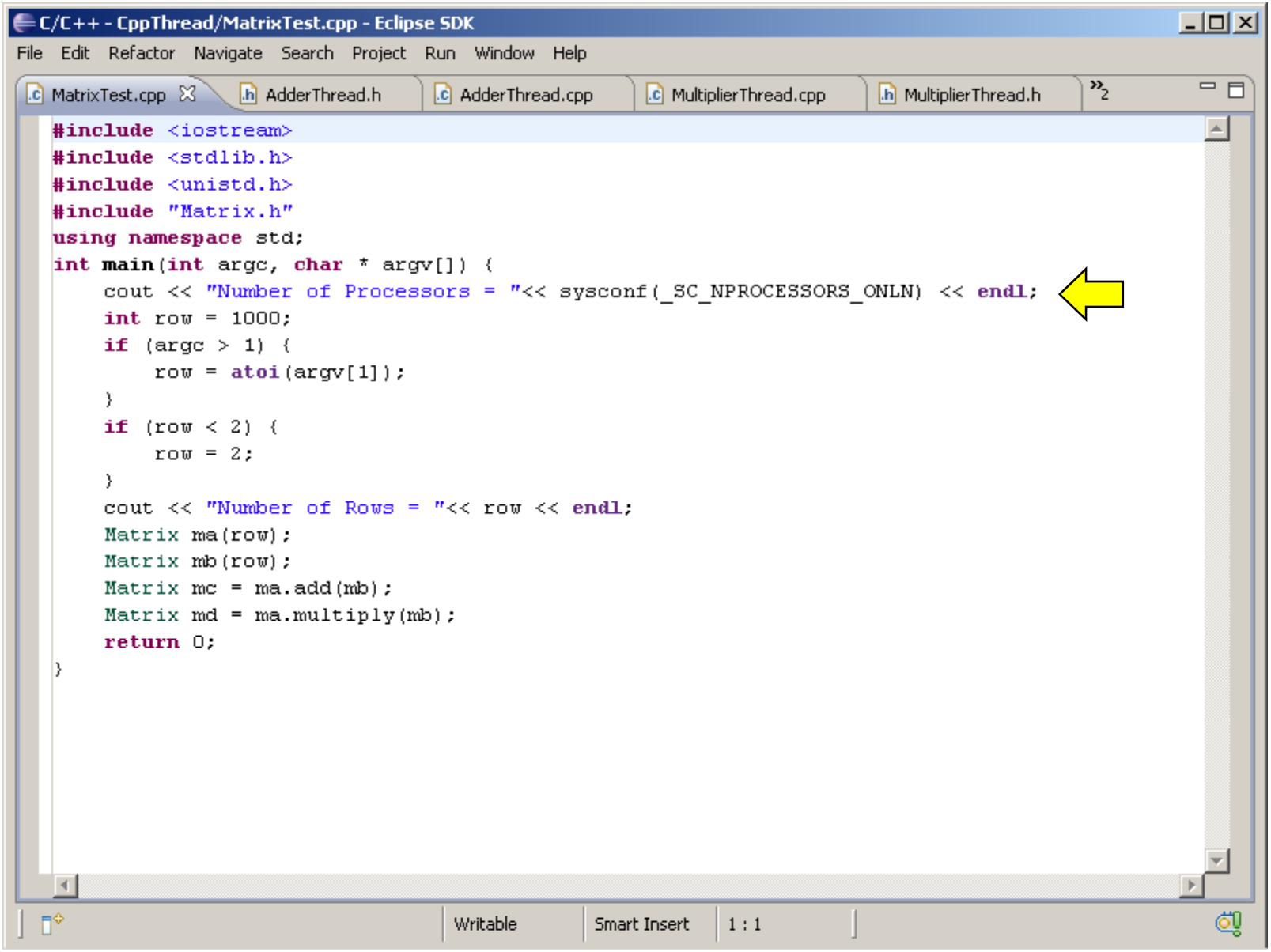

qstruct04.ecn.purdue.edu - ee462b30@qstruct04 - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
thread = 8 time = 8
thread = 9 time = 8
thread = 10 time = 9
thread = 11 time = 10
thread = 12 time = 9
thread = 13 time = 9
thread = 14 time = 9
thread = 15 time = 10
thread = 16 time = 9
Matrix multiply
thread = 1 time = 39316
thread = 2 time = 16074
thread = 3 time = 11006
thread = 4 time = 4614
thread = 5 time = 3976
thread = 6 time = 3375
thread = 7 time = 2950
thread = 8 time = 2507
thread = 9 time = 3294
thread = 10 time = 2934
thread = 11 time = 2874
thread = 12 time = 2572
thread = 13 time = 2650
thread = 14 time = 2757
thread = 15 time = 2594
thread = 16 time = 2571
[(qstruct04) ~/lecturecode/1027/ThreadMatrix/src/ ]

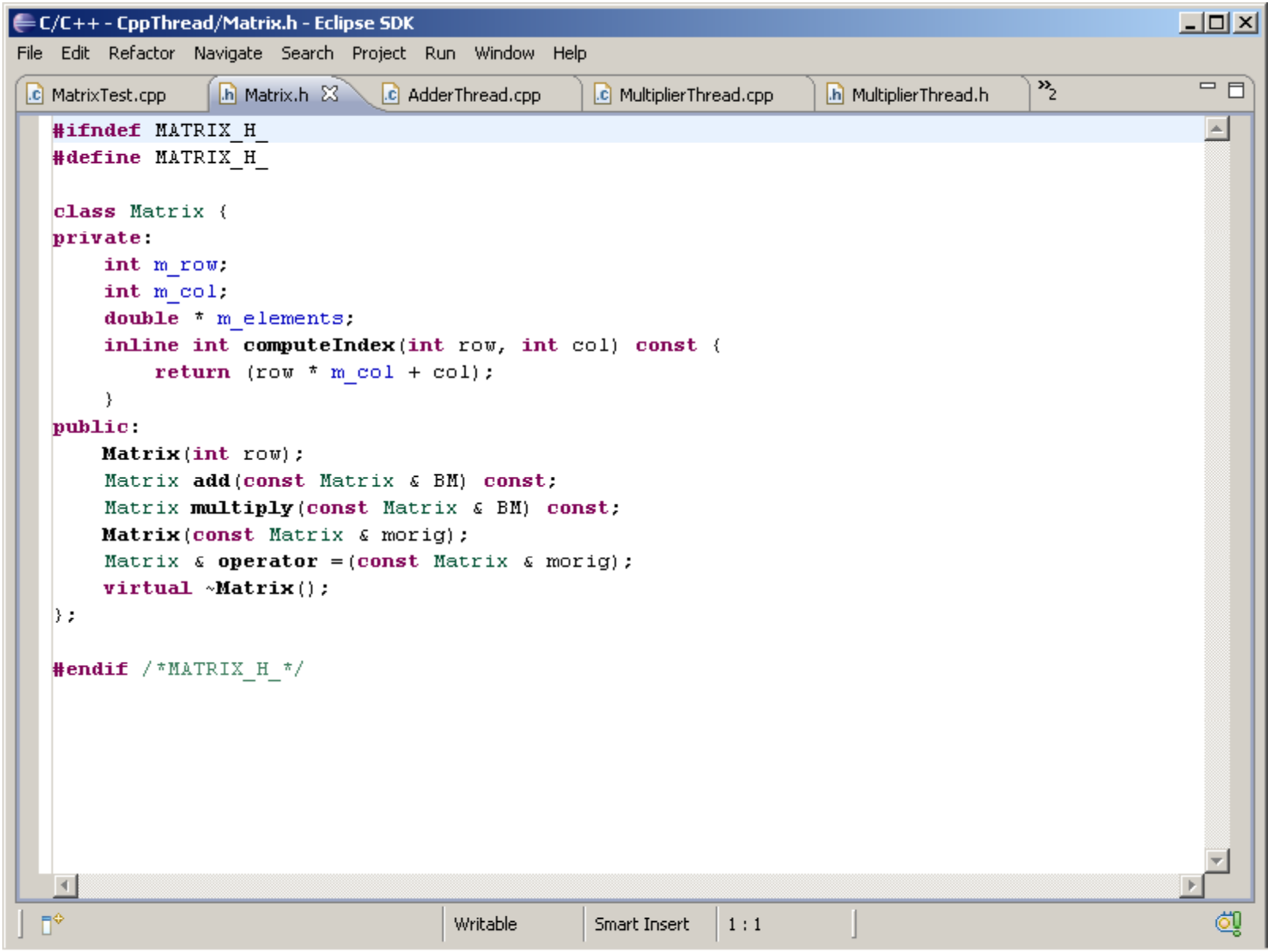
# Qt Thread

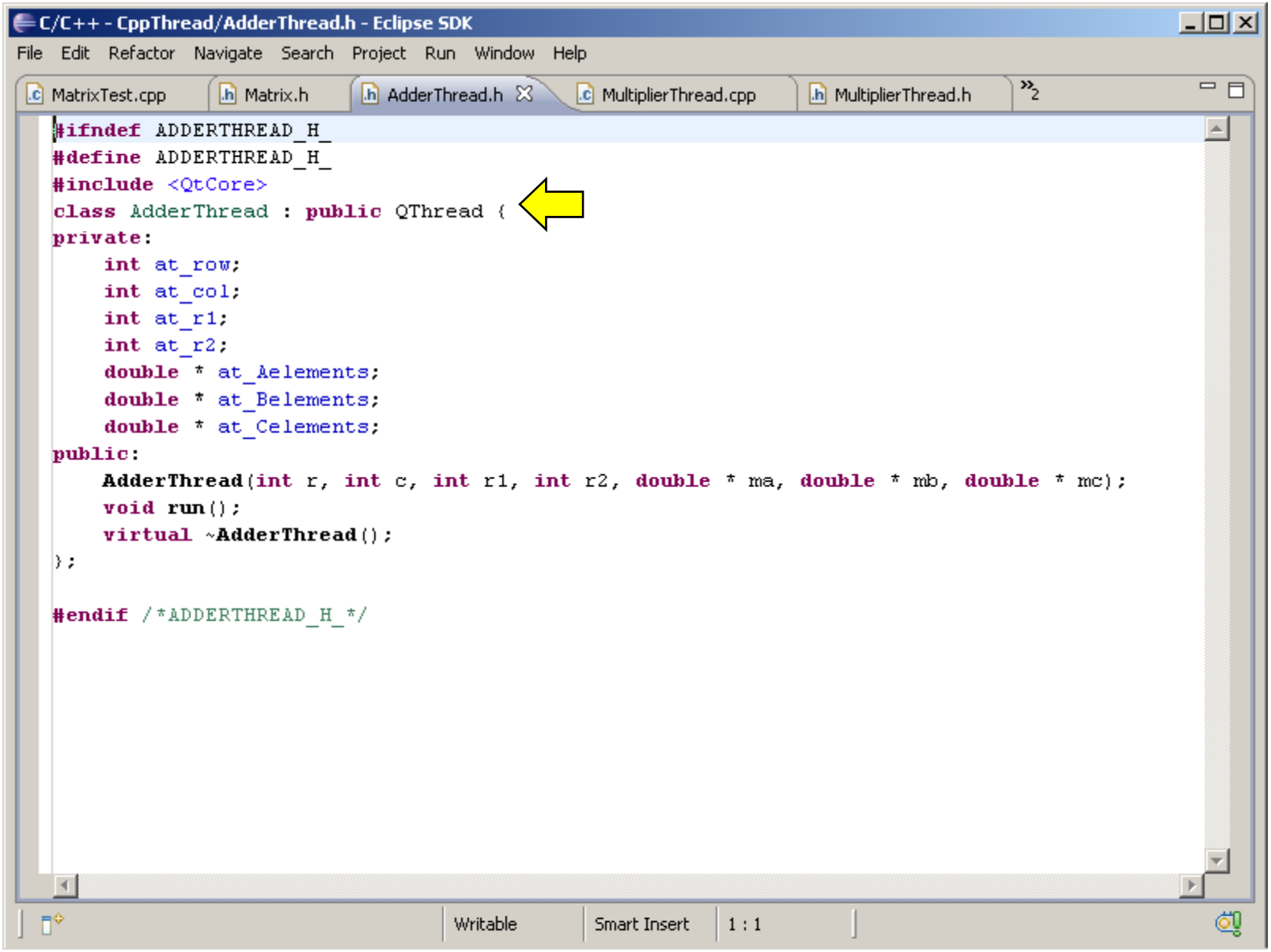

Qt 4.4: QThread Class Reference - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://doc.trolltech.com/4.4/qthread.html
Google
Home · All Classes · Main Classes · Grouped Classes · Modules · Functions
TROLLTECH
QThread Class Reference
[QtCore module]
The QThread class provides platform-independent threads. More...
#include <QThread>
Inherits QObject.
Inherited by
List of all members, including inherited members
Qt 3 support members
Public Types
http://doc.trolltech.com/4.4/qradiobutton.html

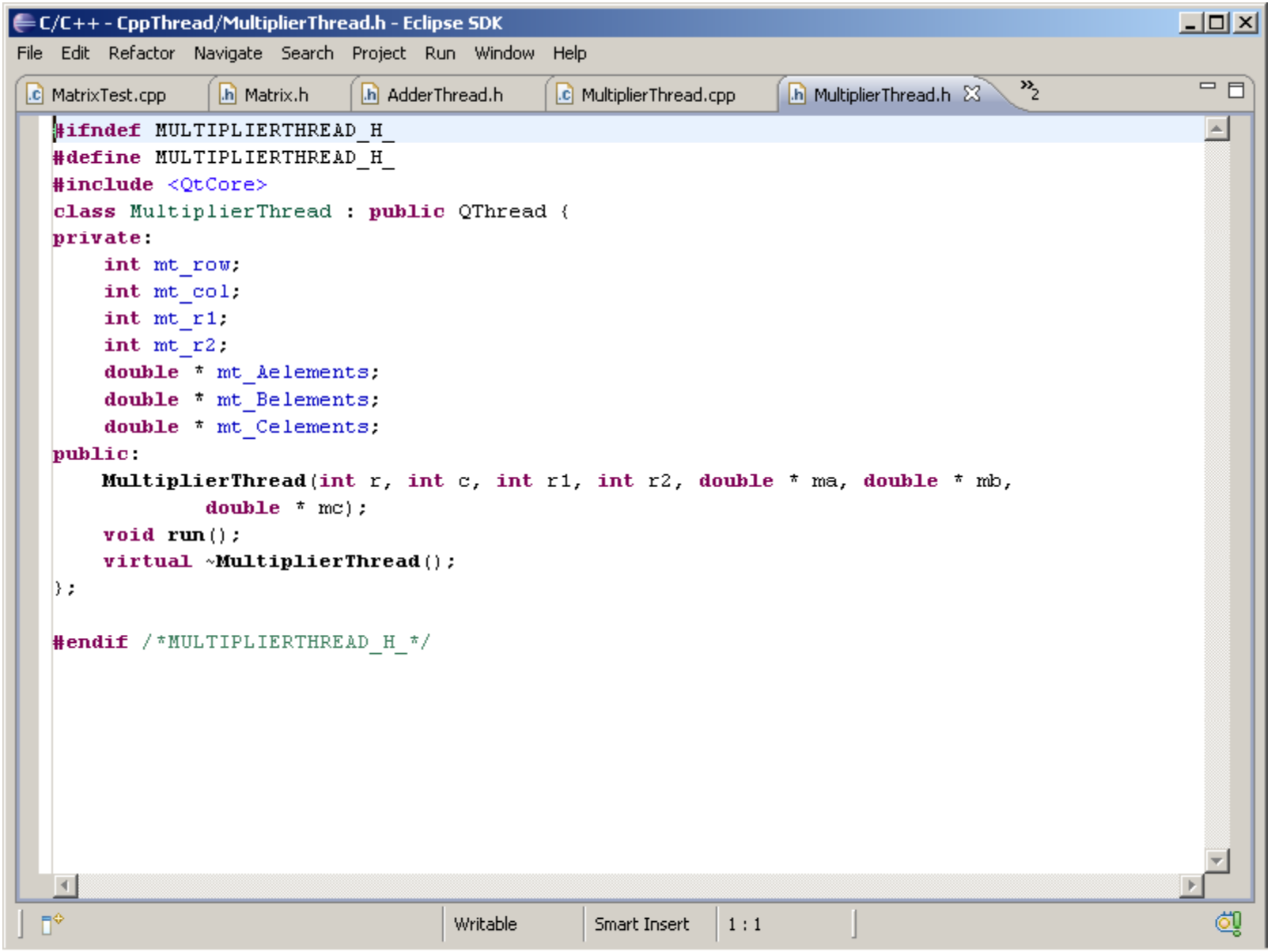

Qt 4.4: QThread Class Reference - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://doc.trolltech.com/4.4/qthread.html
Google
• QThread ( QObject * parent = 0 )
• ~QThread ()
• void exit ( int returnCode = 0 )
• bool isFinished () const
• bool isRunning () const
• Priority priority () const
• void setPriority ( Priority priority )
• void setStackSize ( uint stackSize )
• uint stackSize () const
• bool wait ( unsigned long time = ULONG_MAX )
• 29 public functions inherited from QObject
Public Slots
• void quit ()
• void start ( Priority priority = InheritPriority )
• void terminate ()
• 1 public slot inherited from QObject
http://doc.trolltech.com/4.4/qradiobutton.html

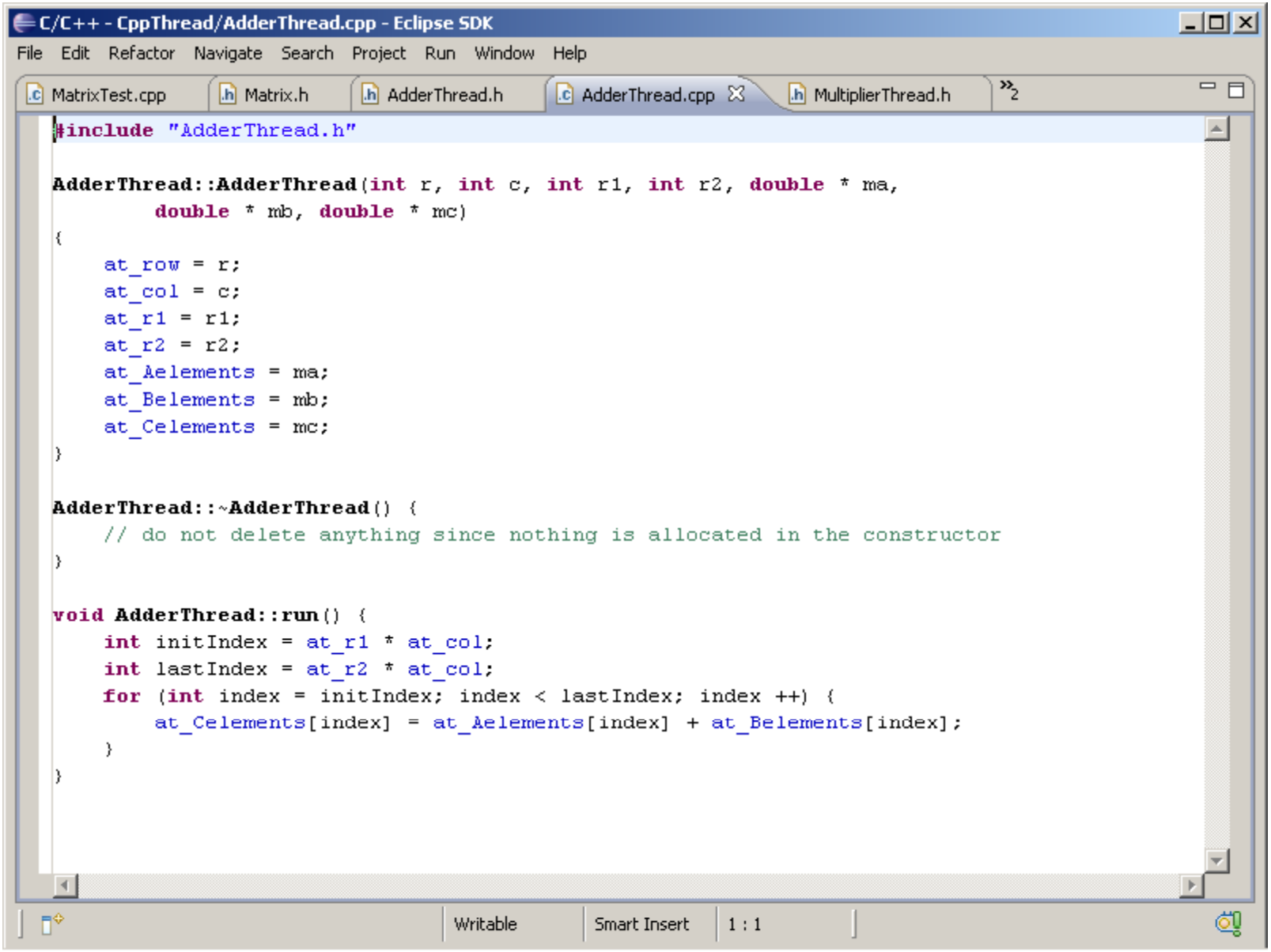

Qt 4.4: QThread Class Reference - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://doc.trolltech.com/4.4/qthread.html#wait
Google
void QThread::usleep ( unsigned long usecs ) [static protected]
Causes the current thread to sleep for usecs microseconds.
See also sleep() and msleep().
bool QThread::wait ( unsigned long time = ULONG_MAX )
Blocks the thread until either of these conditions is met:
The thread associated with this QThread object has finished execution (i.e. when it returns from run()). This function will return true if the thread has finished. It also returns true if the thread has not been started yet.
time milliseconds has elapsed. If time is ULONG_MAX (the default), then the wait will never timeout (the thread must return from run()). This function will return false if the wait timed out.
This provides similar functionality to the POSIX pthread_join() function.
See also sleep() and terminate().
Copyright © 2008 Trolltech
Trademarks
Qt 4.4.0
Done

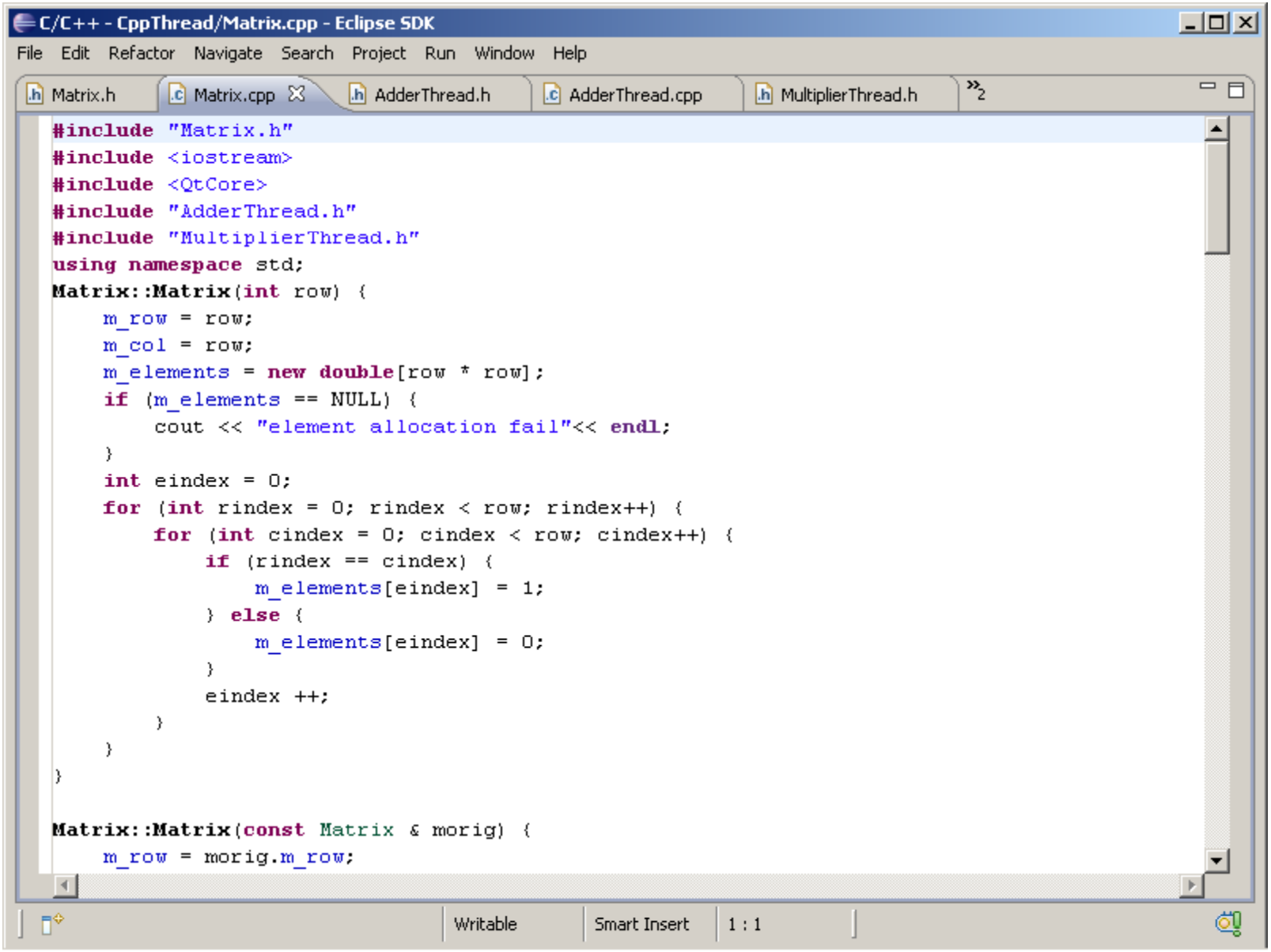

C/C++ - CppThread/MatrixTest.cpp - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
MatrixTest.cpp
AdderThread.h
AdderThread.cpp
MultiplierThread.cpp
MultiplierThread.h
#include <iostream>
#include <stdlib.h>
#include <unistd.h>
#include "Matrix.h"
using namespace std;
int main(int argc, char * argv[]) {
cout << "Number of Processors = "<< sysconf(_SC_NPROCESSORS_ONLN) << endl;
int row = 1000;
if (argc > 1) {
row = atoi(argv[1]);
}
if (row < 2) {
row = 2;
}
cout << "Number of Rows = "<< row << endl;
Matrix ma(row);
Matrix mb(row);
Matrix mc = ma.add(mb);
Matrix md = ma.multiply(mb);
return 0;
}
Writable
Smart Insert
1 : 1

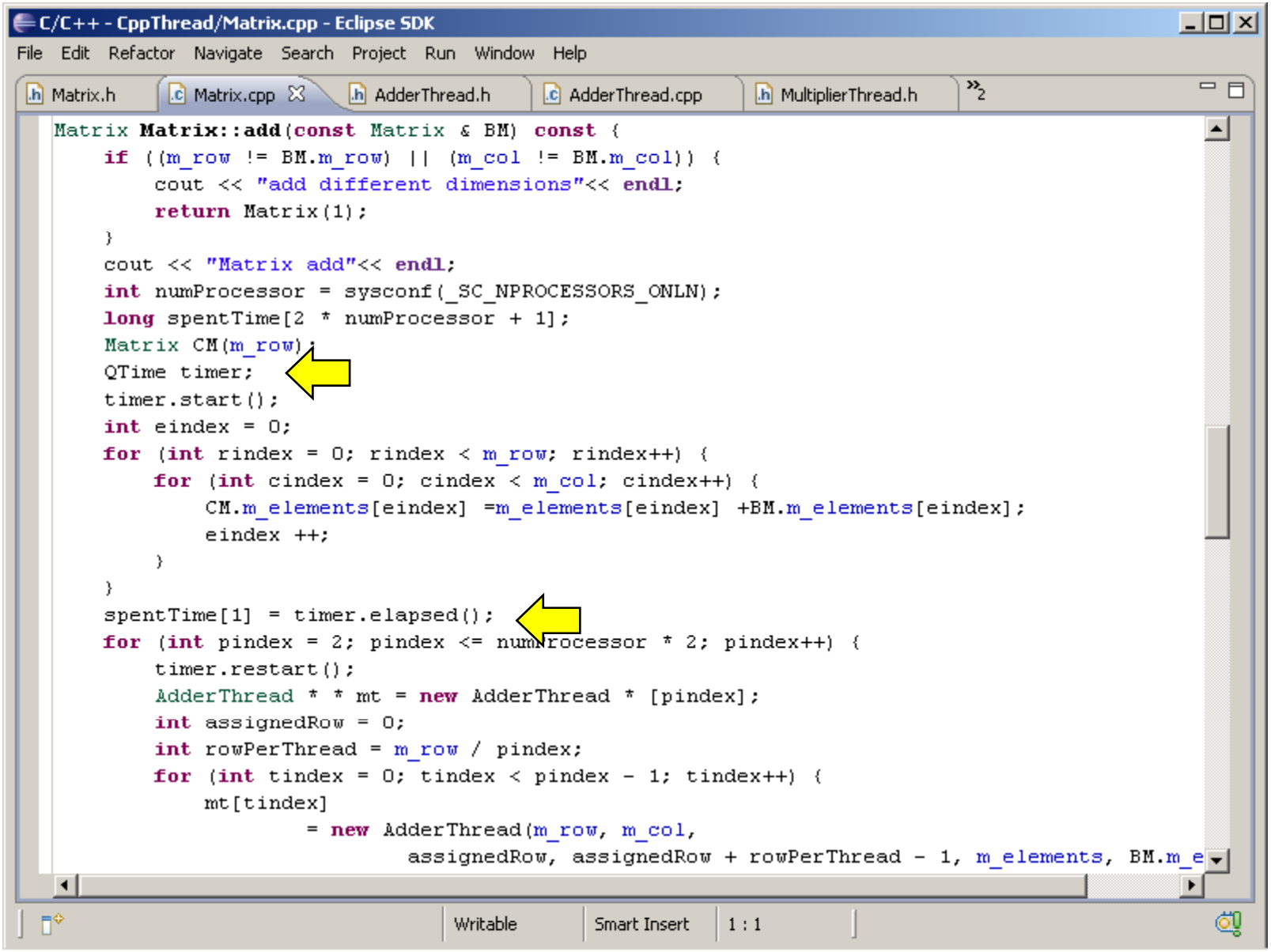

C/C++ - CppThread/Matrix.h - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
MatrixTest.cpp
Matrix.h
AdderThread.cpp
MultiplierThread.cpp
MultiplierThread.h
#ifndef MATRIX_H_
#define MATRIX_H_
class Matrix {
private:
int m_row;
int m_col;
double * m_elements;
inline int computeIndex(int row, int col) const {
return (row * m_col + col);
}
public:
Matrix(int row);
Matrix add(const Matrix & BM) const;
Matrix multiply(const Matrix & BM) const;
Matrix(const Matrix & morig);
Matrix & operator =(const Matrix & morig);
virtual ~Matrix();
};
#endif /*MATRIX_H_*/
Writable
Smart Insert
1 : 1

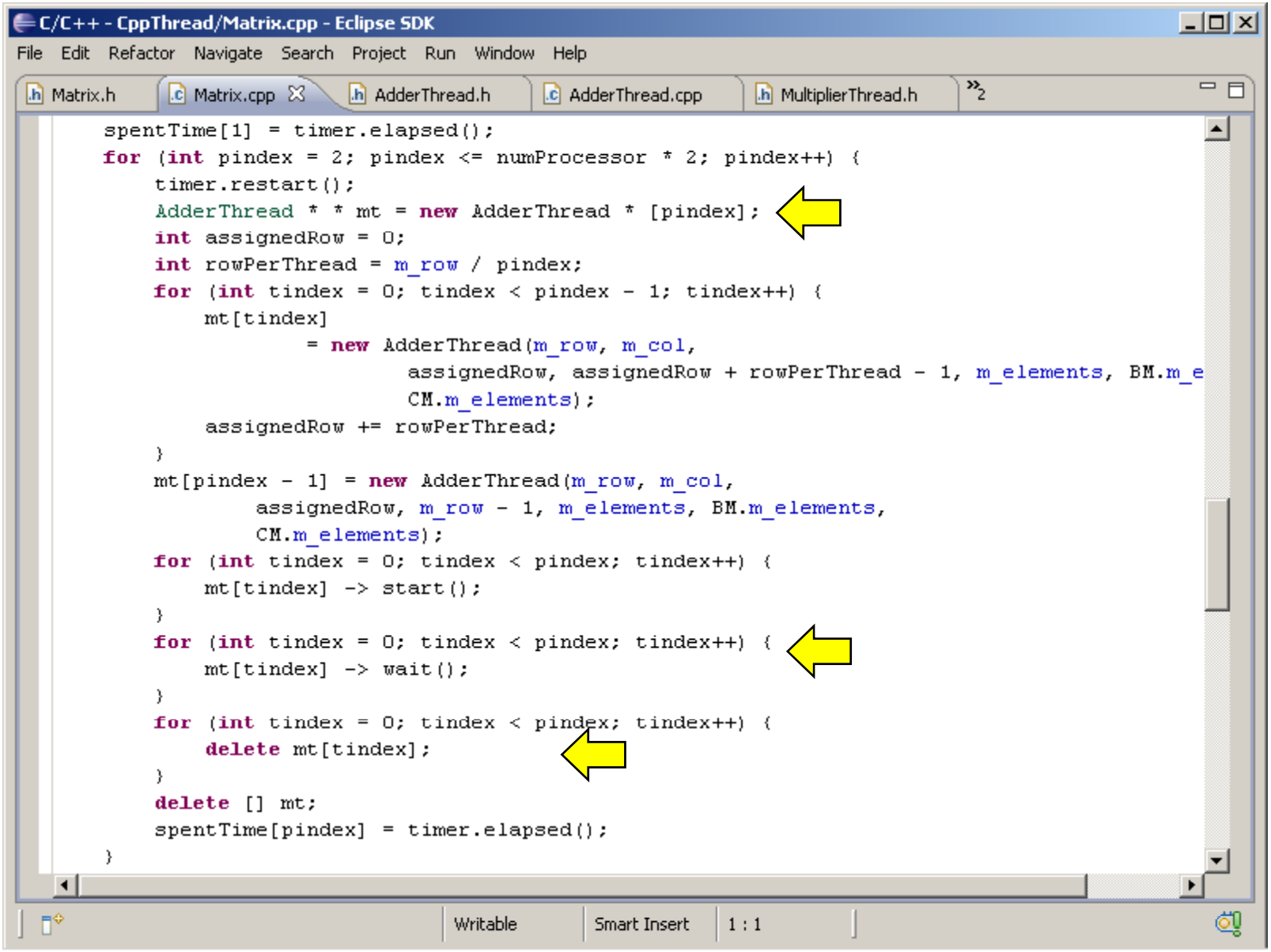

C/C++ - CppThread/AdderThread.h - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
MatrixTest.cpp
Matrix.h
AdderThread.h
MultiplierThread.cpp
MultiplierThread.h
#ifndef ADDERTHREAD_H_
#define ADDERTHREAD_H_
#include <QtCore>
class AdderThread : public QThread {
private:
int at_row;
int at_col;
int at_r1;
int at_r2;
double * at_Aelements;
double * at_Belements;
double * at_Celements;
public:
AdderThread(int r, int c, int r1, int r2, double * ma, double * mb, double * mc);
void run();
virtual ~AdderThread();
);
#endif /*ADDERTHREAD_H_*/
Writable
Smart Insert
1 : 1


C/C++ - CppThread/MultiplierThread.h - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
MatrixTest.cpp
Matrix.h
AdderThread.h
MultiplierThread.cpp
MultiplierThread.h
#ifndef MULTIPLIERTHREAD_H_
#define MULTIPLIERTHREAD_H_
#include <QtCore>
class MultiplierThread : public QThread {
private:
int mt_row;
int mt_col;
int mt_r1;
int mt_r2;
double * mt_Aelements;
double * mt_Belements;
double * mt_Celements;
public:
MultiplierThread(int r, int c, int r1, int r2, double * ma, double * mb,
double * mc);
void run();
virtual ~MultiplierThread();
};
#endif /*MULTIPLIERTHREAD_H_*/
Writable
Smart Insert
1 : 1


C/C++ - CppThread/AdderThread.cpp - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
MatrixTest.cpp
Matrix.h
AdderThread.h
AdderThread.cpp
MultiplierThread.h
#include "AdderThread.h"
AdderThread::AdderThread(int r, int c, int r1, int r2, double * ma,
double * mb, double * mc)
{
at_row = r;
at_col = c;
at_r1 = r1;
at_r2 = r2;
at_Aelements = ma;
at_Belements = mb;
at_Celements = mc;
}
AdderThread::~AdderThread() {
// do not delete anything since nothing is allocated in the constructor
}
void AdderThread::run() {
int initIndex = at_r1 * at_col;
int lastIndex = at_r2 * at_col;
for (int index = initIndex; index < lastIndex; index ++) {
at_Celements[index] = at_Aelements[index] + at_Belements[index];
}
}
Writable
Smart Insert
1 : 1


C/C++ - CppThread/Matrix.cpp - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
Matrix.h
Matrix.cpp
AdderThread.h
AdderThread.cpp
MultiplierThread.h
#include "Matrix.h"
#include <iostream>
#include <QtCore>
#include "AdderThread.h"
#include "MultiplierThread.h"
using namespace std;
Matrix::Matrix(int row) {
m_row = row;
m_col = row;
m_elements = new double[row * row];
if (m_elements == NULL) {
cout << "element allocation fail"<< endl;
}
int eindex = 0;
for (int rindex = 0; rindex < row; rindex++) {
for (int cindex = 0; cindex < row; cindex++) {
if (rindex == cindex) {
m_elements[eindex] = 1;
} else {
m_elements[eindex] = 0;
}
eindex ++;
}
}
}
Matrix::Matrix(const Matrix & morig) {
m_row = morig.m_row;
Writable
Smart Insert
1 : 1


C/C++ - CppThread/Matrix.cpp - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
Matrix.h
Matrix.cpp
AdderThread.h
AdderThread.cpp
MultiplierThread.h
Matrix Matrix::add(const Matrix & BM) const {
if ((m_row != BM.m_row) || (m_col != BM.m_col)) {
cout << "add different dimensions"<< endl;
return Matrix(1);
}
cout << "Matrix add"<< endl;
int numProcessor = sysconf(_SC_NPROCESSORS_ONLN);
long spentTime[2 * numProcessor + 1];
Matrix CM(m_row);
QTime timer;
timer.start();
int eindex = 0;
for (int rindex = 0; rindex < m_row; rindex++) {
for (int cindex = 0; cindex < m_col; cindex++) {
CM.m_elements[eindex] =m_elements[eindex] +BM.m_elements[eindex];
eindex ++;
}
}
spentTime[1] = timer.elapsed();
for (int pindex = 2; pindex <= numProcessor * 2; pindex++) {
timer.restart();
AdderThread * * mt = new AdderThread * [pindex];
int assignedRow = 0;
int rowPerThread = m_row / pindex;
for (int tindex = 0; tindex < pindex - 1; tindex++) {
mt[tindex]
= new AdderThread(m_row, m_col,
assignedRow, assignedRow + rowPerThread - 1, m_elements, BM.m_e
Writable
Smart Insert
1 : 1


C/C++ - CppThread/Matrix.cpp - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
Matrix.h
Matrix.cpp
AdderThread.h
AdderThread.cpp
MultiplierThread.h
spentTime[1] = timer.elapsed();
for (int pindex = 2; pindex <= numProcessor * 2; pindex++) {
timer.restart();
AdderThread * * mt = new AdderThread * [pindex];
int assignedRow = 0;
int rowPerThread = m_row / pindex;
for (int tindex = 0; tindex < pindex - 1; tindex++) {
mt[tindex]
= new AdderThread(m_row, m_col,
assignedRow, assignedRow + rowPerThread - 1, m_elements, BM.m_e
CM.m_elements);
assignedRow += rowPerThread;
}
mt[pindex - 1] = new AdderThread(m_row, m_col,
assignedRow, m_row - 1, m_elements, BM.m_elements,
CM.m_elements);
for (int tindex = 0; tindex < pindex; tindex++) {
mt[tindex] -> start();
}
for (int tindex = 0; tindex < pindex; tindex++) {
mt[tindex] -> wait();
}
for (int tindex = 0; tindex < pindex; tindex++) {
delete mt[tindex];
}
delete [] mt;
spentTime[pindex] = timer.elapsed();
}
Writable
Smart Insert
1 : 1

# ECE 462
# Object-Oriented Programming using C++ and Java

# Parallel Programs

Yung-Hsiang Lu
yunglu@purdue.edu

# Classification

- S: single, M: multiple, I: instruction, D: data
- SISD: single instruction single data $\Rightarrow$ sequential program on a single-processor computer
- SIMD: single instruction multiple data $\Rightarrow$ multiple data elements are operated in the same way, matrix addition
- MISD: multiple instruction single data $\Rightarrow$ relatively rare, find the eigenvalues and transpose of the same matrix
- MIMD: multiple instruction multiple data $\Rightarrow$ most general form of parallel computation

# Parallelization

- data parallelism $\Rightarrow$ break a big piece of data into smaller units
  - different rows of matrices
  - different bank accounts
  - different books
- instruction parallelism $\Rightarrow$ break a long function into smaller functions
  - different bank transactions
  - different purchase options
- granularity of data and instruction parallelism

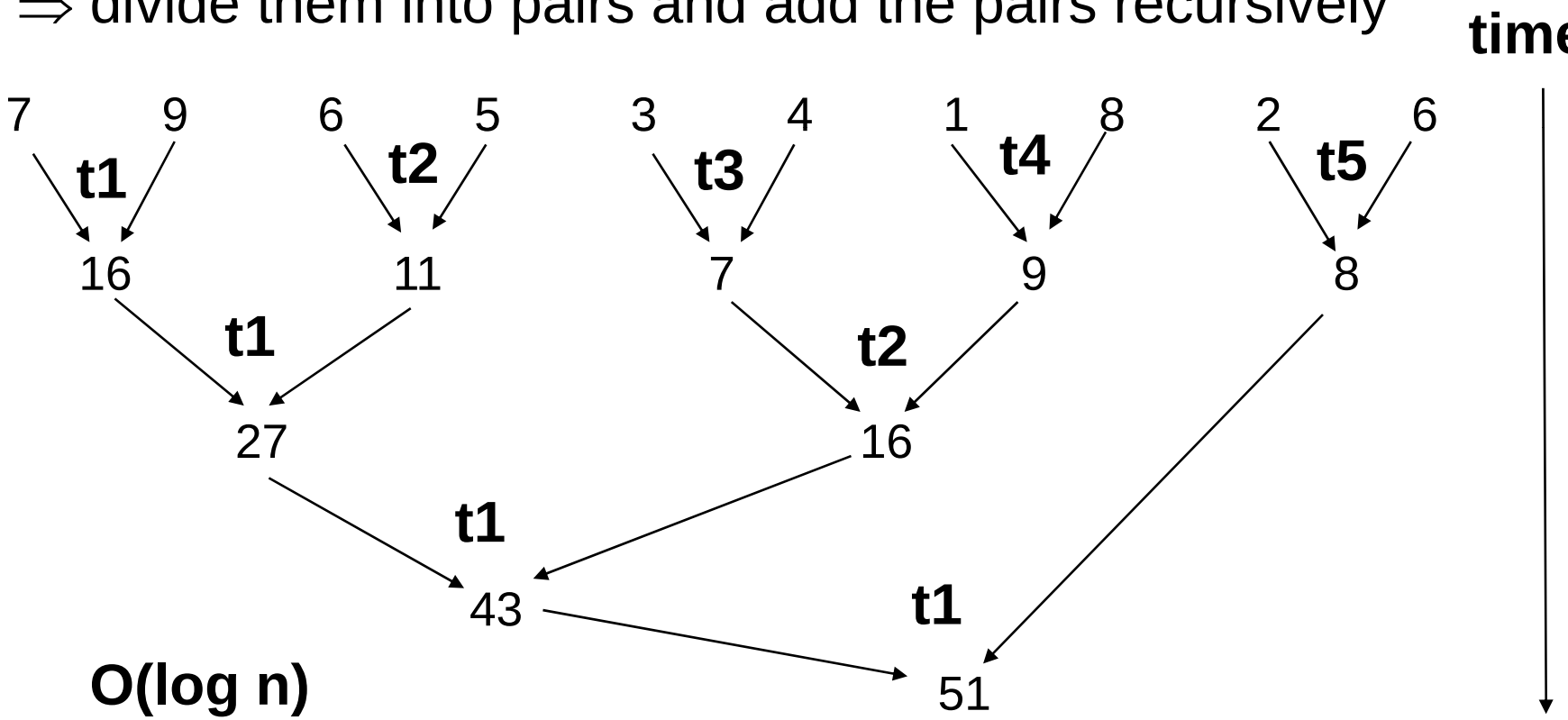
# Data Sharing

- read only $\Rightarrow$ no problem
- write $\Rightarrow$ **BIG** problem, if you are not careful
  - read (d0) before write (d1)
  - write (d0) before read (d1)
  - write (d0) before write (d1)
- will be explained later on the topic of synchronization, mutual exclusion, and critical section

# Data Partition

- how to add a group of number?

$\Rightarrow$ divide them into pairs and add the pairs recursively



# Assumptions

- the time and space (overhead) for thread creation, scheduling, and destruction can be ignored
- number of threads = half the number of operands (initially)
- number of processor = half the number of operands (initially)
- all processors can read the operands within the same amount of time
- the intermediate results are accessible by other processors

# Structure of Parallel Programs