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

Exception Handling in C++

Yung-Hsiang Lu
yunglu@purdue.edu

Prevent, Detect, and Handle Problems

Avoid Assertion

- assert (something must be true);
  - Your program crashes when it is not true.
  - Do **not** use assert. Instead, you should **handle** the situation when it is false.
  - Some people encourage using assertions but these people are wrong. Don't listen to them.
- always check the return value of function calls
  - memory allocation
  - open a file
  - send data through network
  - ...

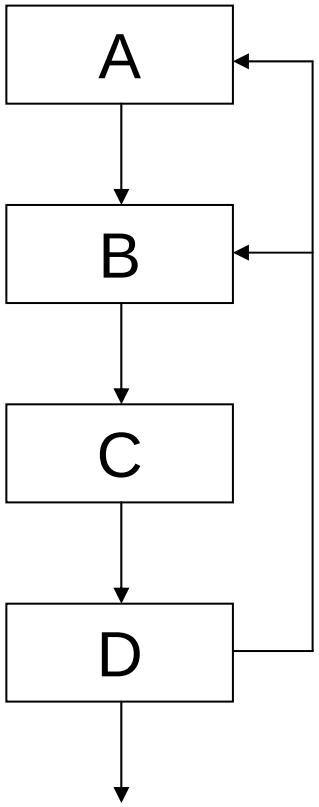
Handle Problems

- A typical approach is to check before proceeding:

```
retval = func(parameters);
if (retval < 0) {
  ...
}
```

- However, sometimes the immediate caller does not know how to handle the problem. This is especially common for reused code.

Call Stack



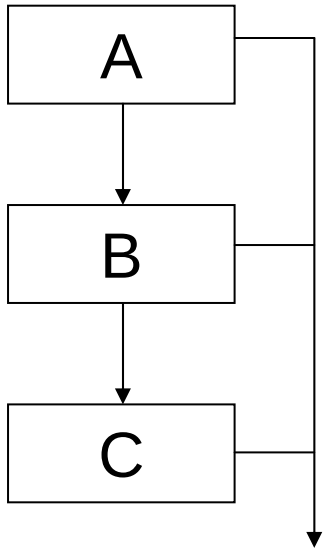
try to open a file
that does not exist

- A calls B; B calls C; C calls D.
- Should D create the file?
- Can D make that decision?
- Maybe, A intends to ask the user to give a different file name.
- Maybe, B wants to simply skip the file.

$\Rightarrow$ need a way to inform a caller that is several "frames" away

Concepts of Exception Handling

try {



} catch (Exception) {

    handle exception;

}

- A, B, or C can throw an exception and it will be caught by

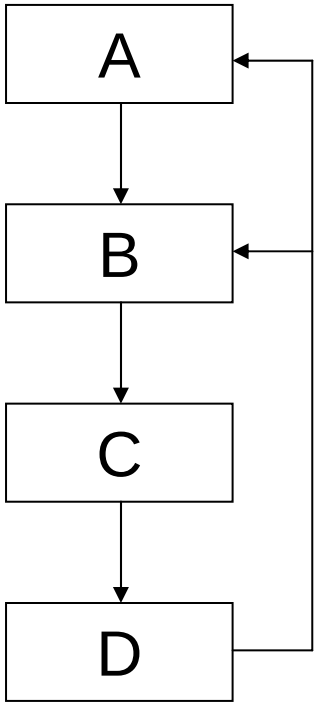
  try { ... } catch

- If an exception is not caught, the program terminates (like “crash”).

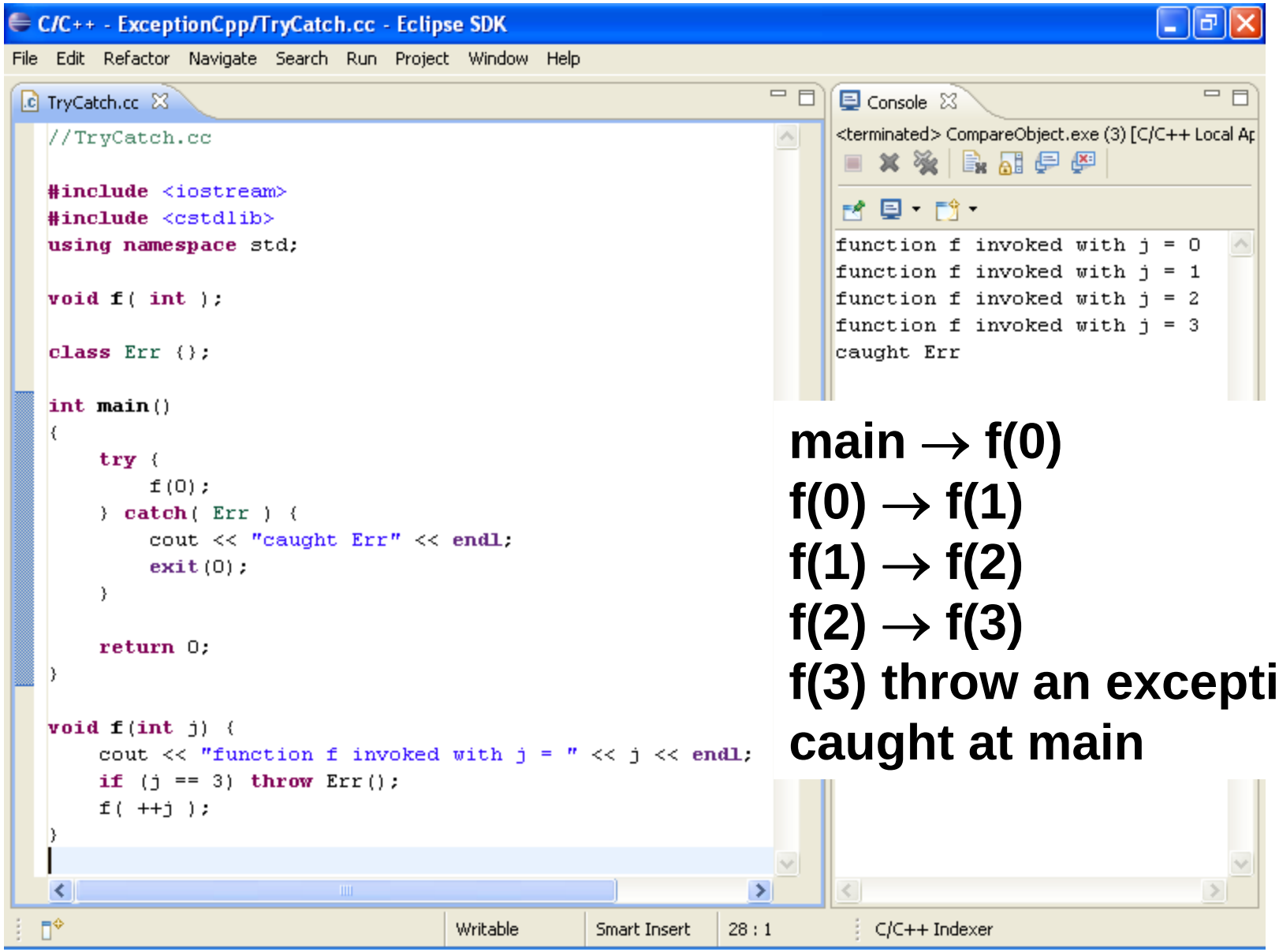
throw-catch

- When an unexpected situation occurs and the current function cannot handle it, it **throw**s an exception.
- The exception is sent to the immediate caller. If it does not **catch** this exception, the exception is thrown to the next caller in the call stack.
- If an exception is not caught anywhere in the call stack, the program terminates.
- In C++, a function may be called even if the caller does not catch the exception that may be thrown by the callee.

Catching an Exception

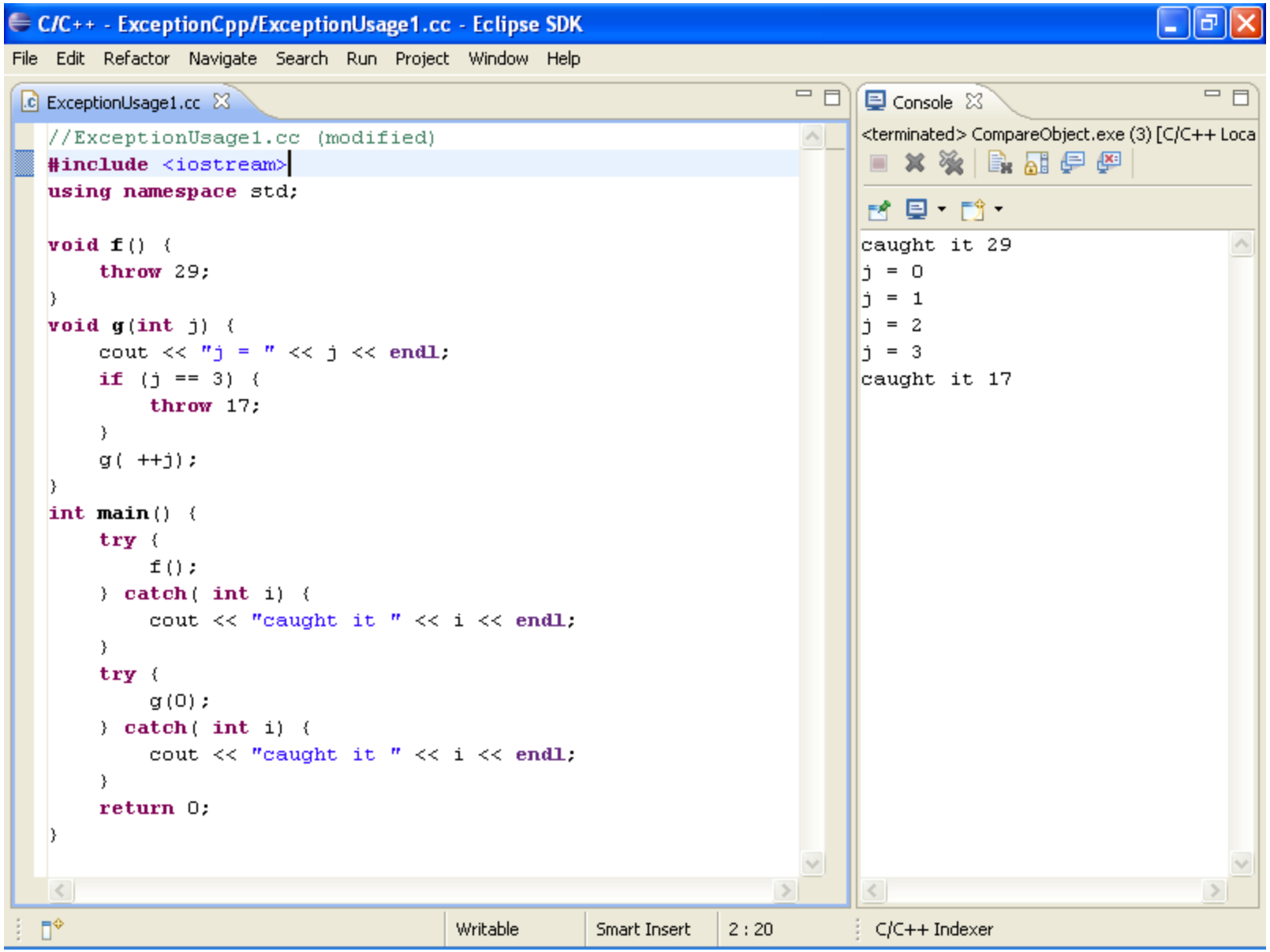


- An exception is caught by the closest caller in the stack.
- If both A and B catch the same exception and D throws and exception, B will catch it.
- The handling code **can throw** the exception **again** to its call stack; the code can also throw a **different** exception.
- An exception may pass **parameters** through the call stack.

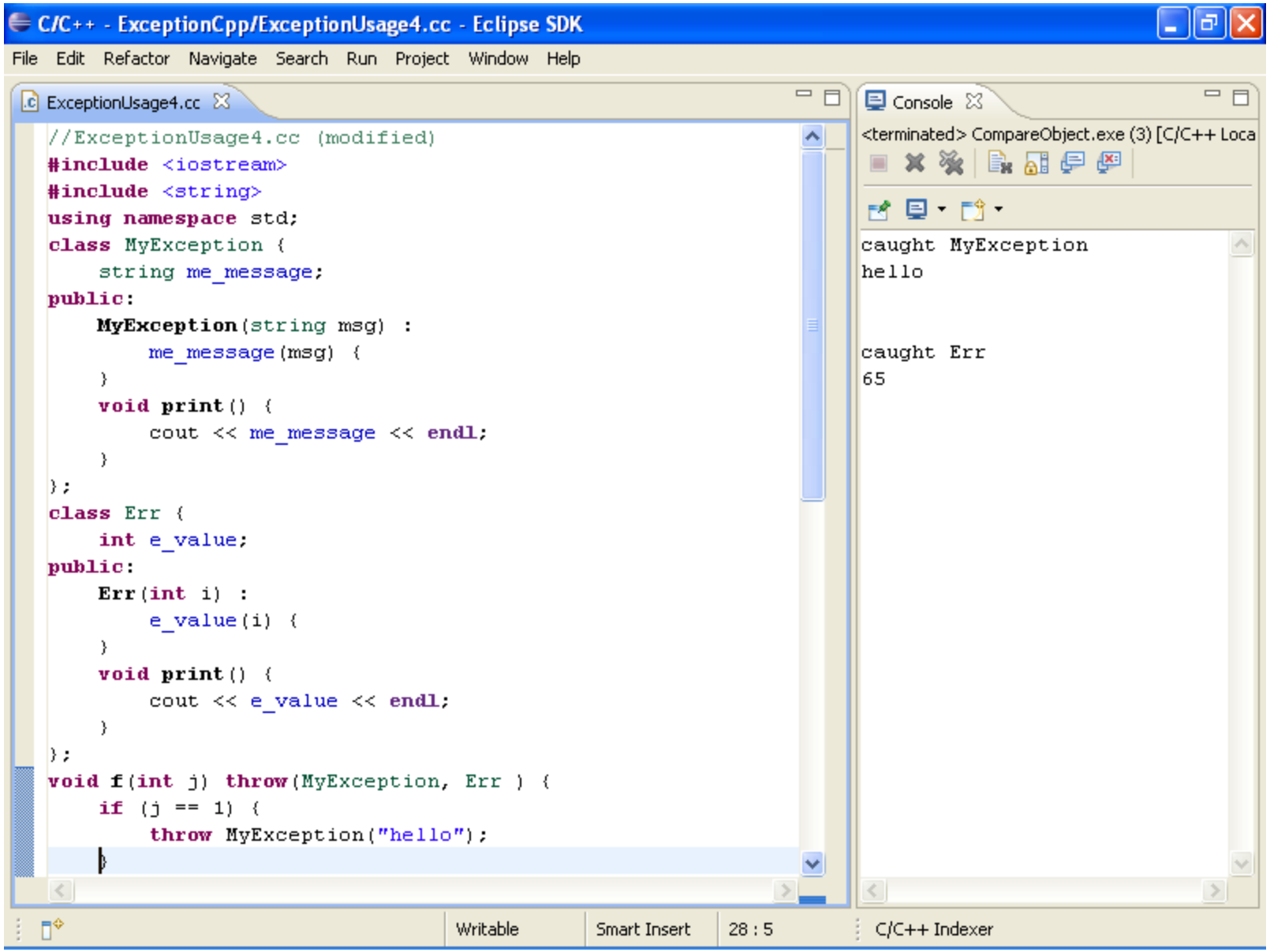


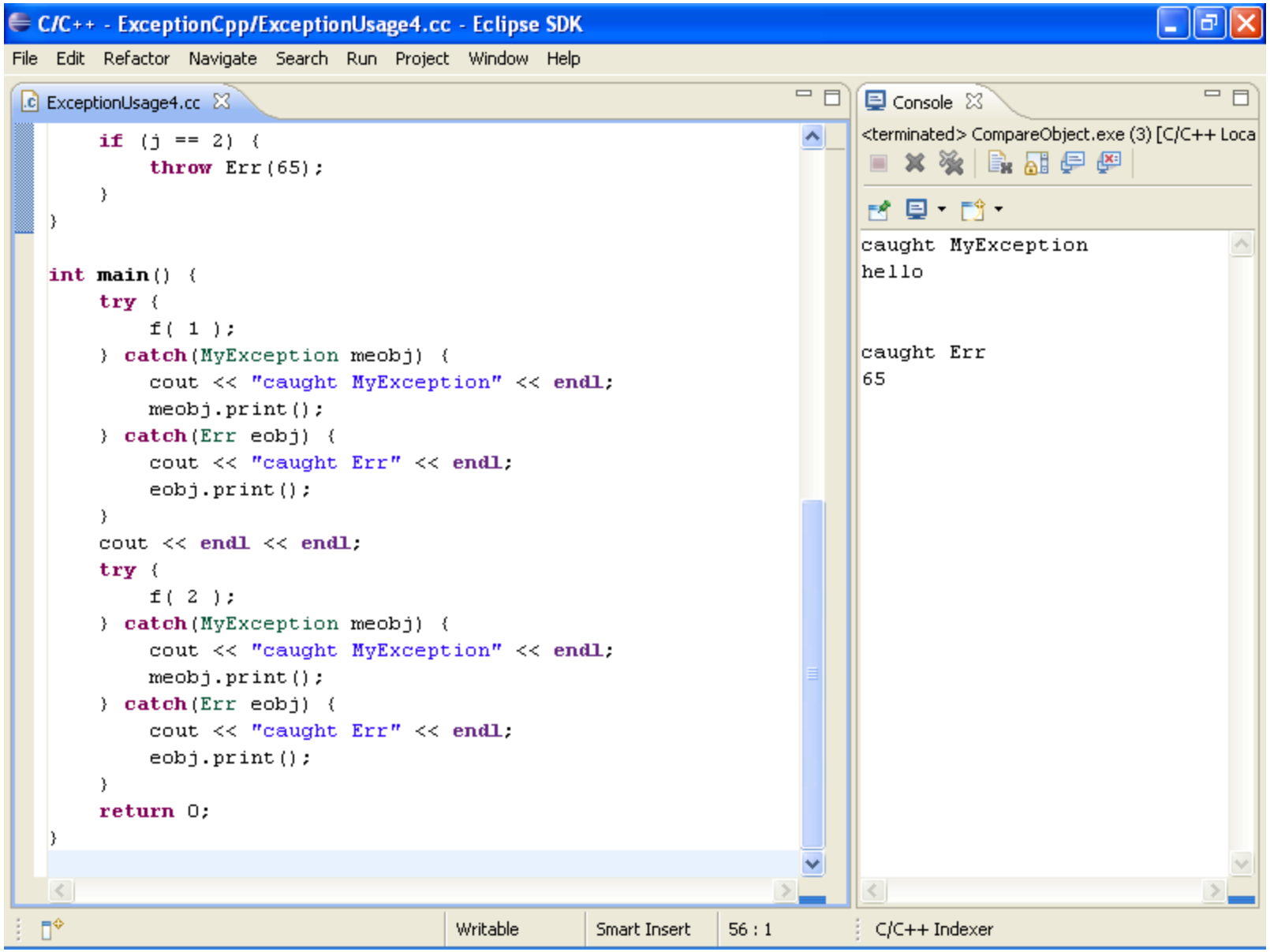
main → f(0)
f(0) → f(1)
f(1) → f(2)
f(2) → f(3)
f(3) throw an exception
caught at main

Exception with Primitive Type (int)


C/C++ - ExceptionCpp/ExceptionUsage1.cc - Eclipse SDK
File Edit Refactor Navigate Search Run Project Window Help
ExceptionUsage1.cc
//ExceptionUsage1.cc (modified)
#include <iostream>
using namespace std;
void f() {
throw 29;
}
void g(int j) {
cout << "j = " << j << endl;
if (j == 3) {
throw 17;
}
g(++j);
}
int main() {
try {
f();
} catch(int i) {
cout << "caught it " << i << endl;
}
try {
g(0);
} catch(int i) {
cout << "caught it " << i << endl;
}
return 0;
}
Console
<terminated> CompareObject.exe (3) [C/C++ Loca
caught it 29
j = 0
j = 1
j = 2
j = 3
caught it 17
Writable
Smart Insert
2 : 20
C/C++ Indexer

Exception with Objects and Declarations


C/C++ - ExceptionCpp/ExceptionUsage4.cc - Eclipse SDK
File Edit Refactor Navigate Search Run Project Window Help
ExceptionUsage4.cc
//ExceptionUsage4.cc (modified)
#include <iostream>
#include <string>
using namespace std;
class MyException {
string me_message;
public:
MyException(string msg) :
me_message(msg) {
}
void print() {
cout << me_message << endl;
}
};
class Err {
int e_value;
public:
Err(int i) :
e_value(i) {
}
void print() {
cout << e_value << endl;
}
};
void f(int j) throw(MyException, Err) {
if (j == 1) {
throw MyException("hello");
}
Console
<terminated> CompareObject.exe (3) [C/C++ Loca
caught MyException
hello
caught Err
65
Writable
Smart Insert
28 : 5
C/C++ Indexer


C/C++ - ExceptionCpp/ExceptionUsage4.cc - Eclipse SDK
File Edit Refactor Navigate Search Run Project Window Help
ExceptionUsage4.cc
if (j == 2) {
throw Err(65);
}
}
int main() {
try {
f(1);
} catch(MyException meobj) {
cout << "caught MyException" << endl;
meobj.print();
} catch(Err eobj) {
cout << "caught Err" << endl;
eobj.print();
}
cout << endl << endl;
try {
f(2);
} catch(MyException meobj) {
cout << "caught MyException" << endl;
meobj.print();
} catch(Err eobj) {
cout << "caught Err" << endl;
eobj.print();
}
return 0;
}
Console
<terminated> CompareObject.exe (3) [C/C++ Loca
caught MyException
hello
caught Err
65
Writable
Smart Insert
56 : 1
C/C++ Indexer

Exceptions in C++ and Java

C++	Java
can throw exceptions of objects or primitive types (such as int)	must be objects of classes derived from Exception
does not have to, but preferred	must declare what exceptions may be thrown
does not have to, but preferred	must be enclosed within a try-catch block, checked by compiler
meobj not necessary, but preferred	the catch block must identify the object, for example, catch (MyException **meobj**) {
may throw different types of exceptions	same
does not have the equivalent	allows **finally**

Self Test

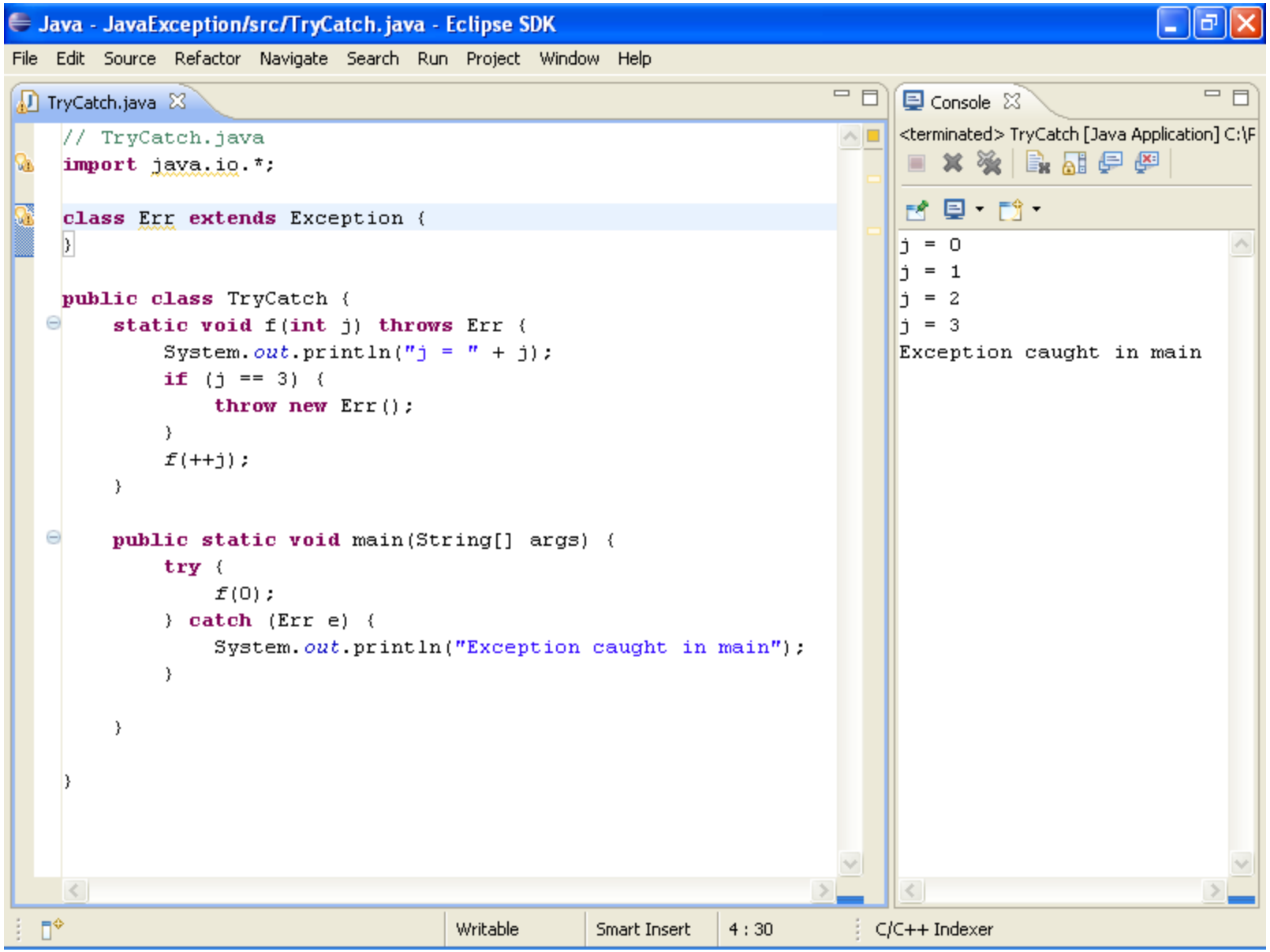
ECE 462
Object-Oriented Programming using C++ and Java

Exception Handling in Java

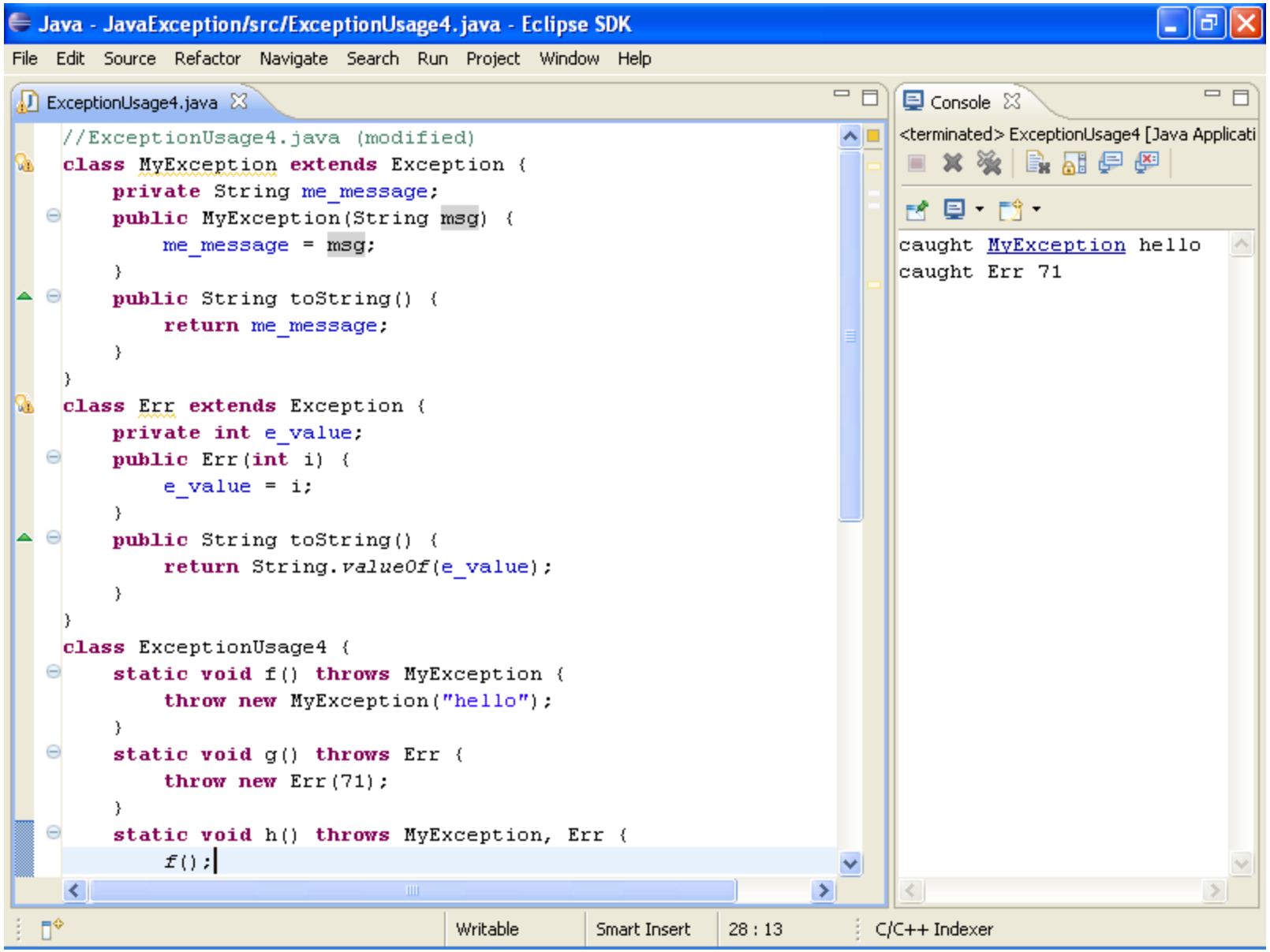
Yung-Hsiang Lu
yunglu@purdue.edu

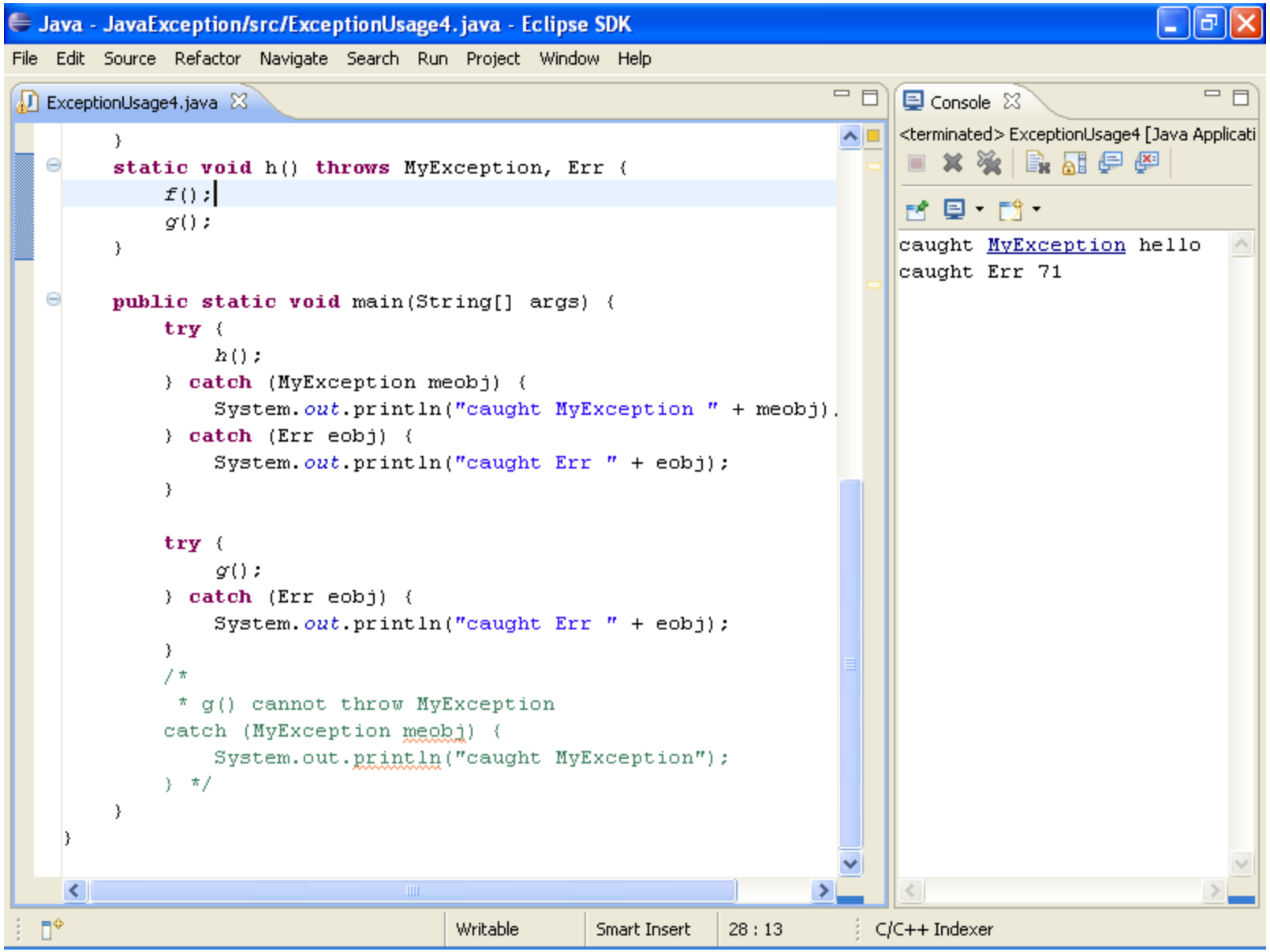
try-catch-finally

```
try {
    ...
} catch (exception_type1 id1) {
    ...
} catch (exception_type2 id2) {
    ...
} catch (exception_type3 id3) {
    ...
} finally {
    // code here always executes
    // regardless whether exception has occurred
}
```

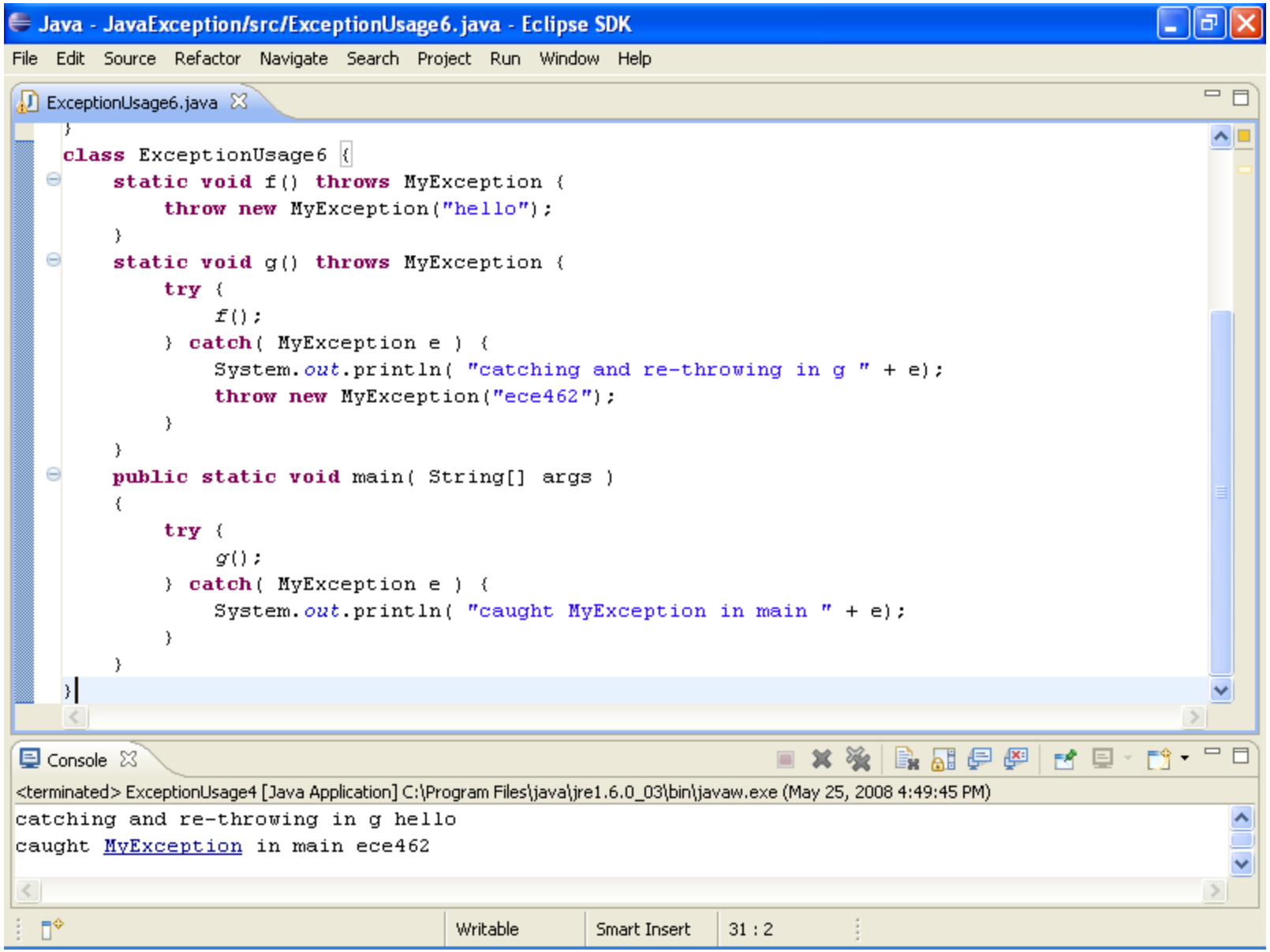

Java - JavaException/src/TryCatch.java - Eclipse SDK
File Edit Source Refactor Navigate Search Run Project Window Help
TryCatch.java
// TryCatch.java
import java.io.*;
class Err extends Exception {
}
public class TryCatch {
static void f(int j) throws Err {
System.out.println("j = " + j);
if (j == 3) {
throw new Err();
}
f(++j);
}
public static void main(String[] args) {
try {
f(0);
} catch (Err e) {
System.out.println("Exception caught in main");
}
}
}
Console
<terminated> TryCatch [Java Application] C:\F
j = 0
j = 1
j = 2
j = 3
Exception caught in main
Writable
Smart Insert
4 : 30
C/C++ Indexer

Multiple Exceptions

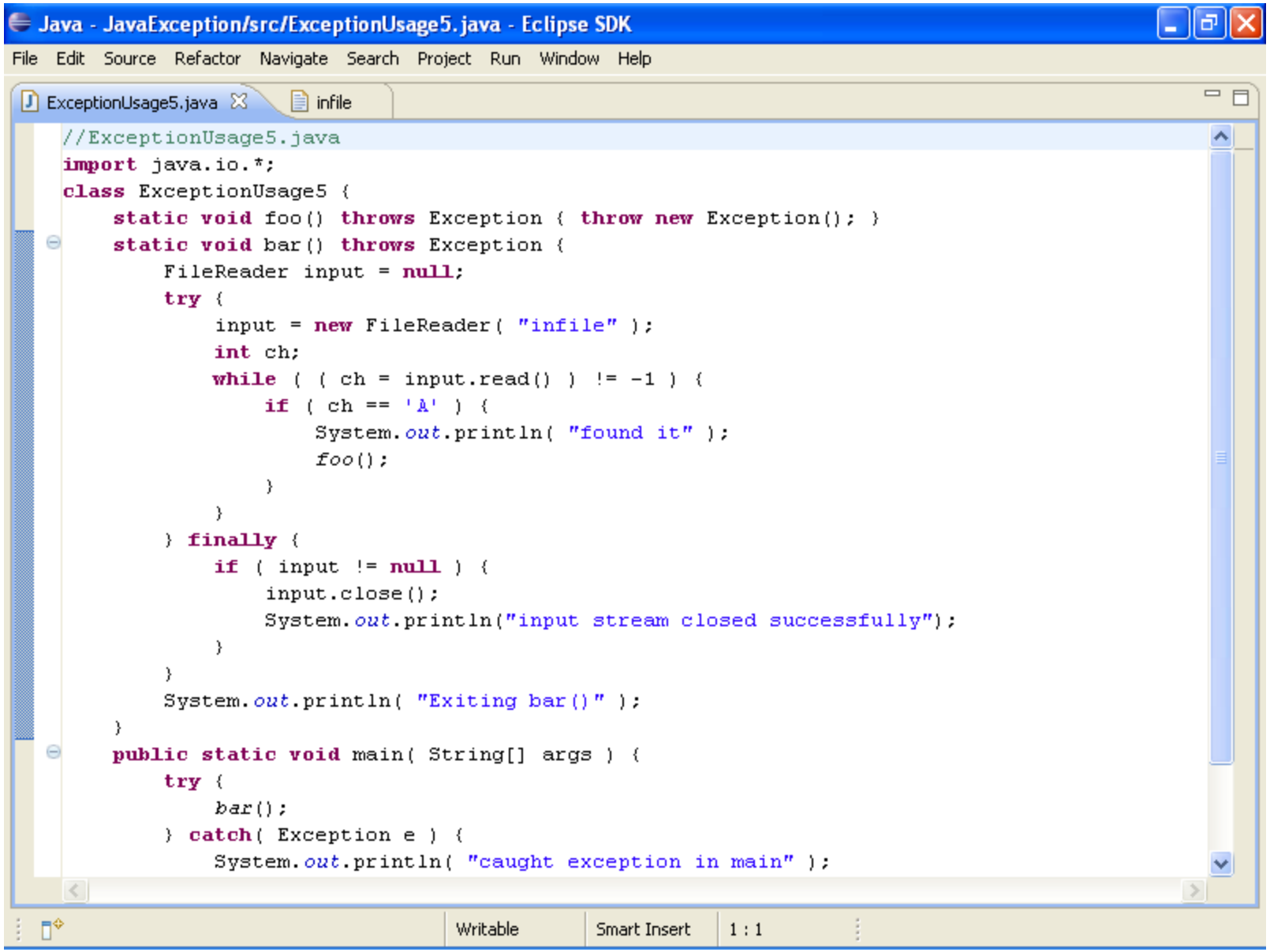

Java - JavaException/src/ExceptionUsage4.java - Eclipse SDK
File Edit Source Refactor Navigate Search Run Project Window Help
ExceptionUsage4.java
//ExceptionUsage4.java (modified)
class MyException extends Exception {
private String me_message;
public MyException(String msg) {
me_message = msg;
}
public String toString() {
return me_message;
}
}
class Err extends Exception {
private int e_value;
public Err(int i) {
e_value = i;
}
public String toString() {
return String.valueOf(e_value);
}
}
class ExceptionUsage4 {
static void f() throws MyException {
throw new MyException("hello");
}
static void g() throws Err {
throw new Err(71);
}
static void h() throws MyException, Err {
f();
Console
<terminated> ExceptionUsage4 [Java Applicati
caught MyException hello
caught Err 71
Writable
Smart Insert
28 : 13
C/C++ Indexer

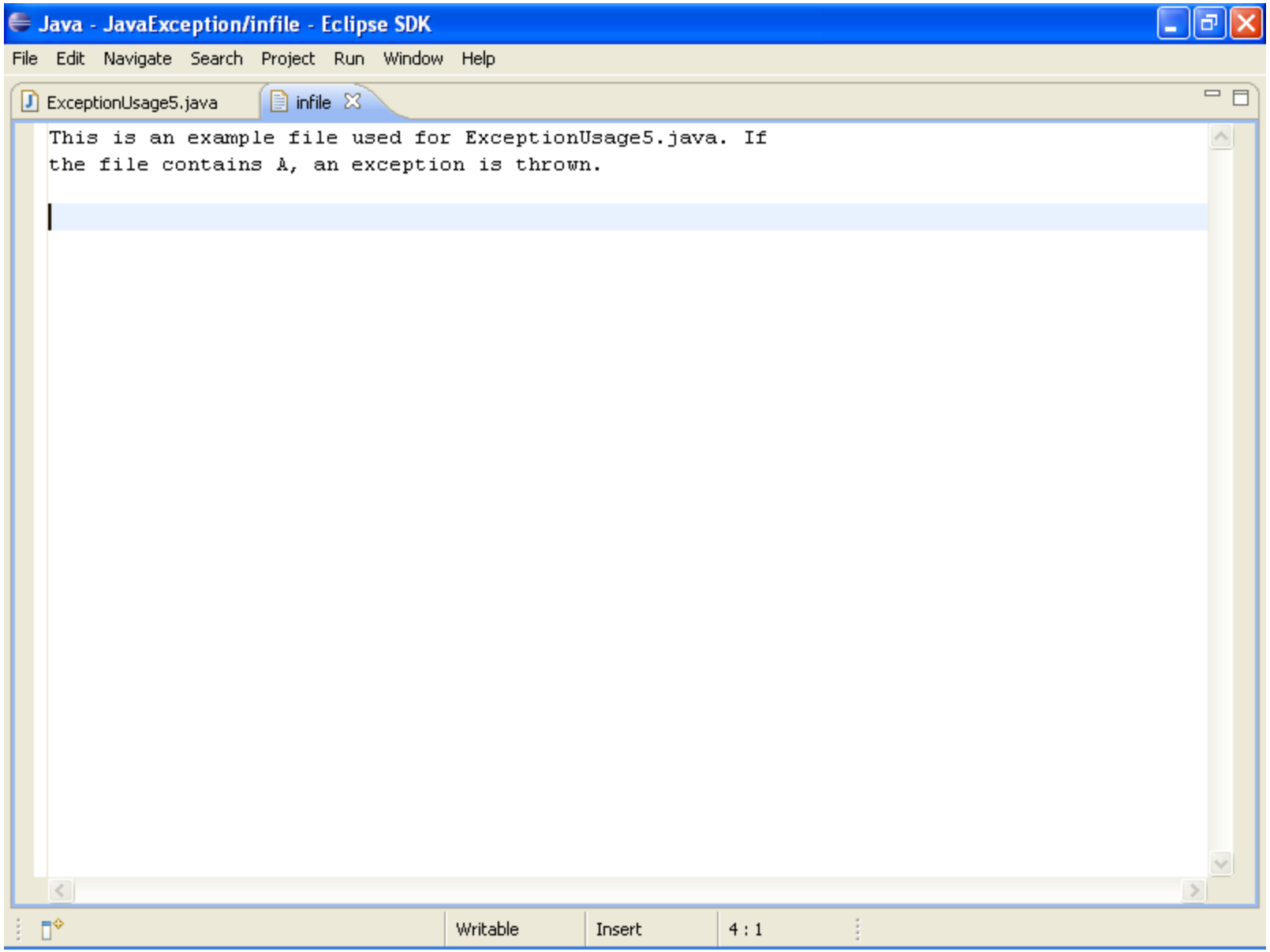

Java - JavaException/src/ExceptionUsage4.java - Eclipse SDK
File Edit Source Refactor Navigate Search Run Project Window Help
ExceptionUsage4.java
}
static void h() throws MyException, Err {
f();
g();
}
public static void main(String[] args) {
try {
h();
} catch (MyException meobj) {
System.out.println("caught MyException " + meobj);
} catch (Err eobj) {
System.out.println("caught Err " + eobj);
}
try {
g();
} catch (Err eobj) {
System.out.println("caught Err " + eobj);
}
/*
* g() cannot throw MyException
catch (MyException meobj) {
System.out.println("caught MyException");
} */
}
}
Console
<terminated> ExceptionUsage4 [Java Applicati
caught MyException hello
caught Err 71
Writable
Smart Insert
28 : 13
C/C++ Indexer

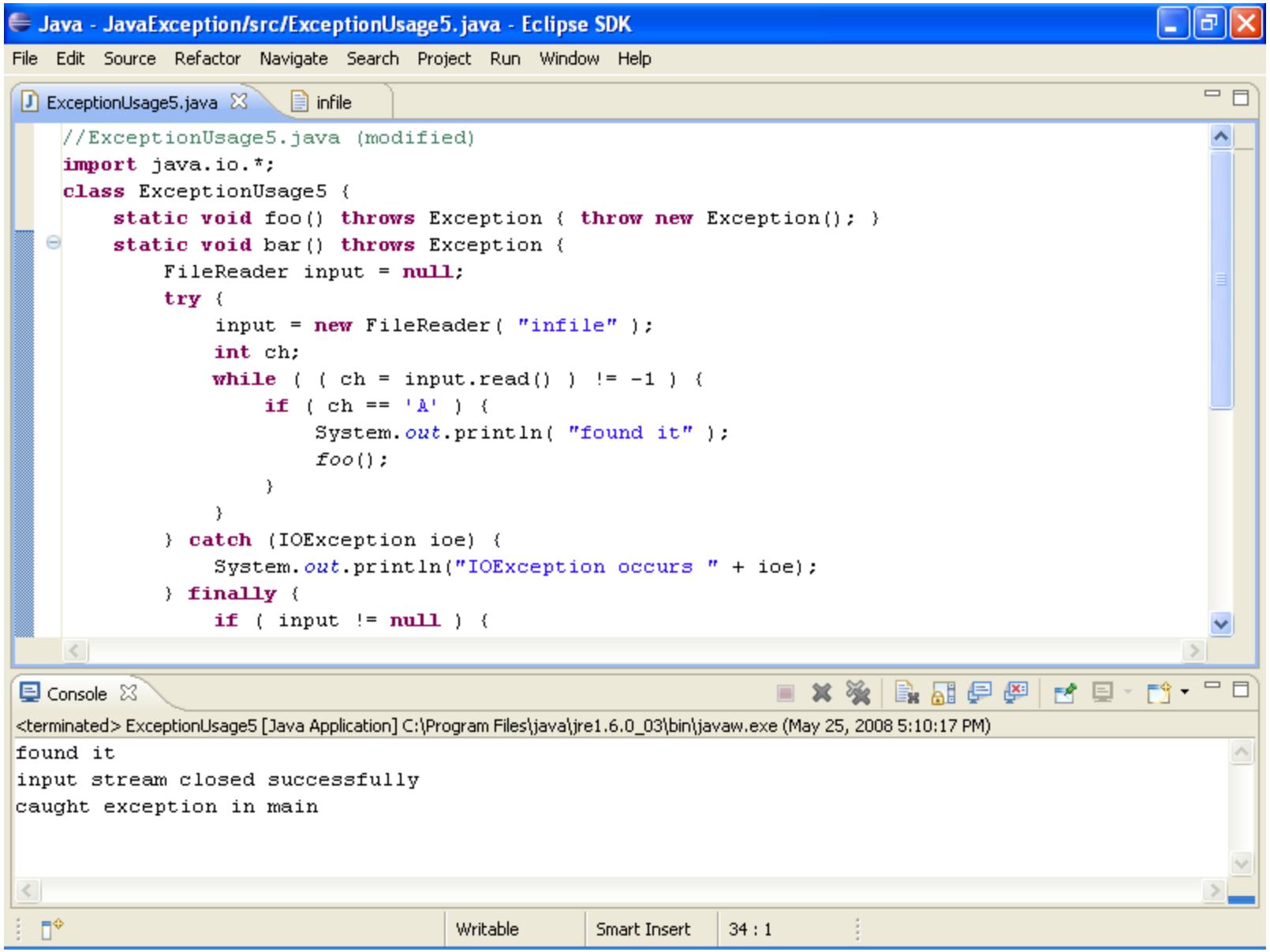
Catch and re-Throw


Java - JavaException/src/ExceptionUsage6.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
ExceptionUsage6.java
}
class ExceptionUsage6 {
static void f() throws MyException {
throw new MyException("hello");
}
static void g() throws MyException {
try {
f();
} catch(MyException e) {
System.out.println("catching and re-throwing in g " + e);
throw new MyException("ece462");
}
}
public static void main(String[] args)
{
try {
g();
} catch(MyException e) {
System.out.println("caught MyException in main " + e);
}
}
}
Console
<terminated> ExceptionUsage4 [Java Application] C:\Program Files\java\jre1.6.0_03\bin\javaw.exe (May 25, 2008 4:49:45 PM)
catching and re-throwing in g hello
caught MyException in main ece462
Writable
Smart Insert
31 : 2

Exception and File IO


Java - JavaException/src/ExceptionUsage5.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
ExceptionUsage5.java
infile
//ExceptionUsage5.java
import java.io.*;
class ExceptionUsage5 {
static void foo() throws Exception { throw new Exception(); }
static void bar() throws Exception {
FileReader input = null;
try {
input = new FileReader("infile");
int ch;
while ((ch = input.read()) != -1) {
if (ch == 'A') {
System.out.println("found it");
foo();
}
}
} finally {
if (input != null) {
input.close();
System.out.println("input stream closed successfully");
}
}
System.out.println("Exiting bar()");
}
public static void main(String[] args) {
try {
bar();
} catch(Exception e) {
System.out.println("caught exception in main");
Writable
Smart Insert
1 : 1


Java - JavaException/infile - Eclipse SDK
File Edit Navigate Search Project Run Window Help
ExceptionUsage5.java
infile
This is an example file used for ExceptionUsage5.java. If
the file contains A, an exception is thrown.
Writable
Insert
4 : 1


Java - JavaException/src/ExceptionUsage5.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
ExceptionUsage5.java
infile
//ExceptionUsage5.java (modified)
import java.io.*;
class ExceptionUsage5 {
static void foo() throws Exception { throw new Exception(); }
static void bar() throws Exception {
FileReader input = null;
try {
input = new FileReader("infile");
int ch;
while ((ch = input.read()) != -1) {
if (ch == 'A') {
System.out.println("found it");
foo();
}
}
} catch (IOException ioe) {
System.out.println("IOException occurs " + ioe);
} finally {
if (input != null) {
Console
<terminated> ExceptionUsage5 [Java Application] C:\Program Files\java\jre1.6.0_03\bin\javaw.exe (May 25, 2008 5:10:17 PM)
found it
input stream closed successfully
caught exception in main
Writable
Smart Insert
34 : 1

Overhead of Exception Handling

- Do not overuse exceptions.
- In many cases, if the errors can be detected by the return code

```
retval = function(...);
if (retval < 0) { ... /* handle error */ }
```

  then, you should do so without using try-catch.
- Detect problems early, instead of using exceptions. For example, check whether network connection is valid before sending data (and catch exception when the sending fails).
- Catch an exception early and prevent its propagation.

Self Test

Execution Flow

```
try {
    f1();
    f2();
} catch (exception_type1 id1) {
    f3();
} catch (exception_type2 id2) {
    ...
} catch (exception_type3 id3) {
    ...
}
```

- If f1() throws an exception, will f2() be executed?
- f1() throws and exception and it is caught by the first catch; f3() throws another exception, will the following catch block handle it?

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

Static Member and Memory Sharing in C++

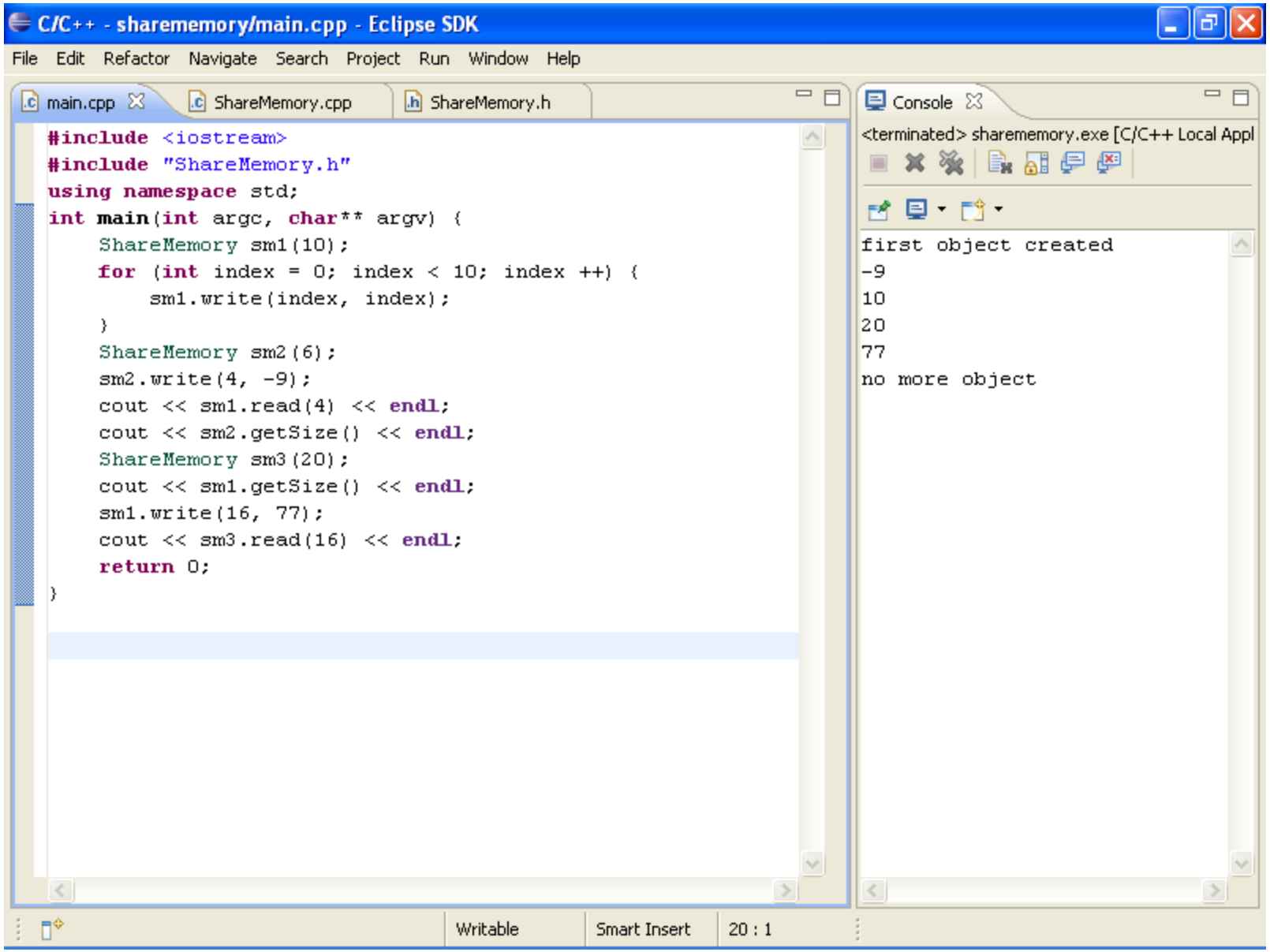
Yung-Hsiang Lu
yunglu@purdue.edu

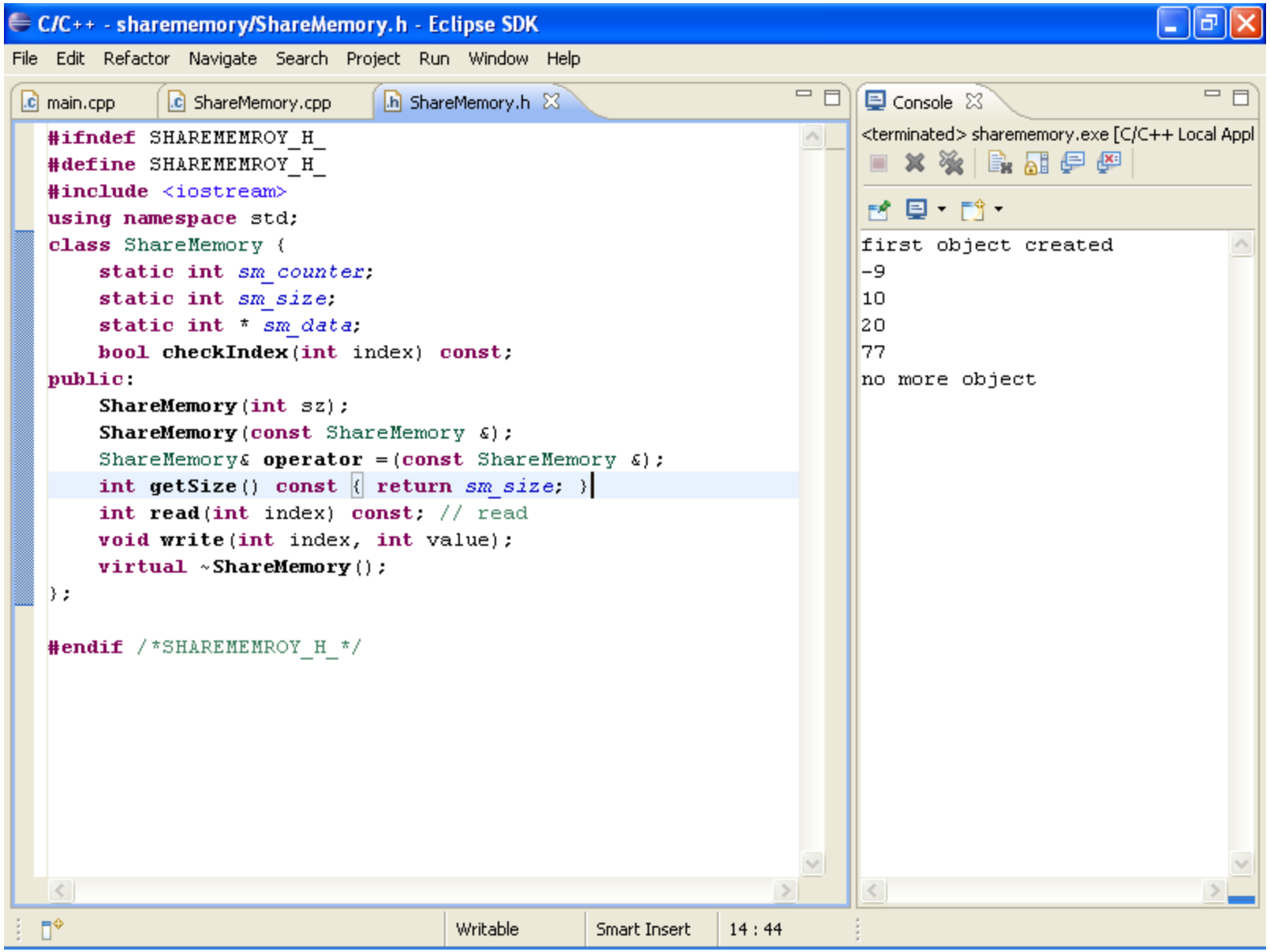
Static Member

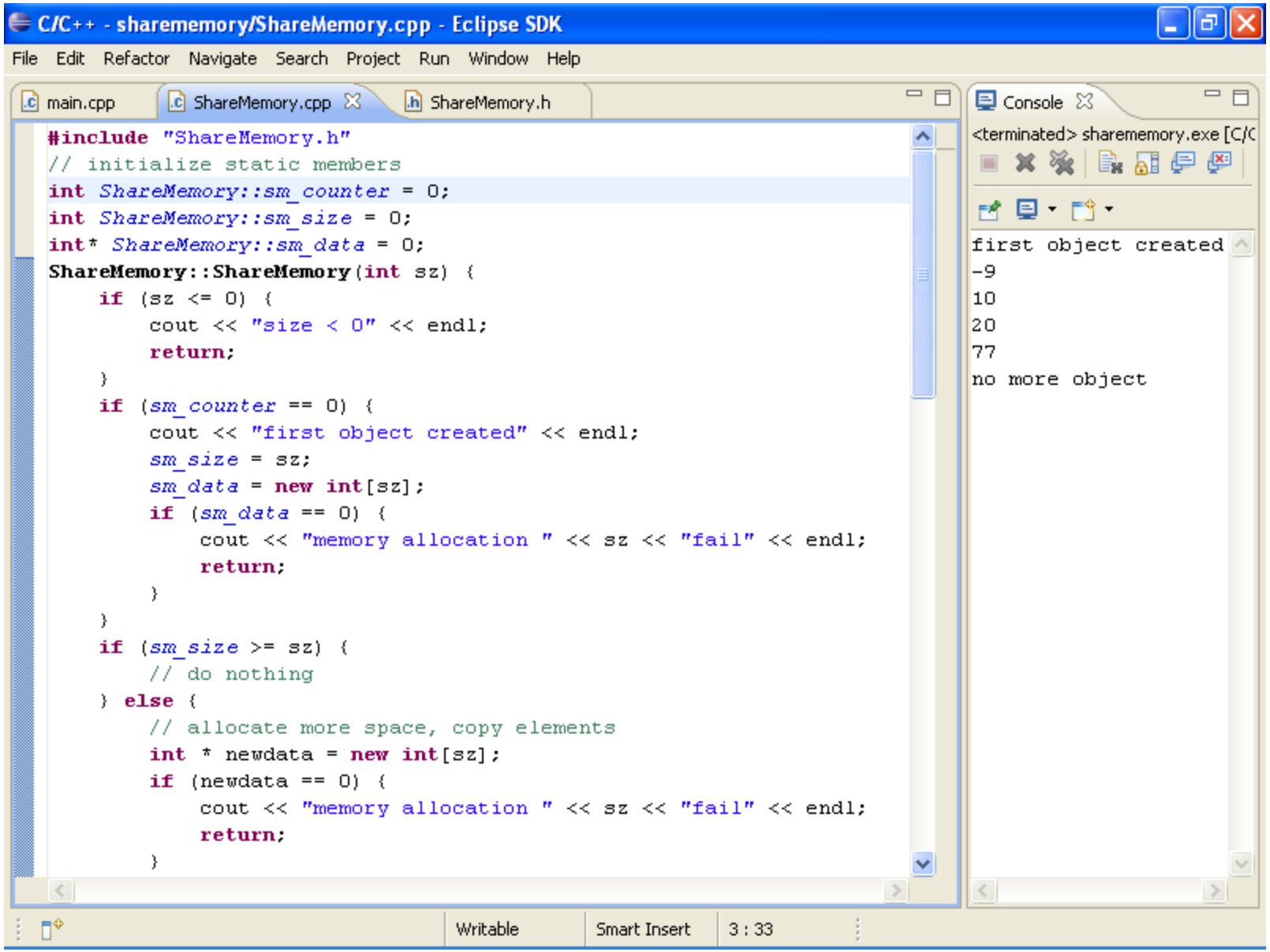
- A static member (attribute) is shared and accessible by all objects of this class.
- A static function can be called without creating an object.
- A static function can only access static attributes or call static functions.
- Static attributes can also be accessed in non-static functions.

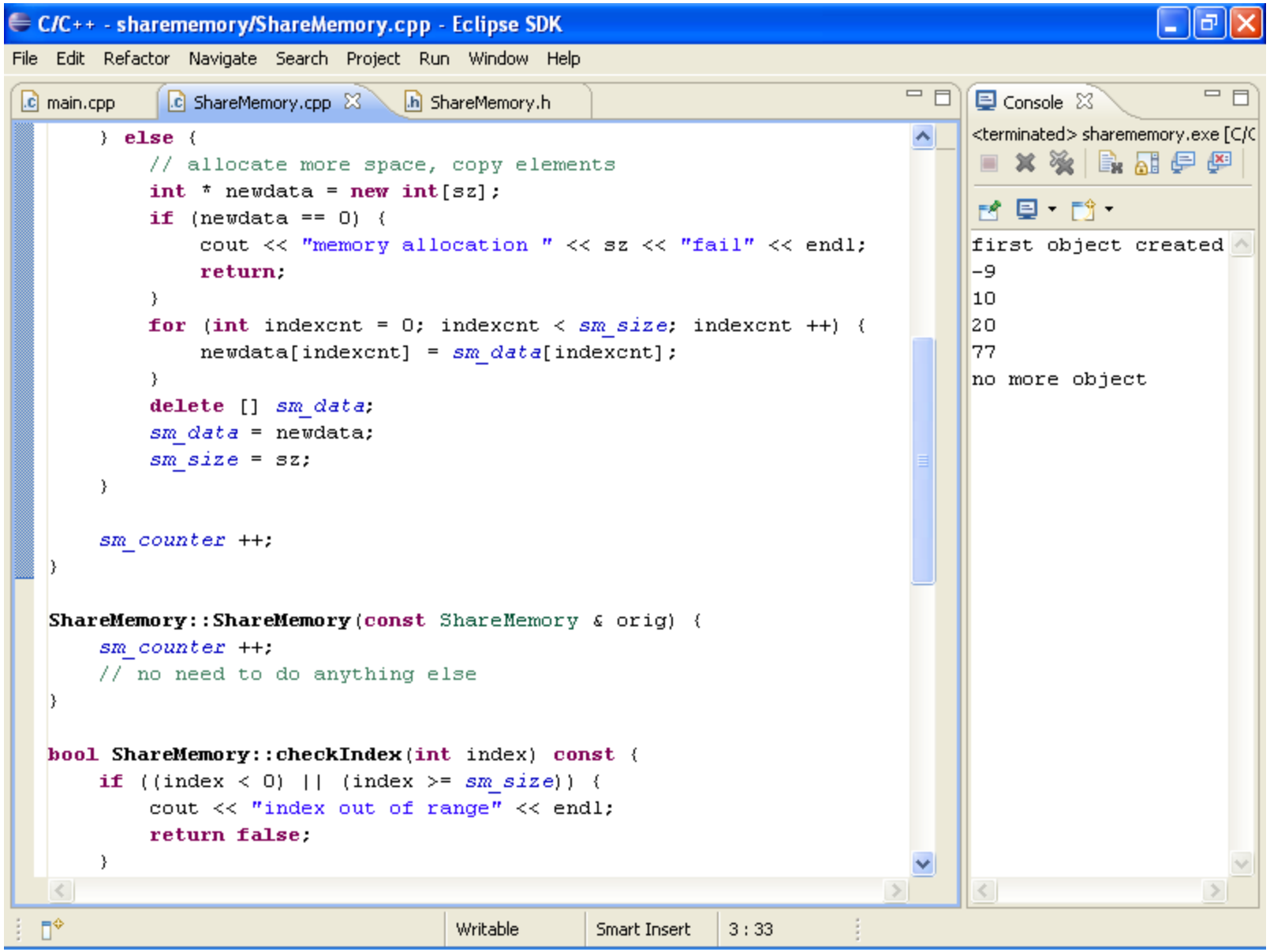
Memory Sharing

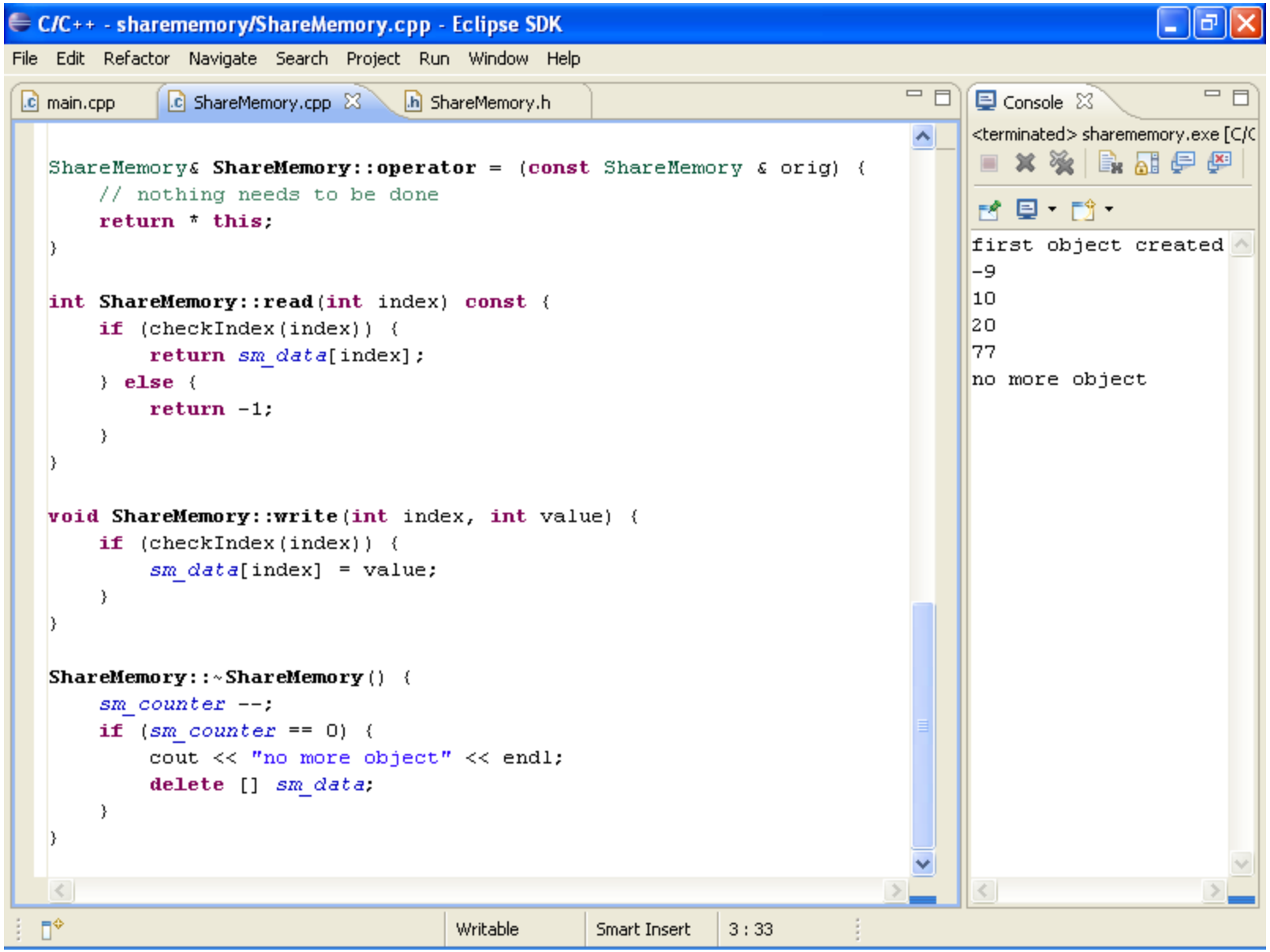
- Copy constructor + assignment operator perform "deep copy": allocate memory for individual objects so that they have different pieces of memory.
- Sometimes, it is preferred to share the memory among objects. This can be achieved by using static members using a "reference counter".
- The reference counter increments in constructor and decrements in destructor. When the counter is zero, delete the memory.

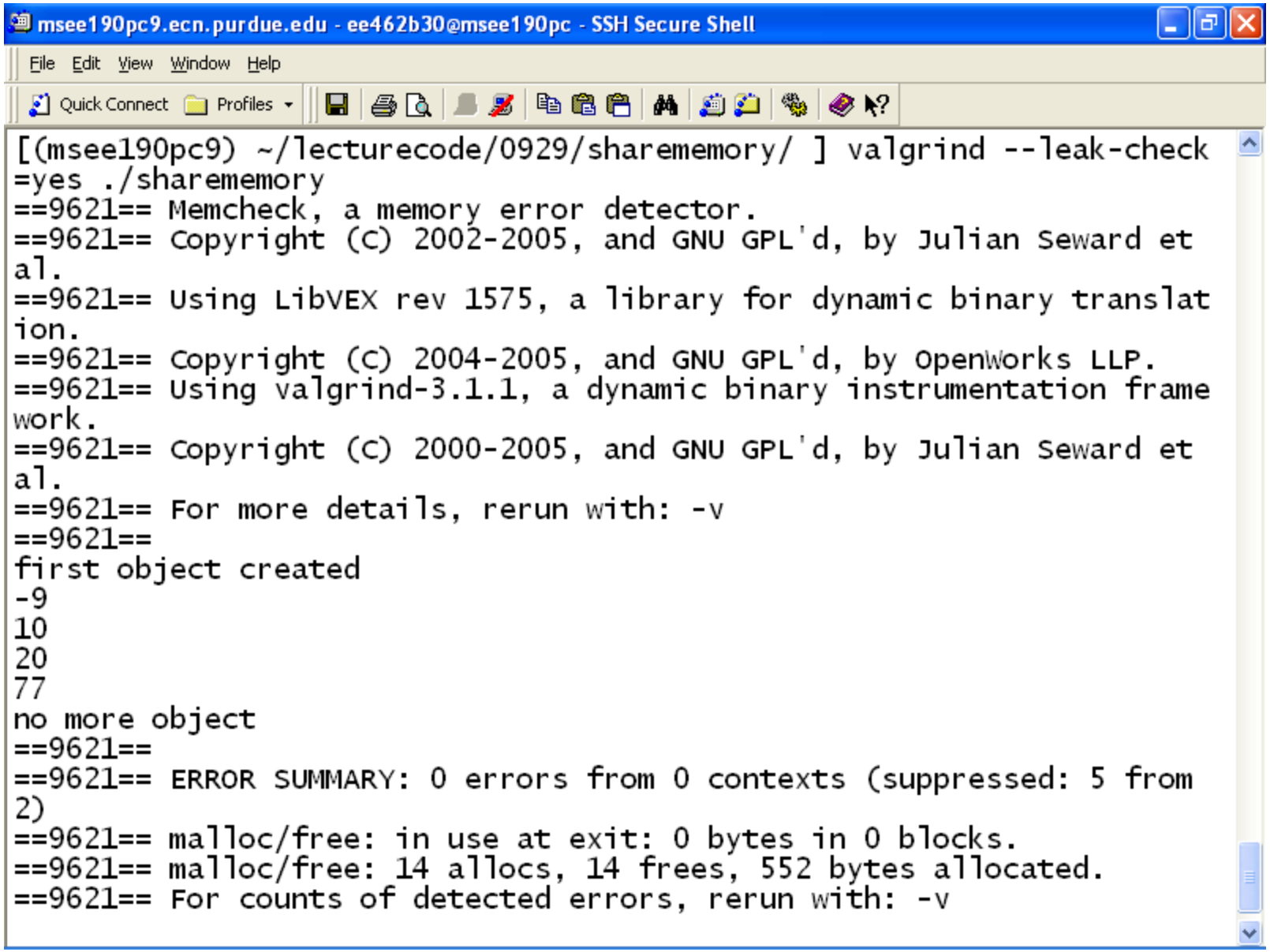

C/C++ - sharememory/main.cpp - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
main.cpp
ShareMemory.cpp
ShareMemory.h
#include <iostream>
#include "ShareMemory.h"
using namespace std;
int main(int argc, char** argv) {
ShareMemory sm1(10);
for (int index = 0; index < 10; index ++) {
sm1.write(index, index);
}
ShareMemory sm2(6);
sm2.write(4, -9);
cout << sm1.read(4) << endl;
cout << sm2.getSize() << endl;
ShareMemory sm3(20);
cout << sm1.getSize() << endl;
sm1.write(16, 77);
cout << sm3.read(16) << endl;
return 0;
}
Console
<terminated> sharememory.exe [C/C++ Local Appl
first object created
-9
10
20
77
no more object
Writable
Smart Insert
20 : 1


C/C++ - sharememory/ShareMemory.h - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
main.cpp
ShareMemory.cpp
ShareMemory.h
#ifndef SHAREMEMROY_H_
#define SHAREMEMROY_H_
#include <iostream>
using namespace std;
class ShareMemory {
static int sm_counter;
static int sm_size;
static int * sm_data;
bool checkIndex(int index) const;
public:
ShareMemory(int sz);
ShareMemory(const ShareMemory &);
ShareMemory& operator =(const ShareMemory &);
int getSize() const { return sm_size; }
int read(int index) const; // read
void write(int index, int value);
virtual ~ShareMemory();
};
#endif /*SHAREMEMROY_H_*/
Console
<terminated> sharememory.exe [C/C++ Local Appl
first object created
-9
10
20
77
no more object
Writable
Smart Insert
14 : 44


C/C++ - sharememory/ShareMemory.cpp - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
main.cpp
ShareMemory.cpp
ShareMemory.h
#include "ShareMemory.h"
// initialize static members
int ShareMemory::sm_counter = 0;
int ShareMemory::sm_size = 0;
int* ShareMemory::sm_data = 0;
ShareMemory::ShareMemory(int sz) {
if (sz <= 0) {
cout << "size < 0" << endl;
return;
}
if (sm_counter == 0) {
cout << "first object created" << endl;
sm_size = sz;
sm_data = new int[sz];
if (sm_data == 0) {
cout << "memory allocation " << sz << "fail" << endl;
return;
}
}
if (sm_size >= sz) {
// do nothing
} else {
// allocate more space, copy elements
int * newdata = new int[sz];
if (newdata == 0) {
cout << "memory allocation " << sz << "fail" << endl;
return;
}
Console
<terminated> sharememory.exe [C/C
first object created
-9
10
20
77
no more object
Writable
Smart Insert
3 : 33


C/C++ - sharememory/ShareMemory.cpp - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
main.cpp
ShareMemory.cpp
ShareMemory.h
} else {
// allocate more space, copy elements
int * newdata = new int[sz];
if (newdata == 0) {
cout << "memory allocation " << sz << "fail" << endl;
return;
}
for (int indexcnt = 0; indexcnt < sm_size; indexcnt ++) {
newdata[indexcnt] = sm_data[indexcnt];
}
delete [] sm_data;
sm_data = newdata;
sm_size = sz;
}
sm_counter ++;
}
ShareMemory::ShareMemory(const ShareMemory & orig) {
sm_counter ++;
// no need to do anything else
}
bool ShareMemory::checkIndex(int index) const {
if ((index < 0) || (index >= sm_size)) {
cout << "index out of range" << endl;
return false;
}
Console
<terminated> sharememory.exe [C/C
first object created
-9
10
20
77
no more object
Writable
Smart Insert
3 : 33


C/C++ - sharememory/ShareMemory.cpp - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
main.cpp
ShareMemory.cpp
ShareMemory.h
ShareMemory& ShareMemory::operator = (const ShareMemory & orig) {
// nothing needs to be done
return * this;
}
int ShareMemory::read(int index) const {
if (checkIndex(index)) {
return sm_data[index];
} else {
return -1;
}
}
void ShareMemory::write(int index, int value) {
if (checkIndex(index)) {
sm_data[index] = value;
}
}
ShareMemory::~ShareMemory() {
sm_counter --;
if (sm_counter == 0) {
cout << "no more object" << endl;
delete [] sm_data;
}
}
Console
<terminated> sharememory.exe [C/C
first object created
-9
10
20
77
no more object
Writable
Smart Insert
3 : 33


msee190pc9.ecn.purdue.edu - ee462b30@msee190pc - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
[(msee190pc9) ~/lecturecode/0929/sharememory/] valgrind --leak-check
=yes ./sharememory
==9621== Memcheck, a memory error detector.
==9621== Copyright (C) 2002-2005, and GNU GPL'd, by Julian Seward et
al.
==9621== Using LibVEX rev 1575, a library for dynamic binary translat
ion.
==9621== Copyright (C) 2004-2005, and GNU GPL'd, by OpenWorks LLP.
==9621== Using valgrind-3.1.1, a dynamic binary instrumentation frame
work.
==9621== Copyright (C) 2000-2005, and GNU GPL'd, by Julian Seward et
al.
==9621== For more details, rerun with: -v
==9621==
first object created
-9
10
20
77
no more object
==9621==
==9621== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 5 from
2)
==9621== malloc/free: in use at exit: 0 bytes in 0 blocks.
==9621== malloc/free: 14 allocs, 14 frees, 552 bytes allocated.
==9621== For counts of detected errors, rerun with: -v

Self Test