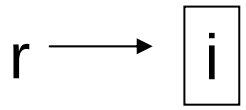
ECE 462
Object-Oriented Programming using C++ and Java

Reference in C++

Yung-Hsiang Lu
yunglu@purdue.edu

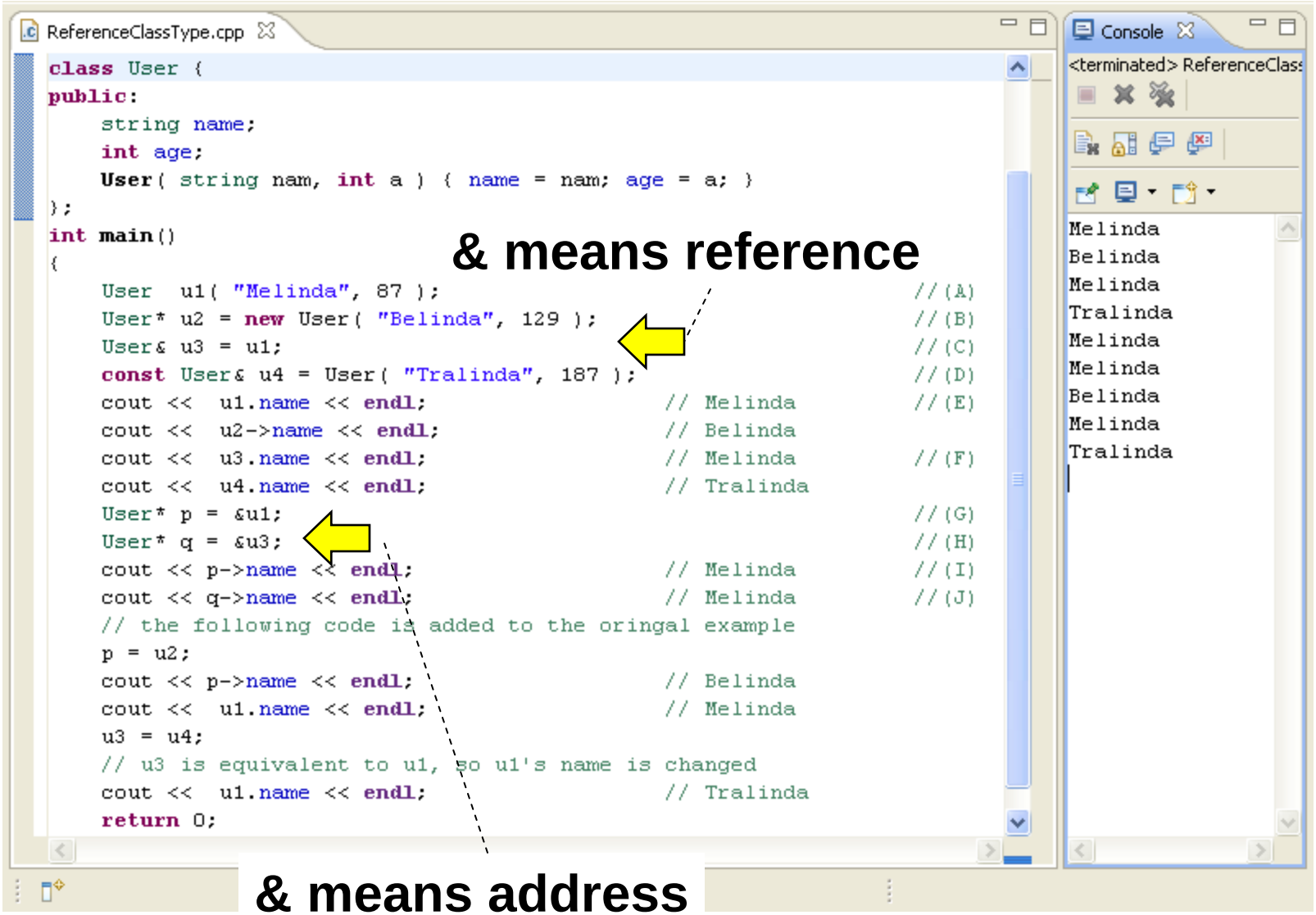
Reference (alias) in C++

- Reference is alias.

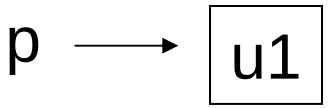


  - int i = 2;
  - int & r = i;
  - r = 3; ⇒ i is 3 now.
- Reference is similar to a pointer but reference, once assigned, **cannot be reassigned**.
  - r ++; ⇒ i is 4 now. // different from pointer arithmetics
  - int j = 100;
  - r = j; ⇒ **i** is 100

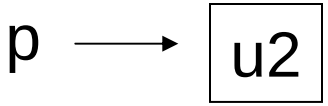
modified from ReferenceClassType.cc in the textbook



User* p = &u1;



p = u2;



(u1 is not affected.)

Self Test

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

Parameter Passing in Functions

Yung-Hsiang Lu
yunglu@purdue.edu

Parameter Passing in C++ and Java

Java	C++

pass by value, primitive types (int, char, double ...)
func(int x) { x = new_value; }
// change **not** seen by caller
	same
	func(int x) { x = 0; }
	y = 6;
	func(y); // calling
	y is still 6

none	pass by pointer, primitive type
	func(int * x) { *x = -1; }
	func(& y); // calling
	// change **seen** by caller (y is -1)

none	pass by reference, primitive type
	func(int & x) { x = 94; }
	func(y); // calling
	// change **seen** by caller (y is 94)

Java	C++

same syntax
but change **seen** by caller
the behavior, however, is similar to “pointer”
	pass by value, object
	AClass x;
	func(AClass obj) { obj.att = new_value; }
	func(x); // calling
	// object **copied** before passing
	// change **not** seen by caller

none	pass by pointer, object
	func(AClass * obj) { obj **->**att = new_value; }
	func(& x); // calling
	// change **seen** by caller

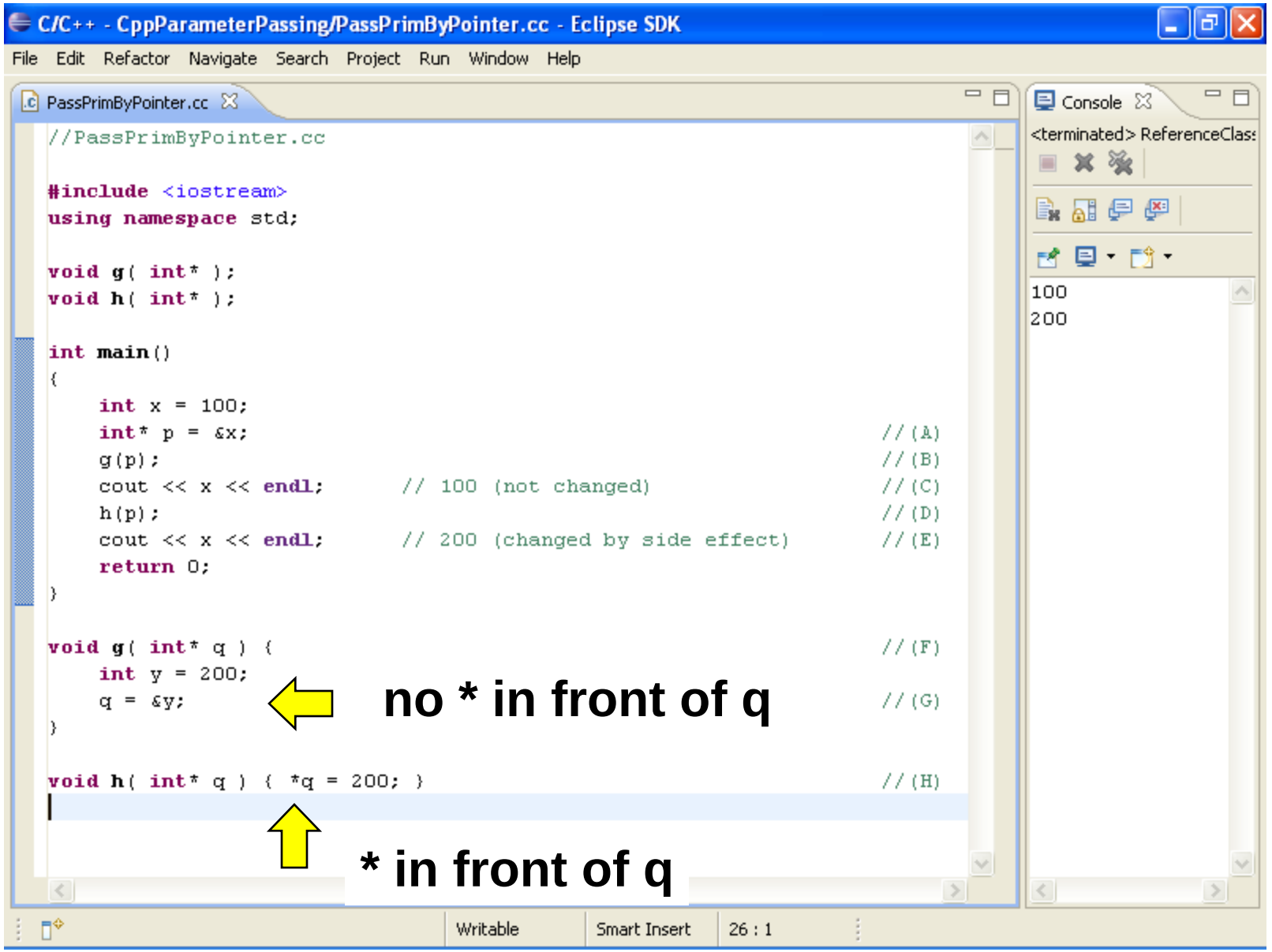
none	pass by reference, object
	func(AClass & obj)
	{ obj.att = new_value; }
	// change **seen** by caller
	// reference = alias

Pass by Value

```
//PassPrimByValue.cc
#include <iostream>
using namespace std;
void g( int );
int main()
{
    int x = 100;
    g(x);
    cout << x << endl; // 100
    return 0;
}
void g( int y )
{ y++; } // change will not been seen
```

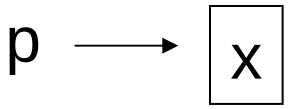
```
//PassPrimByValue.java
class Test
{
    public static void main( String[] args )
    {
        int x = 100;        //(A)
        g(x);               //(B)
        System.out.println( x ); // outputs 100
    }
    static void g( int y ) { y++; } //(C)
}
```

Pass by Pointer

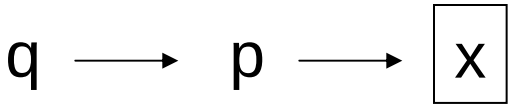

C/C++ - CppParameterPassing/PassPrimByPointer.cc - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
PassPrimByPointer.cc
//PassPrimByPointer.cc
#include <iostream>
using namespace std;
void g(int*);
void h(int*);
int main()
{
int x = 100;
int* p = &x; //(A)
g(p); //(B)
cout << x << endl; // 100 (not changed) //(C)
h(p); //(D)
cout << x << endl; // 200 (changed by side effect) //(E)
return 0;
}
void g(int* q) { //(F)
int y = 200;
q = &y; //(G)
}
void h(int* q) { *q = 200; } //(H)
no * in front of q
* in front of q
Console
<terminated> ReferenceClass
100
200
Writable
Smart Insert
26 : 1

Why doesn't g change x?

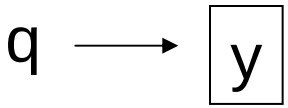
int* p = &x;



g(int* q)



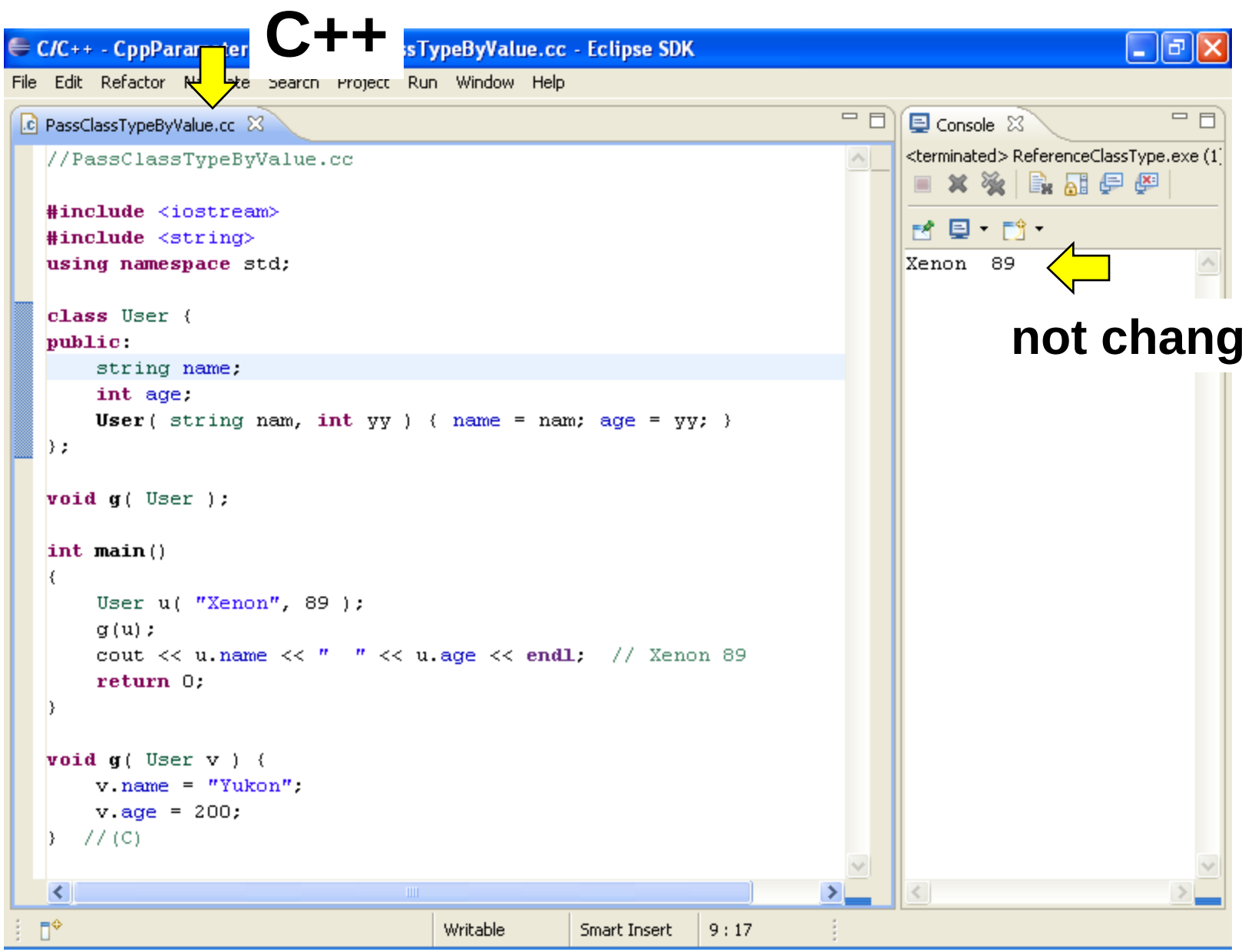
int y = 200;
q = &y;



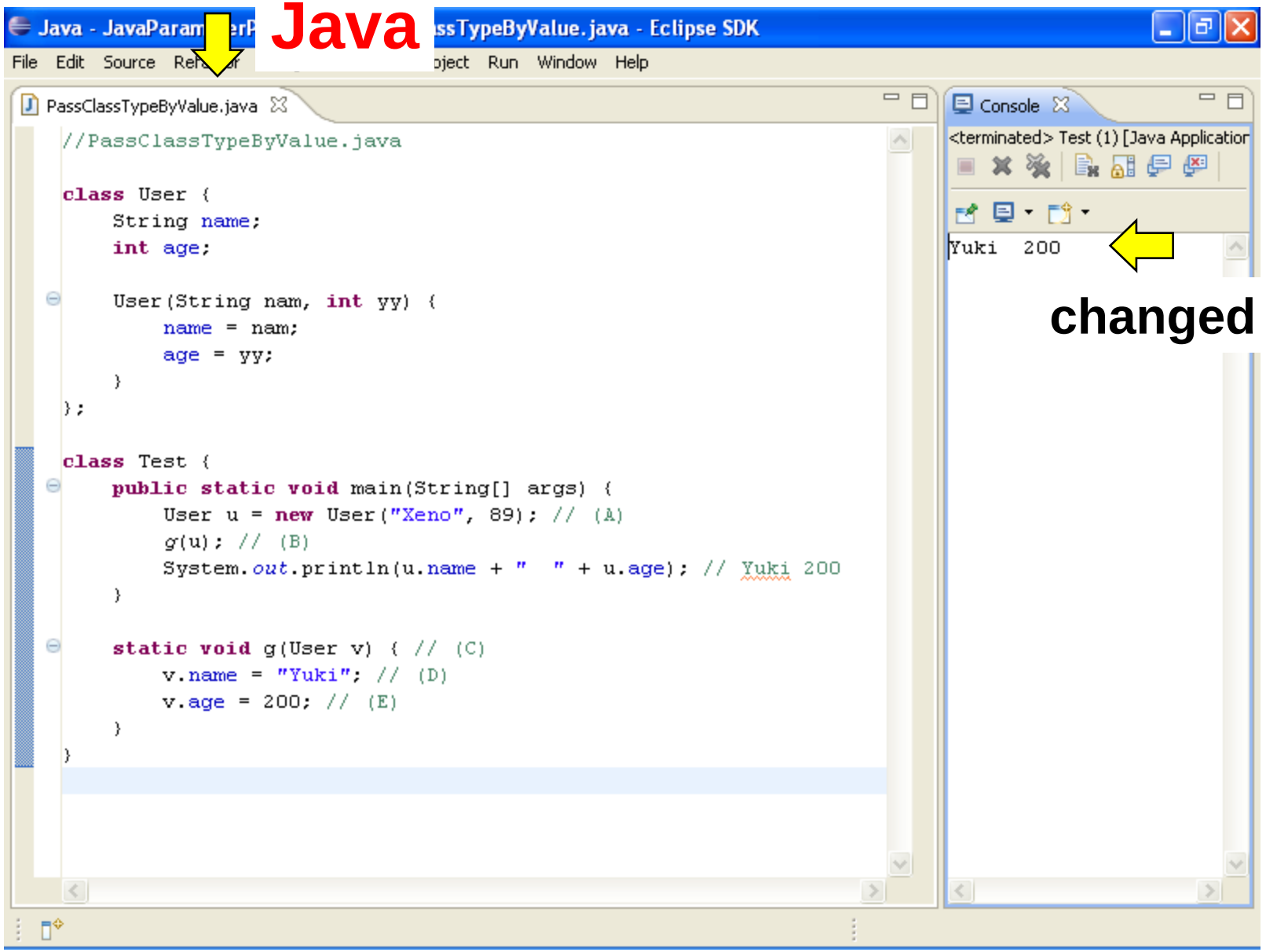
Changing q assigns the pointer to another location.
To change the content, we need to use * q (in function h).

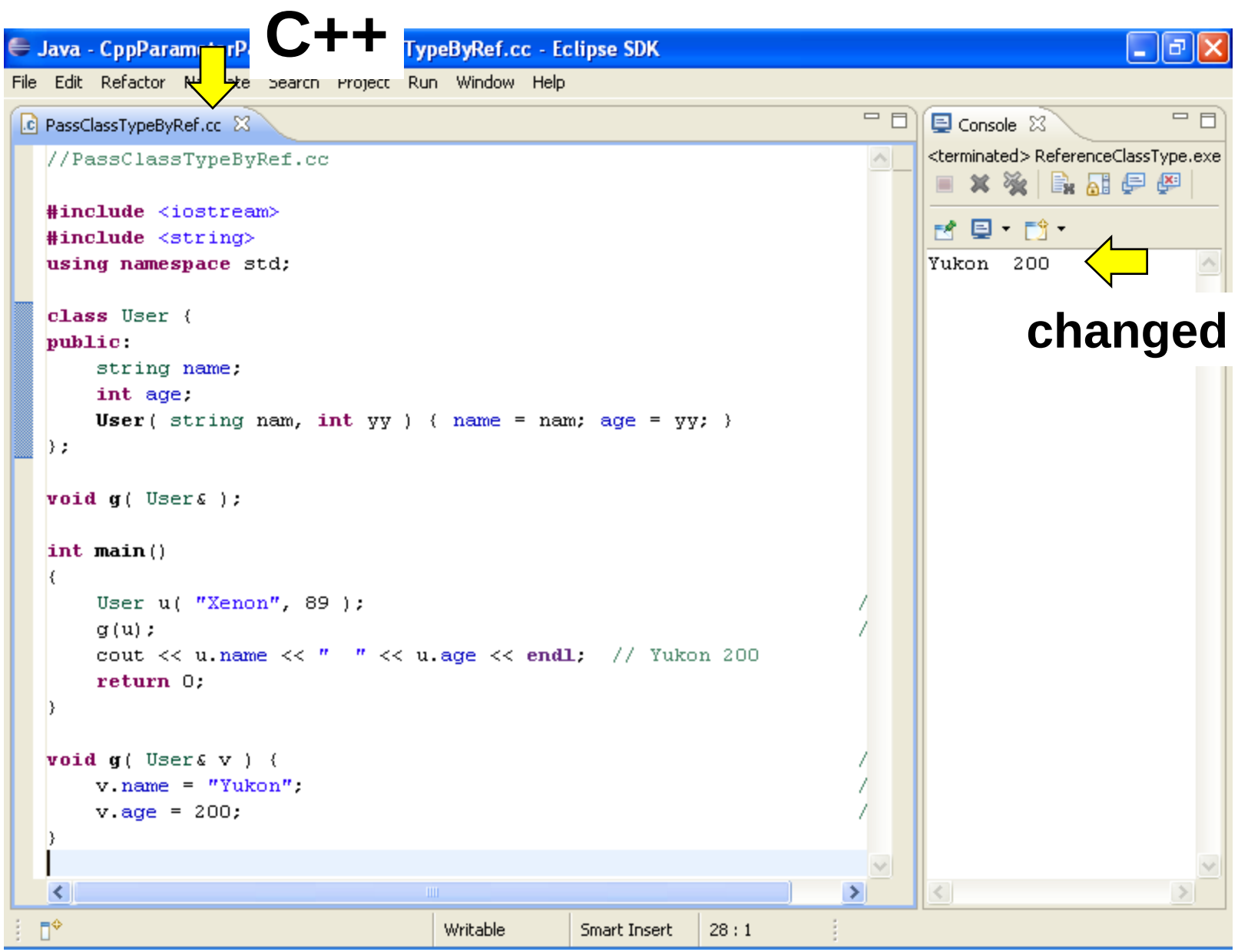
Pass by Reference

```
//PassPrimByRef.cc
#include <iostream>
using namespace std;
void g( int& );
int main()
{
    int x = 100;
    g(x);                                   //(A)
    cout << x << endl;          // 101      //(B)
    return 0;
}
void g( int& y ) { y++; }                   //(C)
```


C++
C/C++ - CppParameter
sTypeByValue.cc - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
PassClassTypeByValue.cc
//PassClassTypeByValue.cc
#include <iostream>
#include <string>
using namespace std;
class User {
public:
string name;
int age;
User(string nam, int yy) { name = nam; age = yy; }
};
void g(User);
int main()
{
User u("Xenon", 89);
g(u);
cout << u.name << " " << u.age << endl; // Xenon 89
return 0;
}
void g(User v) {
v.name = "Yukon";
v.age = 200;
} //(C)
Console
<terminated> ReferenceClassType.exe (1
Xenon 89
Writable
Smart Insert
9 : 17

not changed


Java
Java - JavaParam erP ssTypeByValue.java - Eclipse SDK
File Edit Source Ref or oject Run Window Help
PassClassTypeByValue.java
//PassClassTypeByValue.java
class User {
String name;
int age;
User(String nam, int yy) {
name = nam;
age = yy;
}
};
class Test {
public static void main(String[] args) {
User u = new User("Xeno", 89); // (A)
g(u); // (B)
System.out.println(u.name + " " + u.age); // Yuki 200
}
static void g(User v) { // (C)
v.name = "Yuki"; // (D)
v.age = 200; // (E)
}
}
Console
<terminated> Test (1) [Java Applicatior
Yuki 200
changed


C++
Java - CppParam
TypeByRef.cc - Eclipse SDK
File Edit Refactor
Search Project Run Window Help
PassClassTypeByRef.cc
//PassClassTypeByRef.cc
#include <iostream>
#include <string>
using namespace std;
class User {
public:
string name;
int age;
User(string nam, int yy) { name = nam; age = yy; }
};
void g(User&);
int main()
{
User u("Xenon", 89);
g(u);
cout << u.name << " " << u.age << endl; // Yukon 200
return 0;
}
void g(User& v) {
v.name = "Yukon";
v.age = 200;
}
Console
<terminated> ReferenceClassType.exe
Yukon 200
changed
Writable
Smart Insert
28 : 1

Self Test

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

Copy Constructor

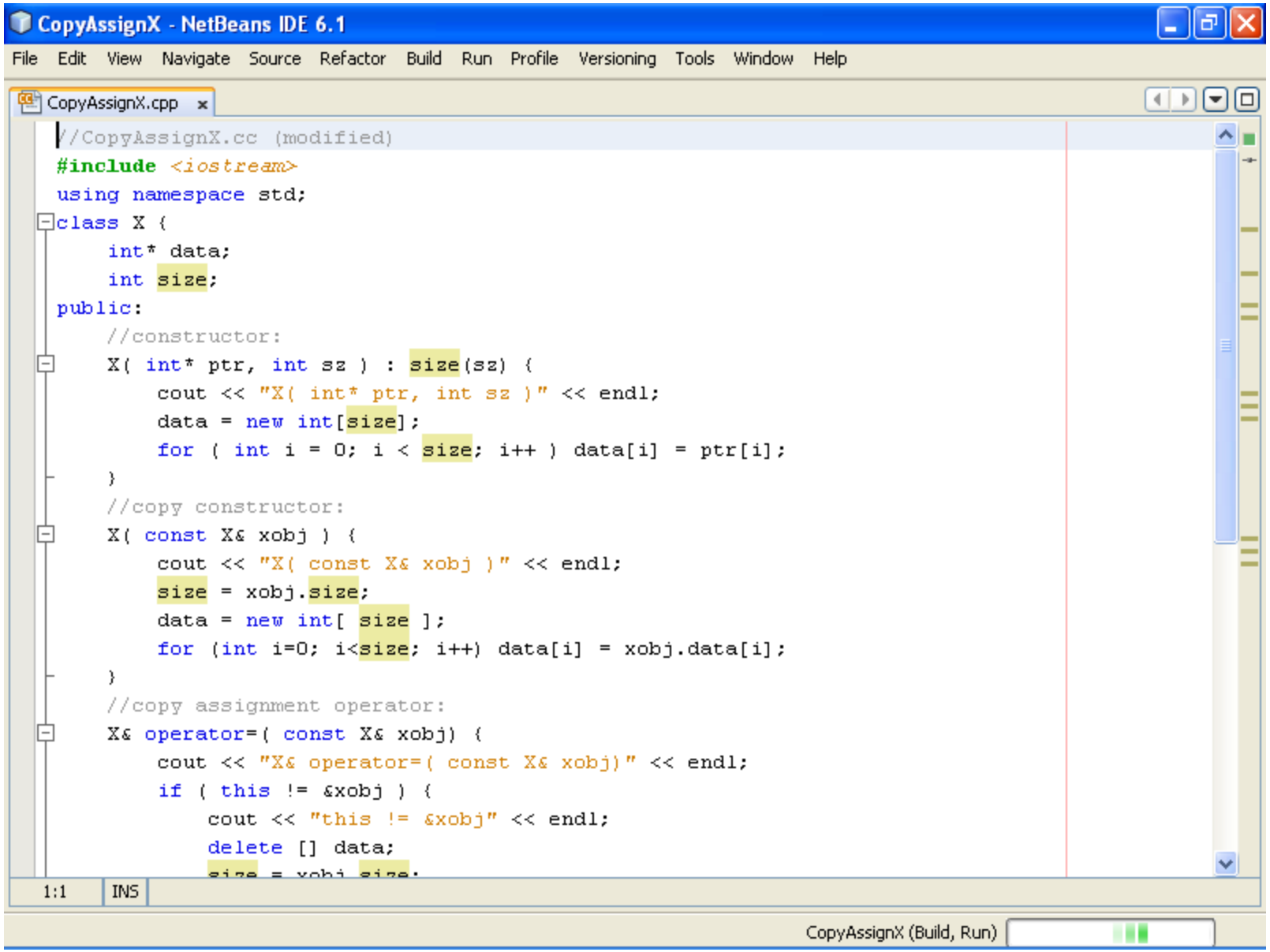
Yung-Hsiang Lu
yunglu@purdue.edu

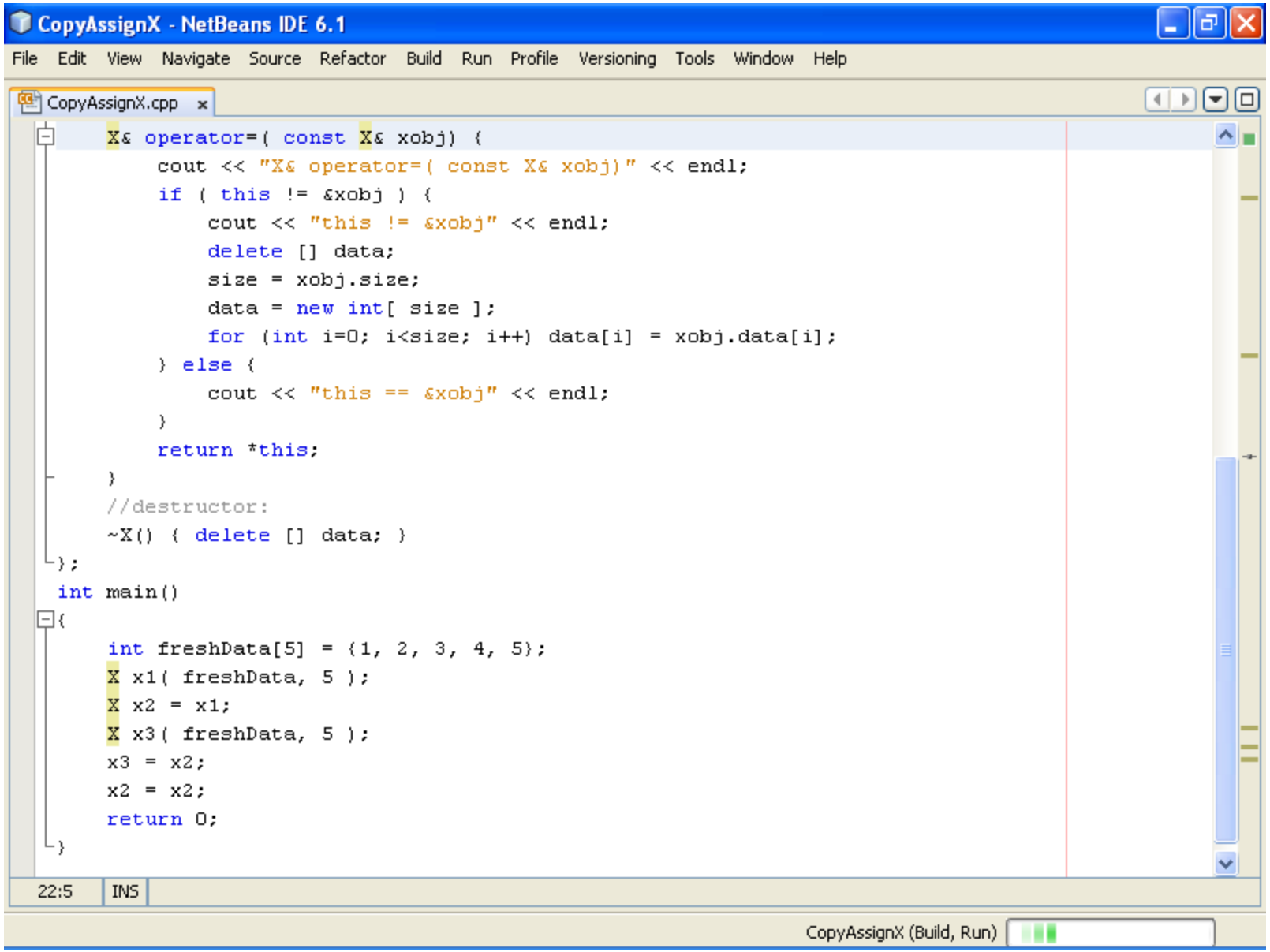
C++ Copy Constructor

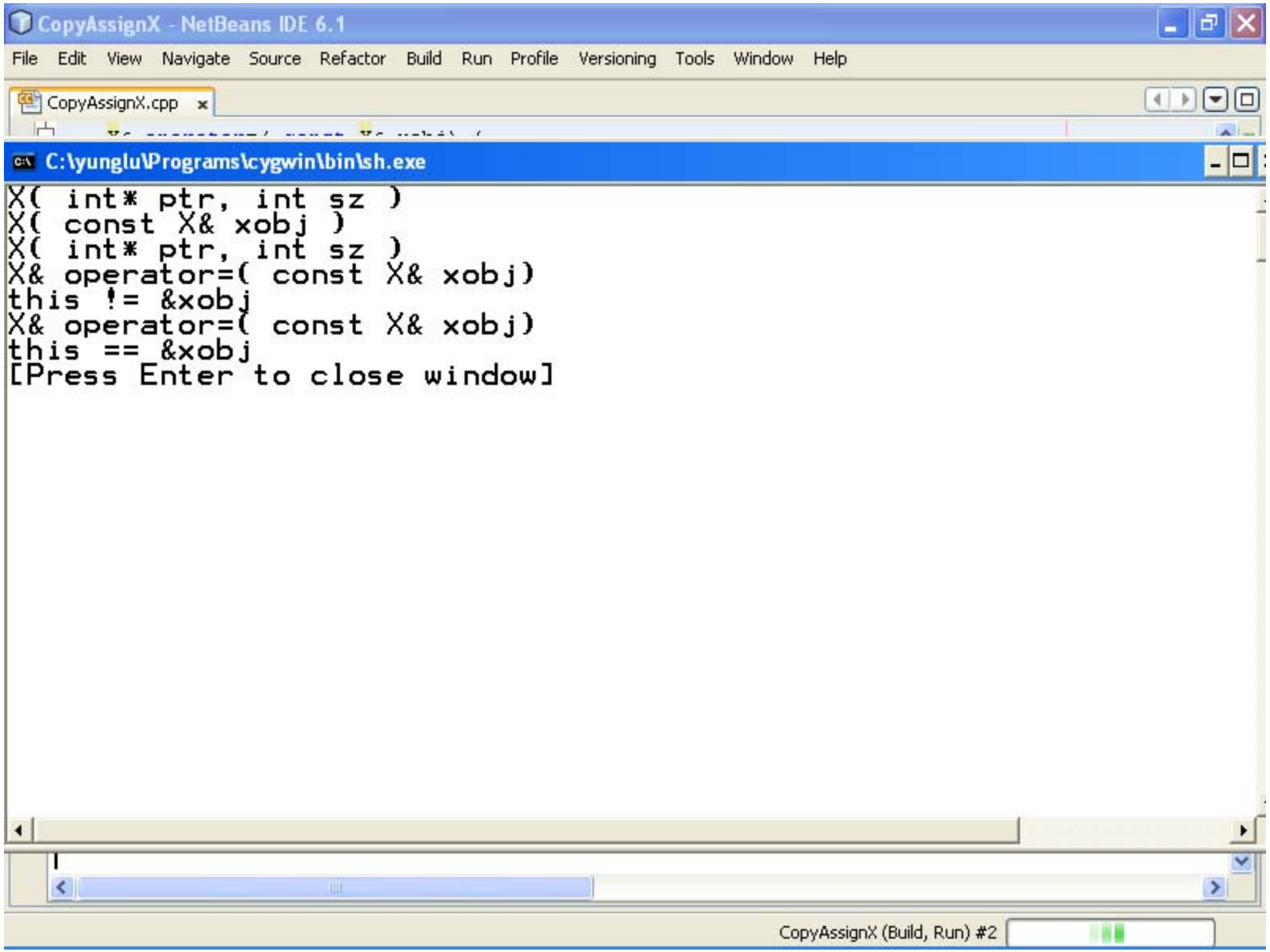
```
class NameOfClass
{
    NameOfClass(const NameOfClass & origobj)
    {
        // a constructor with special syntax
        // If a programmer does not provide a copy constructor
        // C++ compiler automatically creates one

        // allocate memory for attributes, if necessary
        // copy each attributes from origobj to this new object
    }
}
```

Assignment Operator =

```
class NameOfClass
{
    NameOfClass& operator = (const NameOfClass & origobj)
    {
        // C++ compiler automatically creates one, if not by a programmer
        if (this != & origobj)
        {
            // release memory, if necessary
            // allocate memory for attributes, if necessary
            // copy each attribute from origobj to this new object
        }
        return * this;
    }
}
```

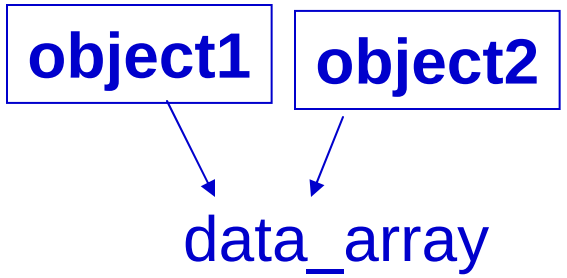

CopyAssignX - NetBeans IDE 6.1
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
CopyAssignX.cpp
//CopyAssignX.cc (modified)
#include <iostream>
using namespace std;
class X {
int* data;
int size;
public:
//constructor:
X(int* ptr, int sz) : size(sz) {
cout << "X(int* ptr, int sz)" << endl;
data = new int[size];
for (int i = 0; i < size; i++) data[i] = ptr[i];
}
//copy constructor:
X(const X& xobj) {
cout << "X(const X& xobj)" << endl;
size = xobj.size;
data = new int[size];
for (int i=0; i<size; i++) data[i] = xobj.data[i];
}
//copy assignment operator:
X& operator=(const X& xobj) {
cout << "X& operator=(const X& xobj)" << endl;
if (this != &xobj) {
cout << "this != &xobj" << endl;
delete [] data;
1:1
INS
CopyAssignX (Build, Run)

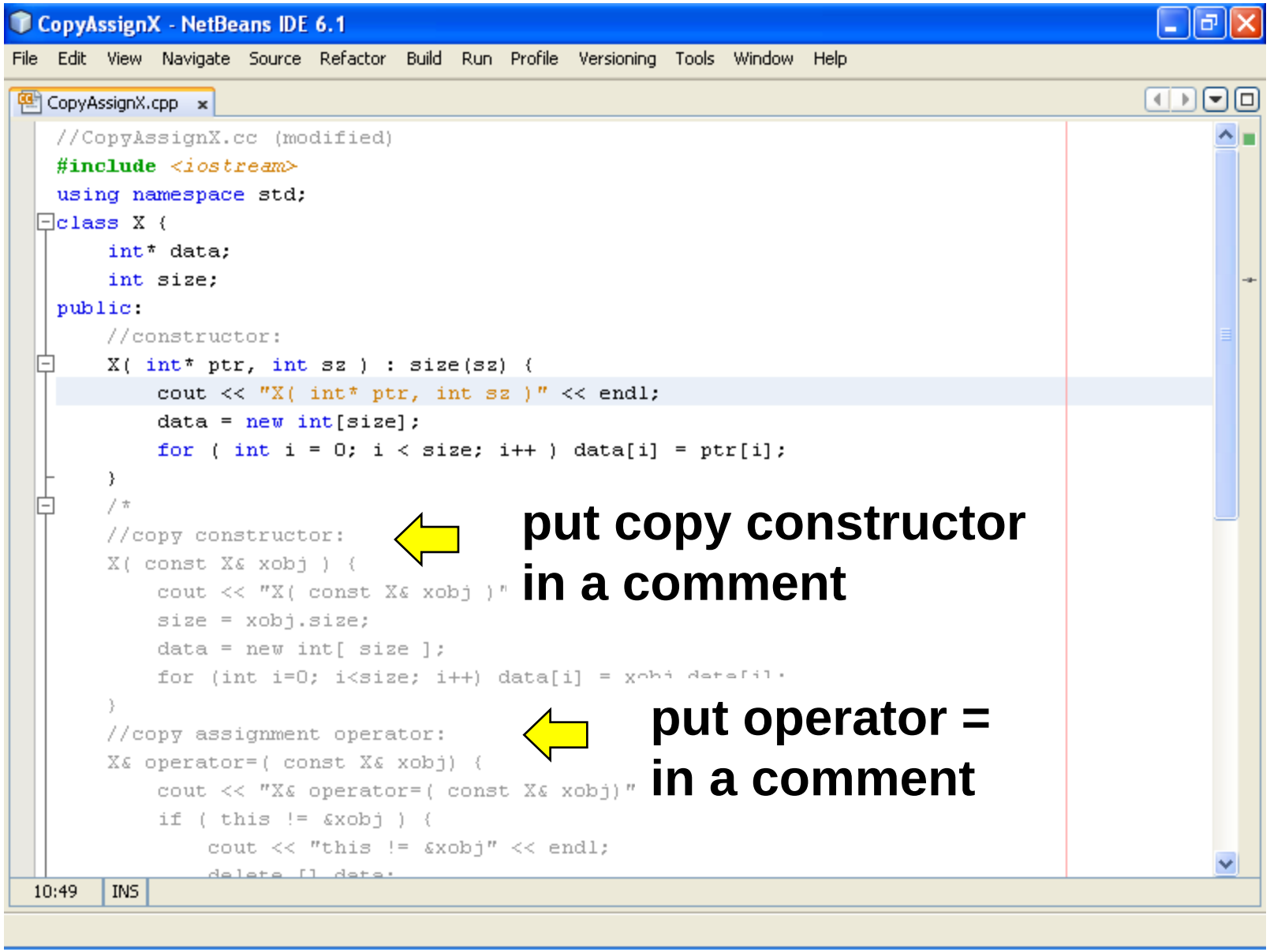

CopyAssignX - NetBeans IDE 6.1
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
CopyAssignX.cpp
X& operator=(const X& xobj) {
cout << "X& operator=(const X& xobj)" << endl;
if (this != &xobj) {
cout << "this != &xobj" << endl;
delete [] data;
size = xobj.size;
data = new int[size];
for (int i=0; i<size; i++) data[i] = xobj.data[i];
} else {
cout << "this == &xobj" << endl;
}
return *this;
}
//destructor:
~X() { delete [] data; }
};
int main()
{
int freshData[5] = {1, 2, 3, 4, 5};
X x1(freshData, 5);
X x2 = x1;
X x3(freshData, 5);
x3 = x2;
x2 = x2;
return 0;
}
22:5
INS
CopyAssignX (Build, Run)

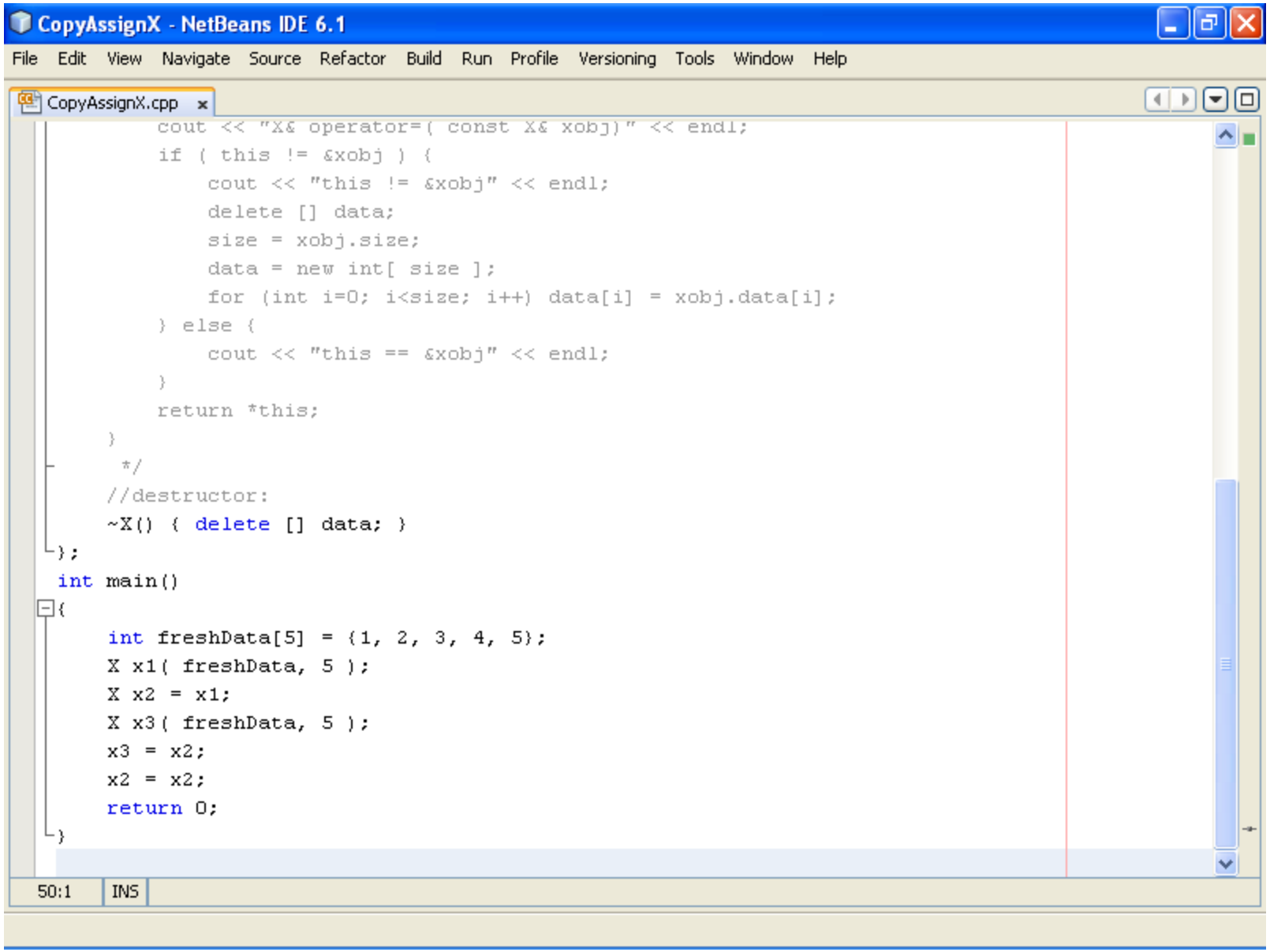

CopyAssignX - NetBeans IDE 6.1
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
CopyAssignX.cpp
C:\yunglu\Programs\cygwin\bin\sh.exe
X(int* ptr, int sz)
X(const X& xobj)
X(int* ptr, int sz)
X& operator=(const X& xobj)
this != &xobj
X& operator=(const X& xobj)
this == &xobj
[Press Enter to close window]
CopyAssignX (Build, Run) #2

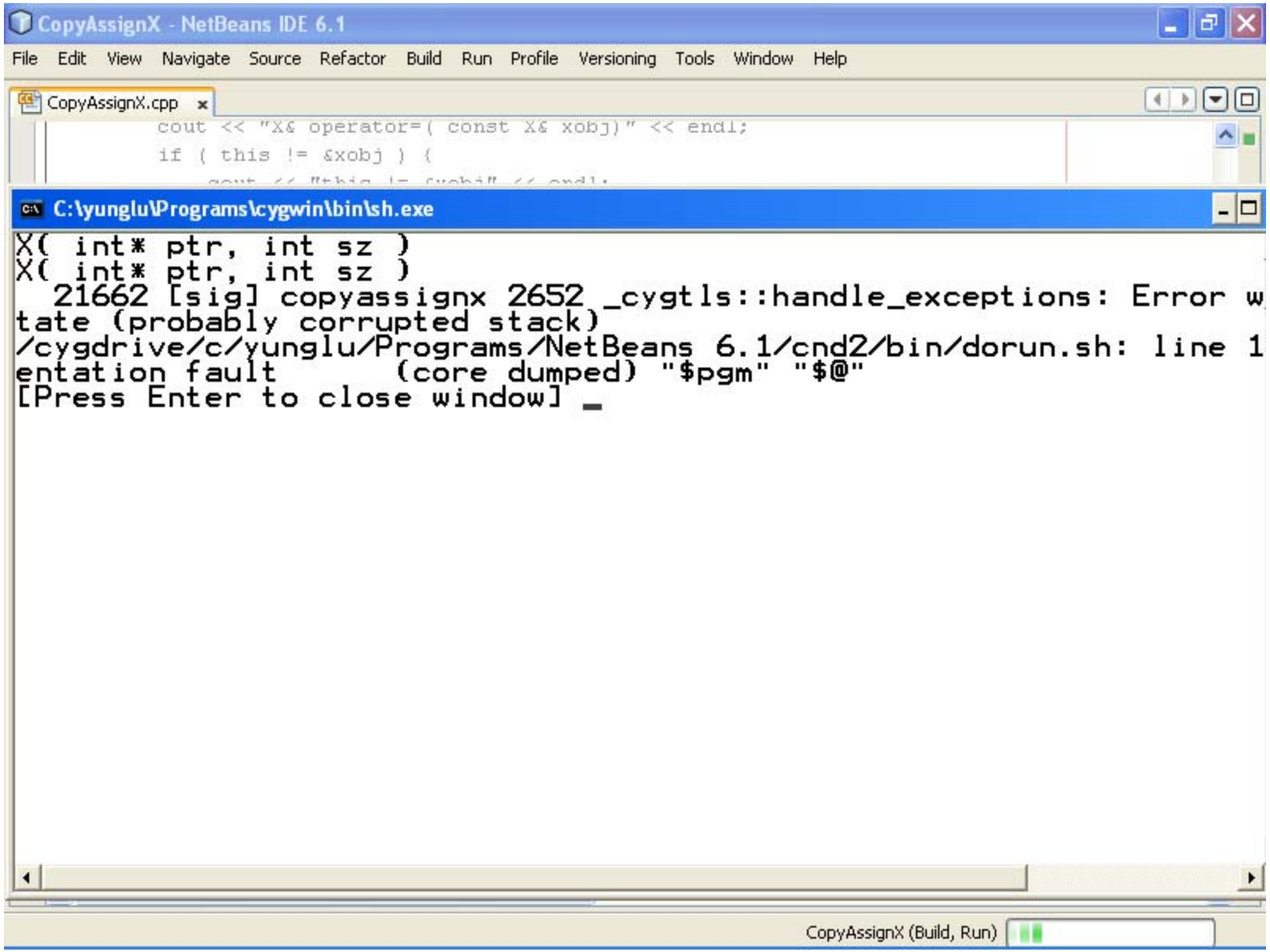
Shallow or Deep Copy

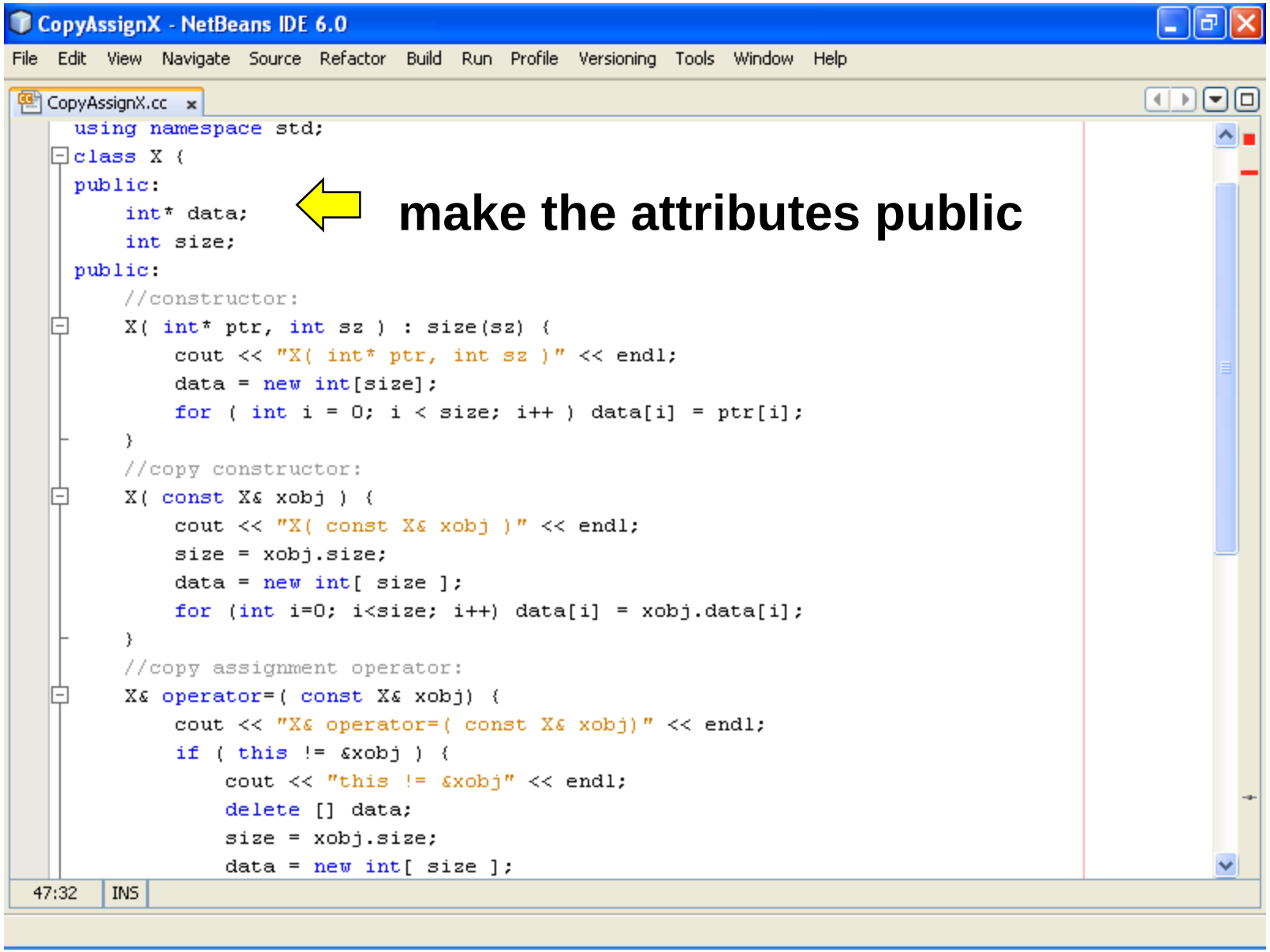
- If you do not provide a copy constructor (or an assignment operator), C++ will create one for you.
- The one created by the compiler copies the values of attributes.
- When an attribute is a pointer, two objects' attributes point to the same location.
- If one object releases the memory (for example, by the destructor), the second cannot access the memory any more.

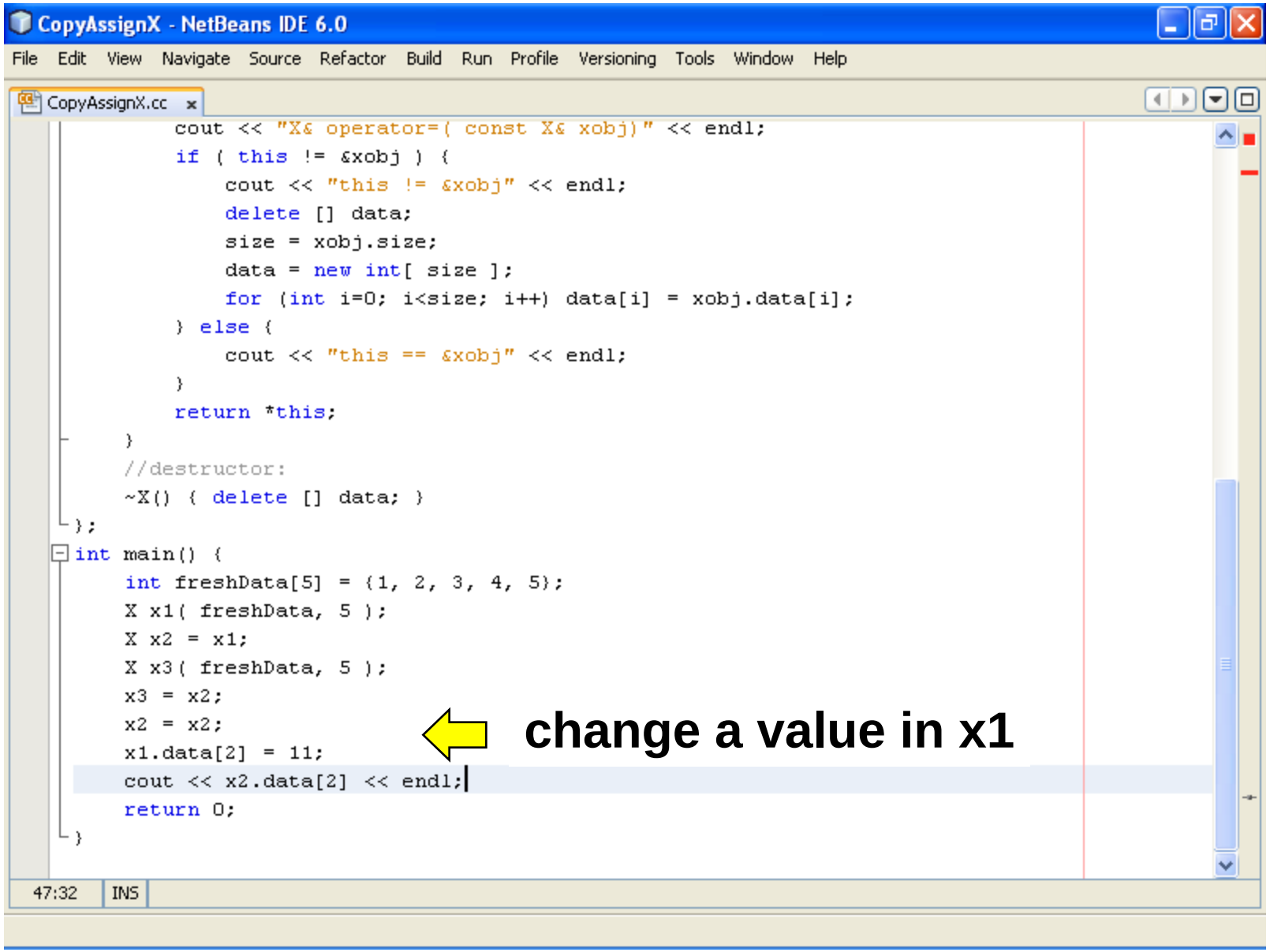


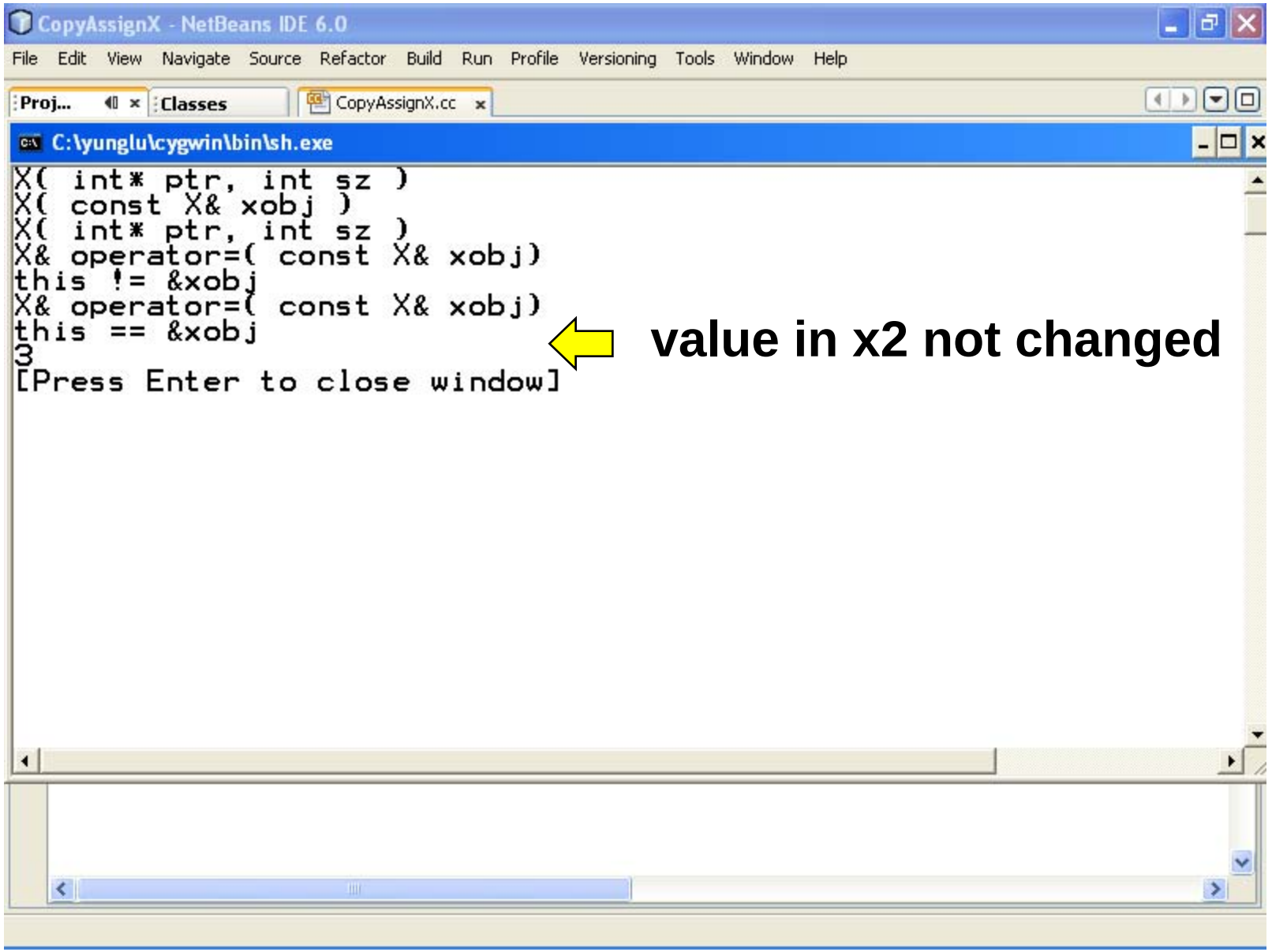

CopyAssignX - NetBeans IDE 6.1
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
CopyAssignX.cpp
//CopyAssignX.cc (modified)
#include <iostream>
using namespace std;
class X {
int* data;
int size;
public:
//constructor:
X(int* ptr, int sz) : size(sz) {
cout << "X(int* ptr, int sz)" << endl;
data = new int[size];
for (int i = 0; i < size; i++) data[i] = ptr[i];
}
/*
//copy constructor:
X(const X& xobj) {
cout << "X(const X& xobj)"
size = xobj.size;
data = new int[size];
for (int i=0; i<size; i++) data[i] = xobj.data[i];
}
//copy assignment operator:
X& operator=(const X& xobj) {
cout << "X& operator=(const X& xobj)"
if (this != &xobj) {
cout << "this != &xobj" << endl;
delete [] data;
10:49
INS
put copy constructor
in a comment
put operator =
in a comment

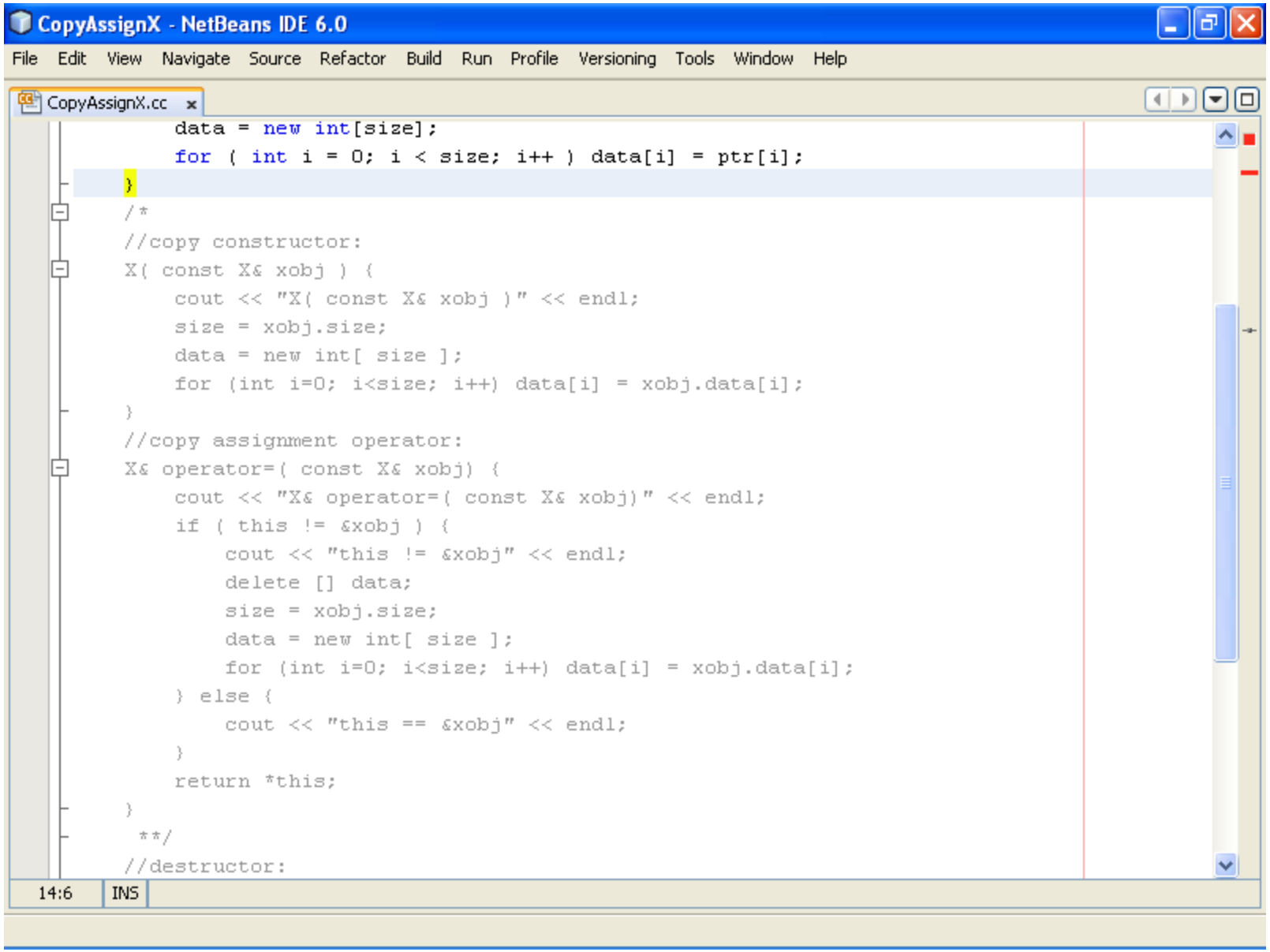

CopyAssignX - NetBeans IDE 6.1
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
CopyAssignX.cpp
cout << "X& operator=(const X& xobj)" << endl;
if (this != &xobj) {
cout << "this != &xobj" << endl;
delete [] data;
size = xobj.size;
data = new int[size];
for (int i=0; i<size; i++) data[i] = xobj.data[i];
} else {
cout << "this == &xobj" << endl;
}
return *this;
}
*/
//destructor:
~X() { delete [] data; }
};
int main()
{
int freshData[5] = {1, 2, 3, 4, 5};
X x1(freshData, 5);
X x2 = x1;
X x3(freshData, 5);
x3 = x2;
x2 = x2;
return 0;
}
50:1
INS

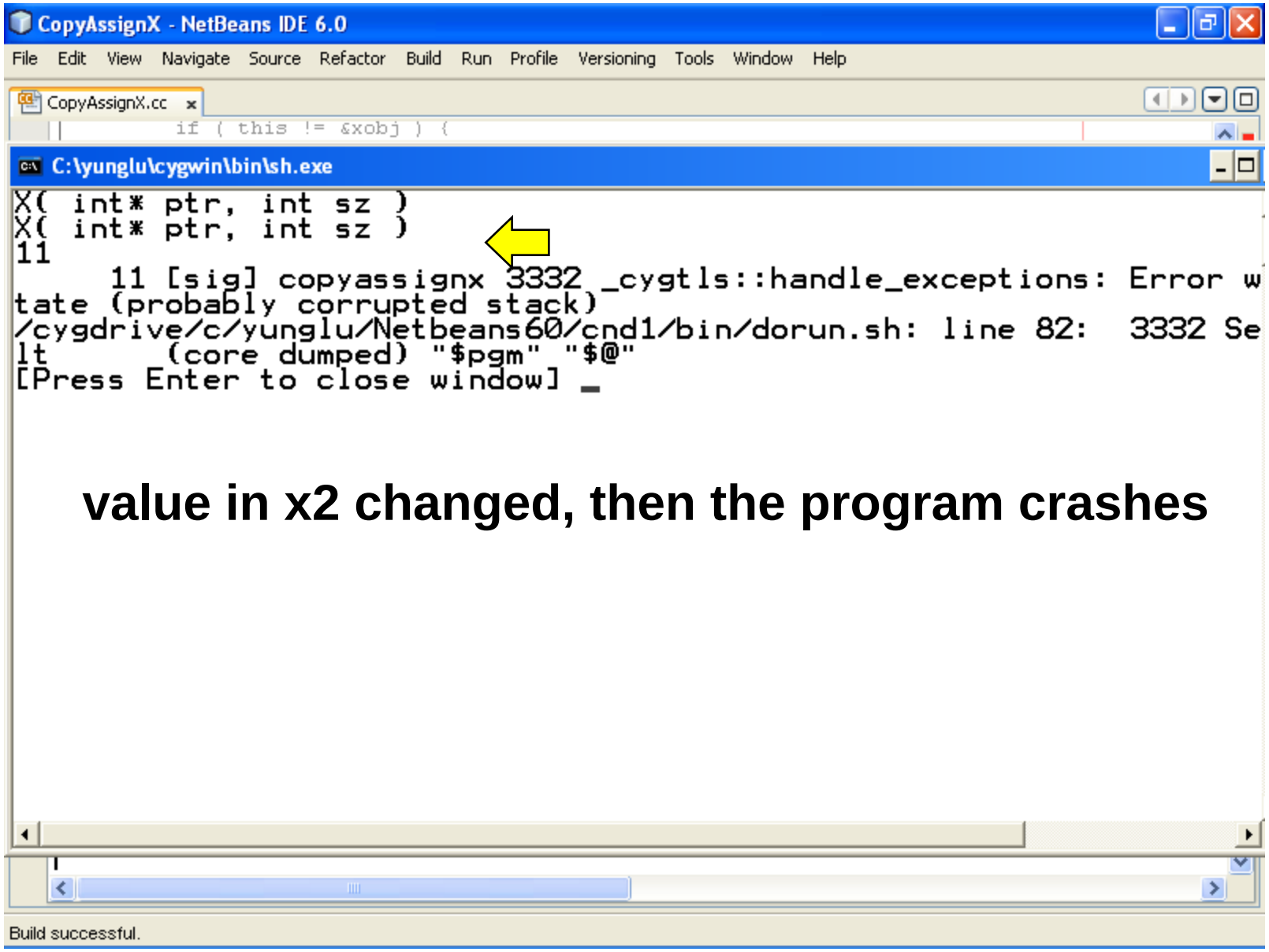

CopyAssignX - NetBeans IDE 6.1
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
CopyAssignX.cpp
cout << "X& operator=(const X& xobj)" << endl;
if (this != &xobj) {
C:\yunglu\Programs\cygwin\bin\sh.exe
X(int* ptr, int sz)
X(int* ptr, int sz)
21662 [sig] copyassignx 2652 _cygtls::handle_exceptions: Error w
tate (probably corrupted stack)
/cygdrive/c/yunglu/Programs/NetBeans 6.1/cnd2/bin/dorun.sh: line 1
entation fault (core dumped) "$pgm" "$@"
[Press Enter to close window]
CopyAssignX (Build, Run)


CopyAssignX - NetBeans IDE 6.0
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
CopyAssignX.cc
using namespace std;
class X {
public:
int* data;
int size;
public:
//constructor:
X(int* ptr, int sz) : size(sz) {
cout << "X(int* ptr, int sz)" << endl;
data = new int[size];
for (int i = 0; i < size; i++) data[i] = ptr[i];
}
//copy constructor:
X(const X& xobj) {
cout << "X(const X& xobj)" << endl;
size = xobj.size;
data = new int[size];
for (int i=0; i<size; i++) data[i] = xobj.data[i];
}
//copy assignment operator:
X& operator=(const X& xobj) {
cout << "X& operator=(const X& xobj)" << endl;
if (this != &xobj) {
cout << "this != &xobj" << endl;
delete [] data;
size = xobj.size;
data = new int[size];
47:32
INS
make the attributes public


CopyAssignX - NetBeans IDE 6.0
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
CopyAssignX.cc
cout << "X& operator=(const X& xobj)" << endl;
if (this != &xobj) {
cout << "this != &xobj" << endl;
delete [] data;
size = xobj.size;
data = new int[size];
for (int i=0; i<size; i++) data[i] = xobj.data[i];
} else {
cout << "this == &xobj" << endl;
}
return *this;
}
//destructor:
~X() { delete [] data; }
};
int main() {
int freshData[5] = {1, 2, 3, 4, 5};
X x1(freshData, 5);
X x2 = x1;
X x3(freshData, 5);
x3 = x2;
x2 = x2;
x1.data[2] = 11;
cout << x2.data[2] << endl;
return 0;
}
change a value in x1
47:32
INS


CopyAssignX - NetBeans IDE 6.0
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
Proj...
Classes
CopyAssignX.cc
C:\yunglu\cygwin\bin\sh.exe
X(int* ptr, int sz)
X(const X& xobj)
X(int* ptr, int sz)
X& operator=(const X& xobj)
this != &xobj
X& operator=(const X& xobj)
this == &xobj
3
[Press Enter to close window]
value in x2 not changed


CopyAssignX - NetBeans IDE 6.0
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
CopyAssignX.cc
data = new int[size];
for (int i = 0; i < size; i++) data[i] = ptr[i];
}
/*
//copy constructor:
X(const X& xobj) {
cout << "X(const X& xobj)" << endl;
size = xobj.size;
data = new int[size];
for (int i=0; i<size; i++) data[i] = xobj.data[i];
}
//copy assignment operator:
X& operator=(const X& xobj) {
cout << "X& operator=(const X& xobj)" << endl;
if (this != &xobj) {
cout << "this != &xobj" << endl;
delete [] data;
size = xobj.size;
data = new int[size];
for (int i=0; i<size; i++) data[i] = xobj.data[i];
} else {
cout << "this == &xobj" << endl;
}
return *this;
}
**/
//destructor:
14:6
INS



value in x2 changed, then the program crashes

How about Java?

- Java does not need (neither allow) copy constructor.
- Java does not allow programmer-defined operator overloading.
- Java performs garbage collection. Destructors are not needed (and not allowed).

Self Test

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

Parameter Passing in Functions II

Yung-Hsiang Lu
yunglu@purdue.edu

C++ Parameter Passing and Copying

pass by value, object

```
func(AClass obj) { ... }
// object copied before passing,  change not seen by caller
```

pass by pointer, object

```
func(AClass * x) { ... }
// object not copied, change seen by caller
```

pass by reference, object

```
func(AClass & obj)  { ... }
// object not copied, change seen by caller
```

Return Object and Copying

return object

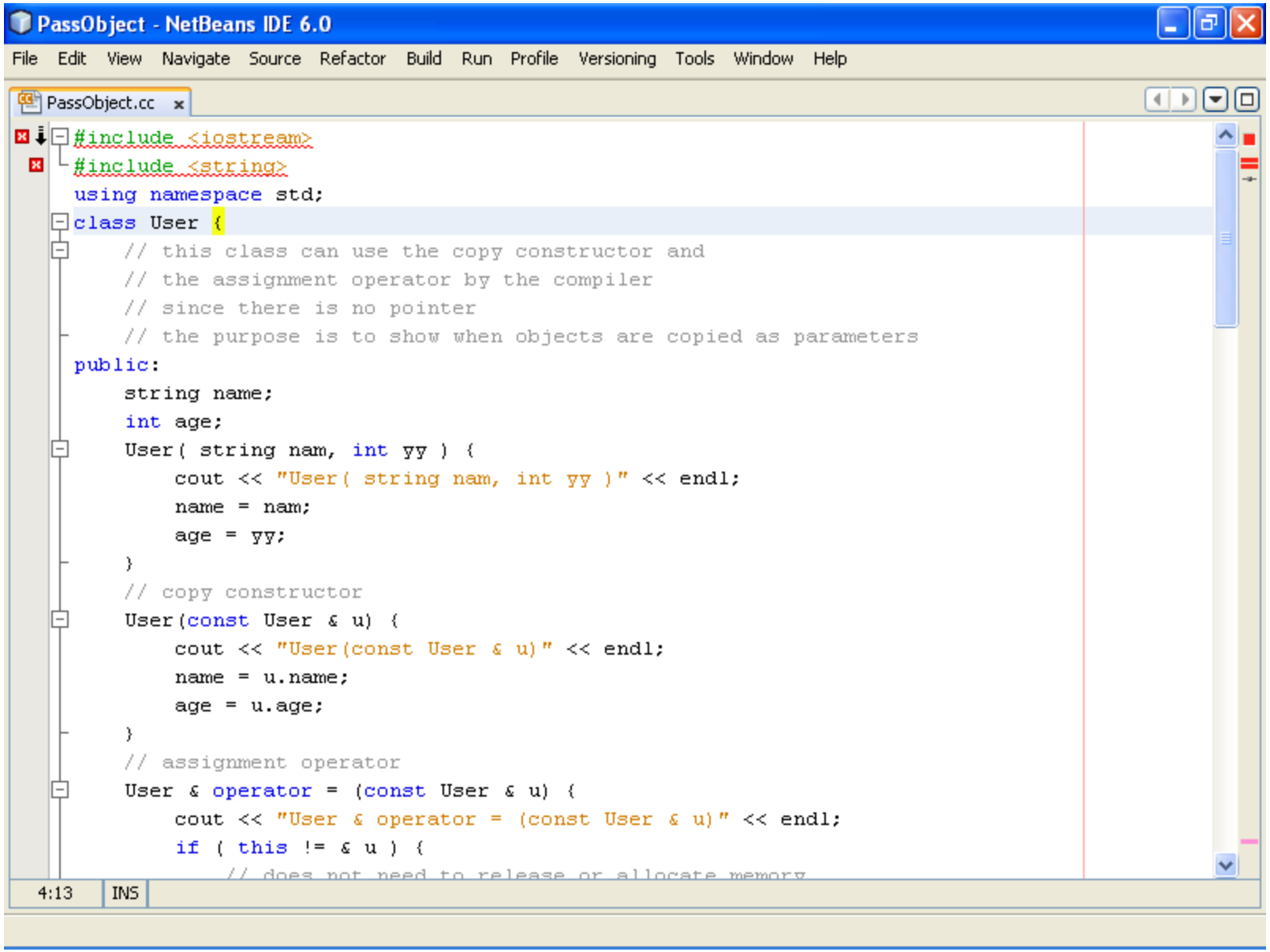
```
AClass func(...) { ... }
// object copied before return
```

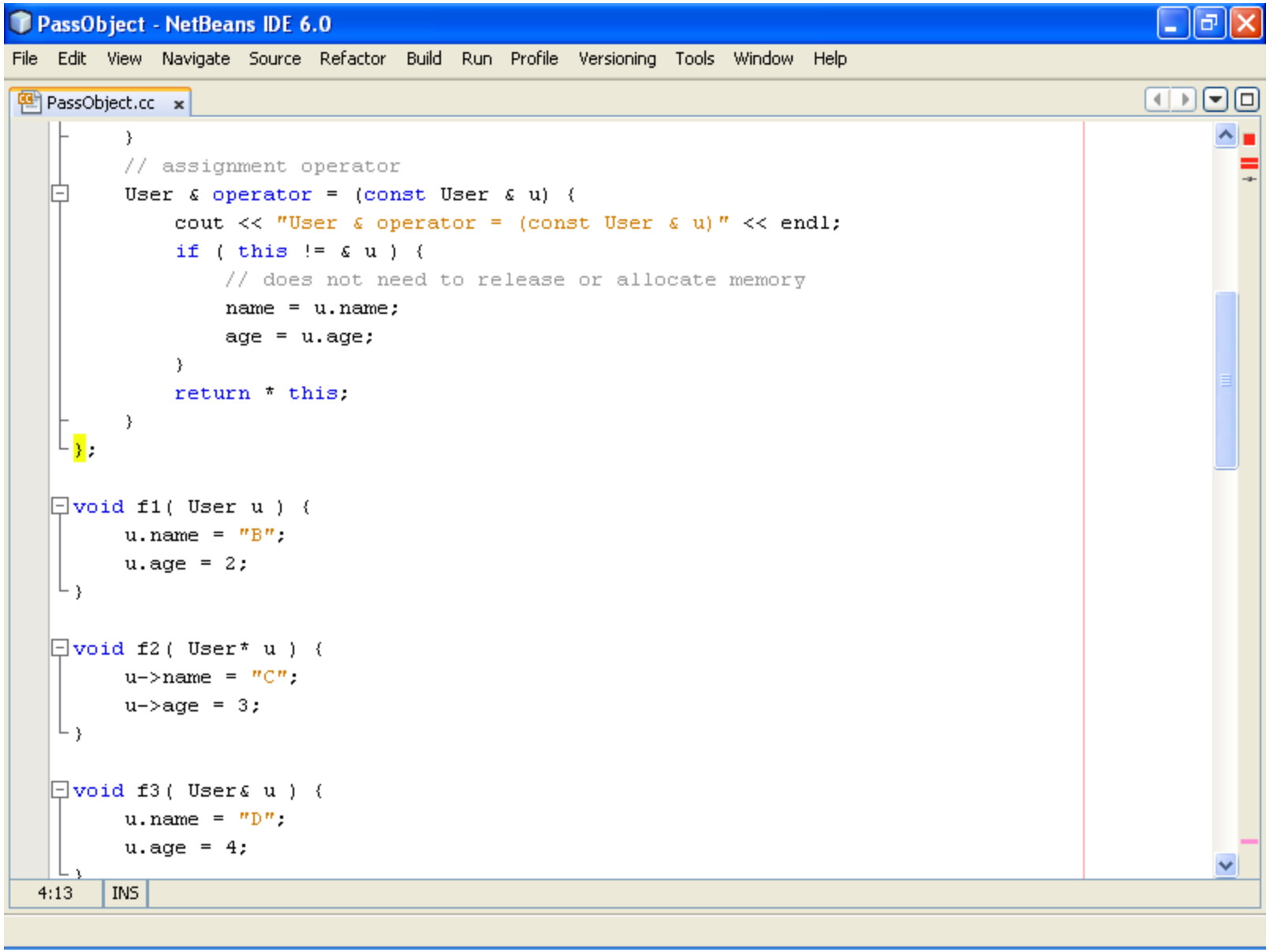
return pointer

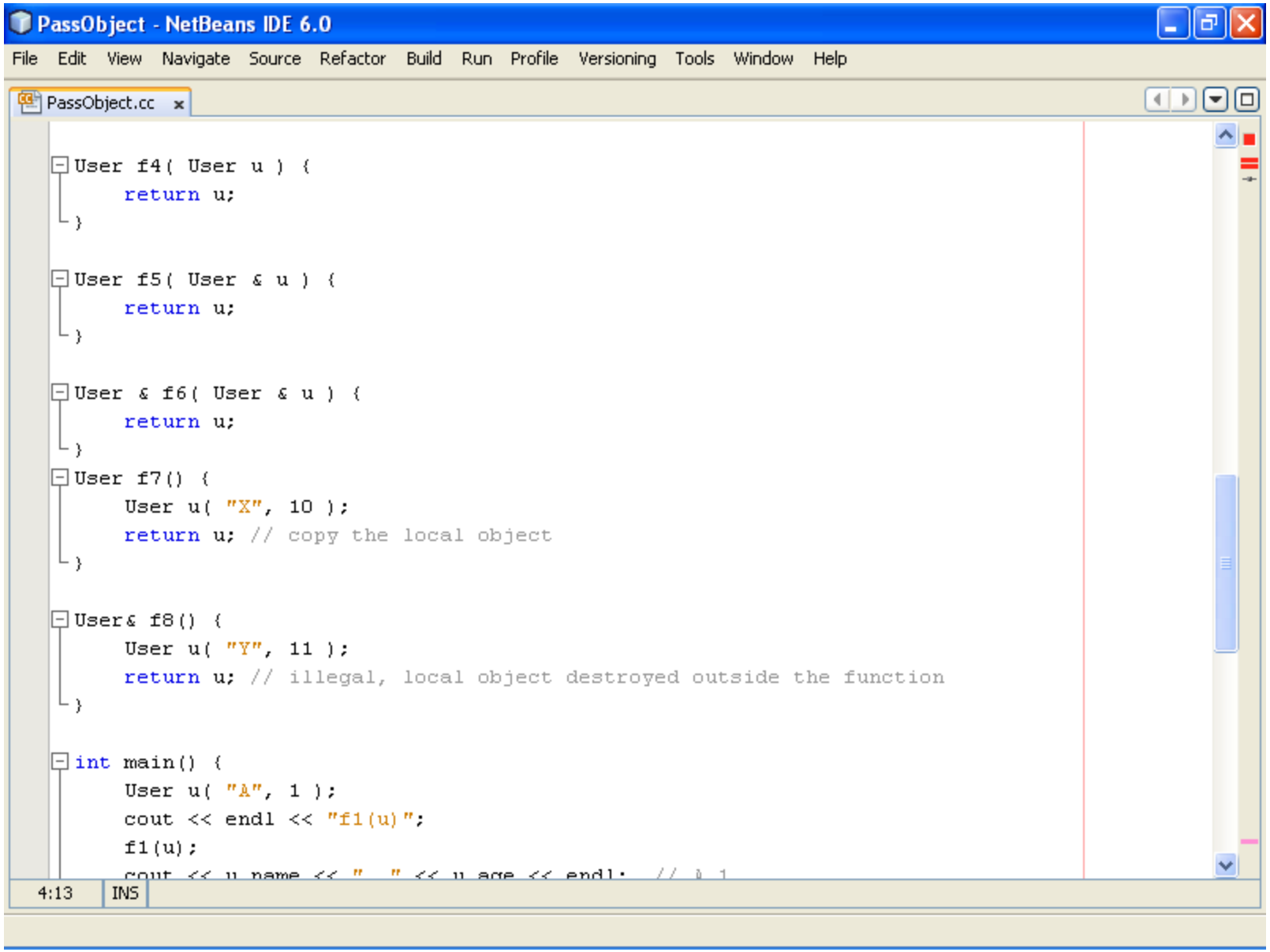
```
AClass * func(...) { ... }
// object not copied, cannot return a local object
```

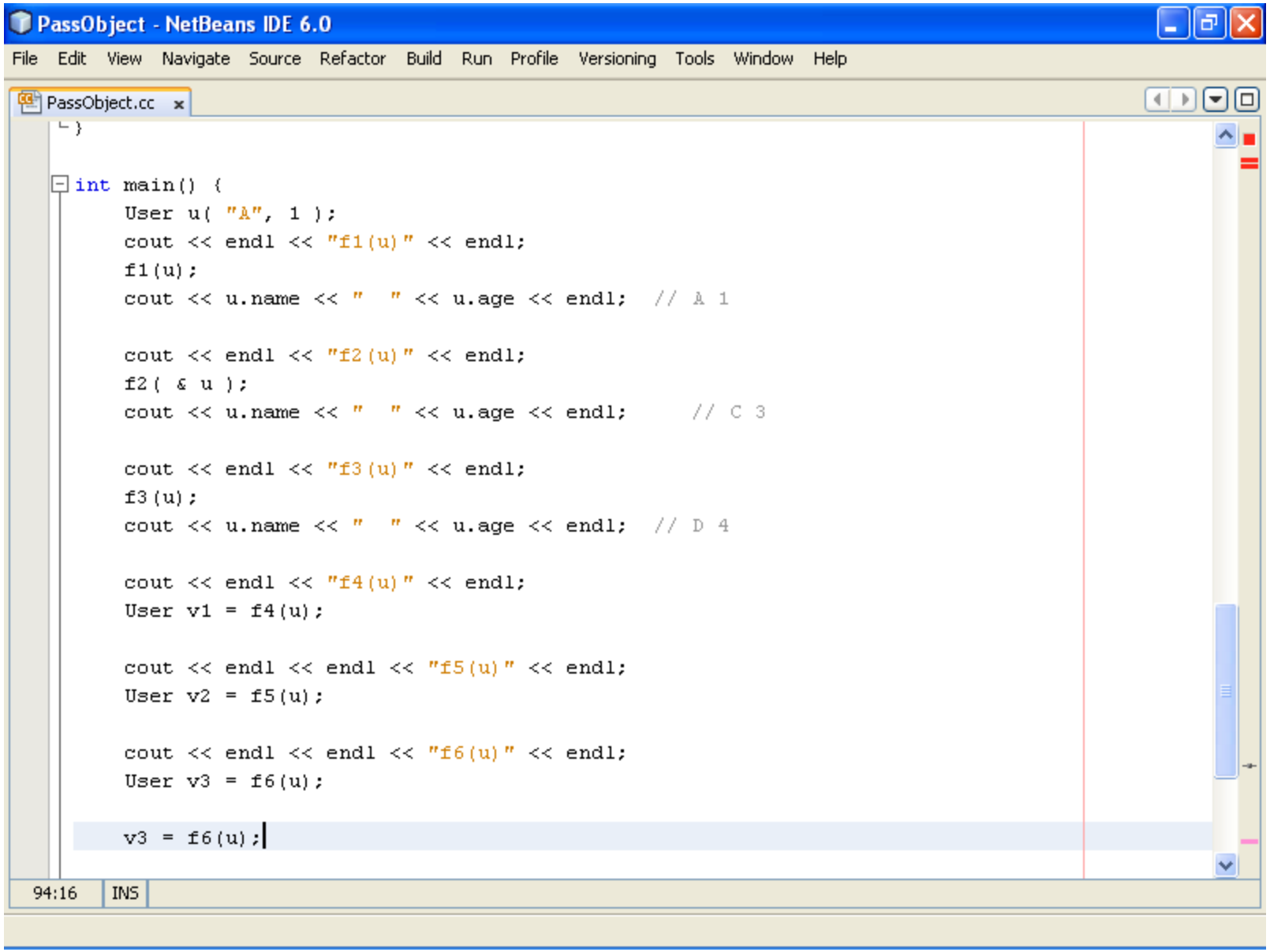
return reference

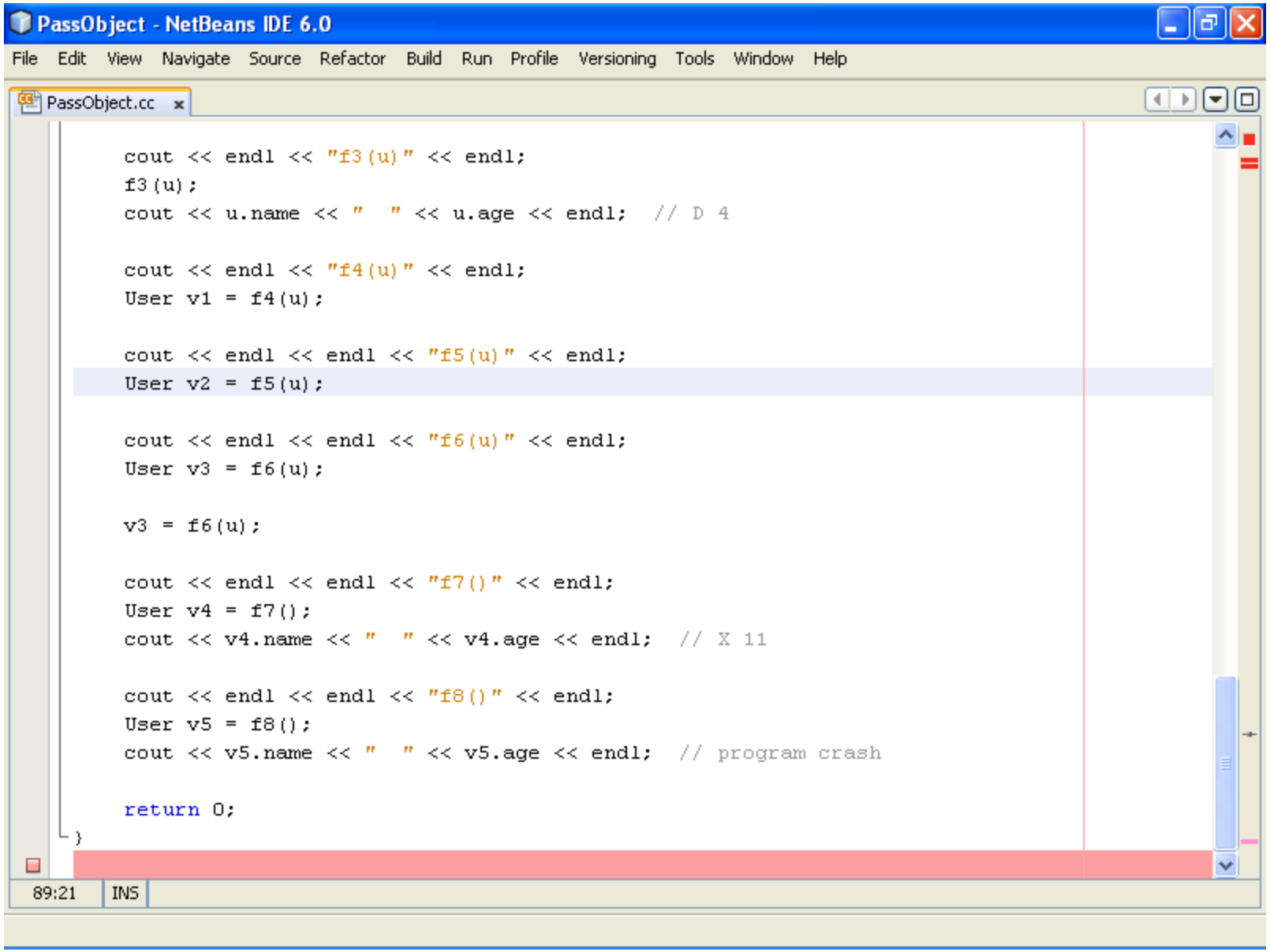
```
AClass & func(...) { ... }
// object not copied, cannot return a local object
```

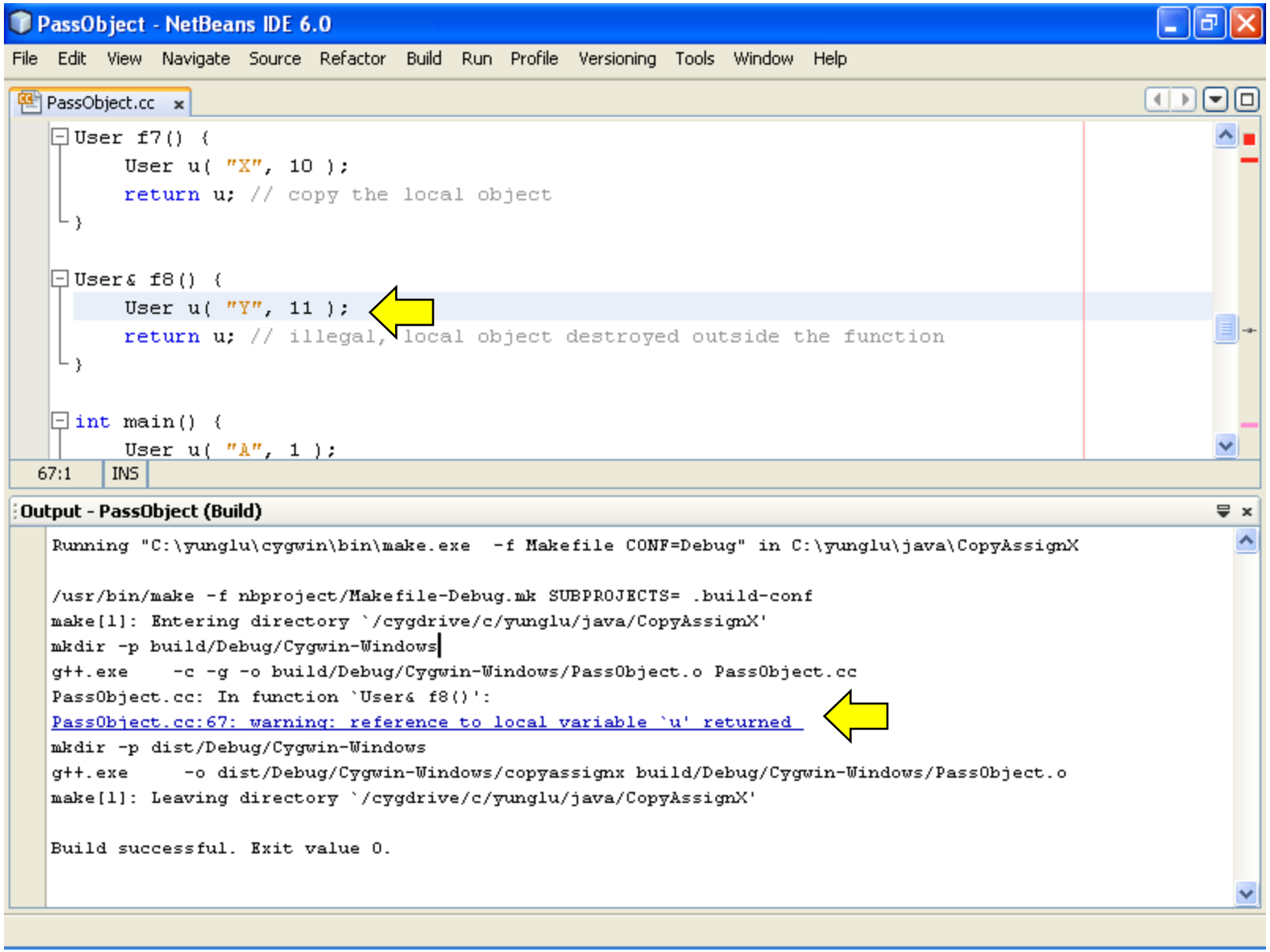

PassObject - NetBeans IDE 6.0
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
PassObject.cc
#include <iostream>
#include <string>
using namespace std;
class User {
// this class can use the copy constructor and
// the assignment operator by the compiler
// since there is no pointer
// the purpose is to show when objects are copied as parameters
public:
string name;
int age;
User(string nam, int yy) {
cout << "User(string nam, int yy)" << endl;
name = nam;
age = yy;
}
// copy constructor
User(const User & u) {
cout << "User(const User & u)" << endl;
name = u.name;
age = u.age;
}
// assignment operator
User & operator = (const User & u) {
cout << "User & operator = (const User & u)" << endl;
if (this != & u) {
4:13
INS

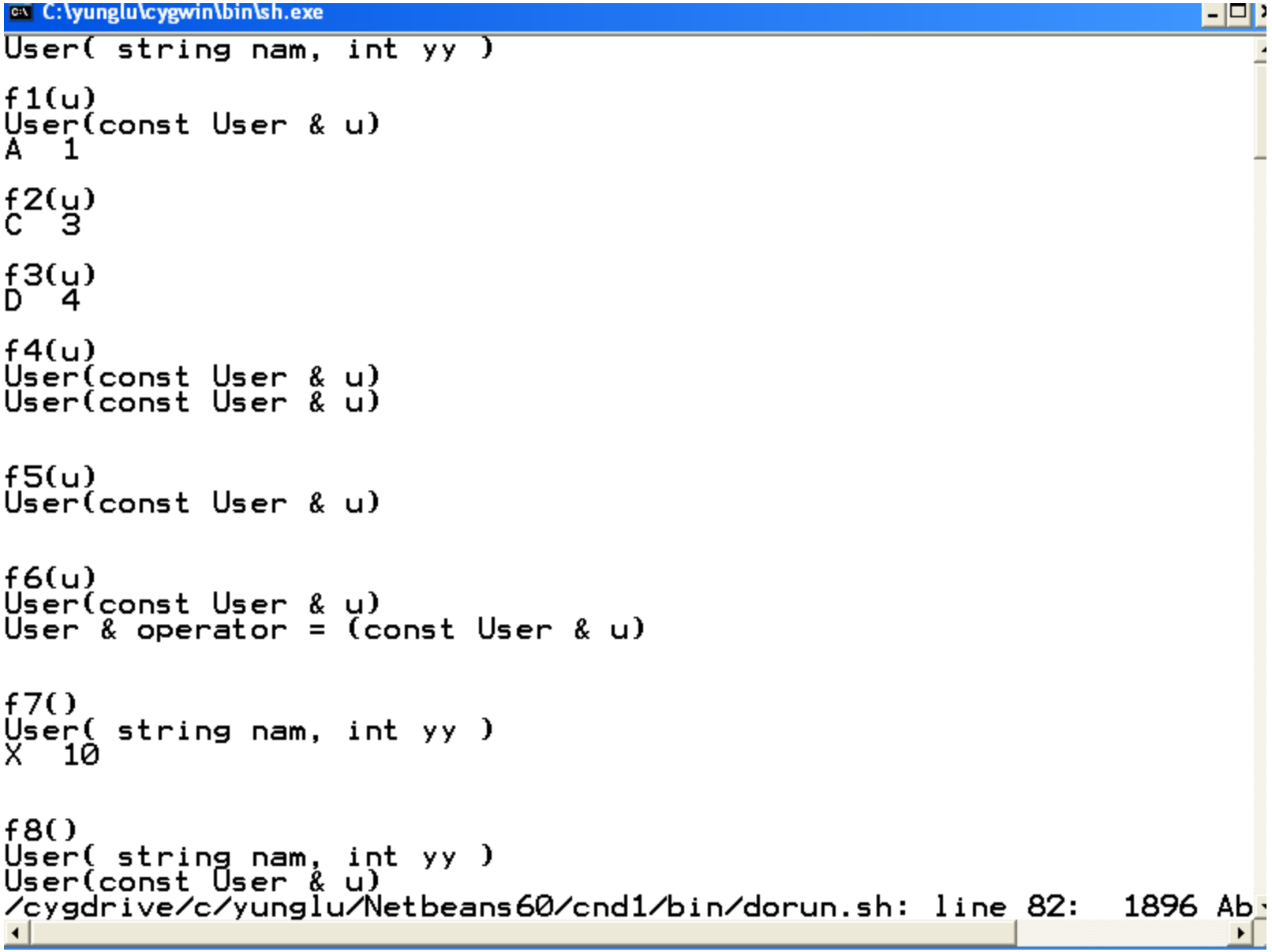

PassObject - NetBeans IDE 6.0
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
PassObject.cc
}
// assignment operator
User & operator = (const User & u) {
cout << "User & operator = (const User & u)" << endl;
if (this != & u) {
// does not need to release or allocate memory
name = u.name;
age = u.age;
}
return * this;
}
};
void f1(User u) {
u.name = "B";
u.age = 2;
}
void f2(User* u) {
u->name = "C";
u->age = 3;
}
void f3(User& u) {
u.name = "D";
u.age = 4;
4:13
INS


PassObject - NetBeans IDE 6.0
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
PassObject.cc
User f4(User u) {
return u;
}
User f5(User & u) {
return u;
}
User & f6(User & u) {
return u;
}
User f7() {
User u("X", 10);
return u; // copy the local object
}
User& f8() {
User u("Y", 11);
return u; // illegal, local object destroyed outside the function
}
int main() {
User u("A", 1);
cout << endl << "f1(u)";
f1(u);
4:13
INS


PassObject - NetBeans IDE 6.0
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
PassObject.cc
}
int main() {
User u("A", 1);
cout << endl << "f1(u)" << endl;
f1(u);
cout << u.name << " " << u.age << endl; // A 1
cout << endl << "f2(u)" << endl;
f2(& u);
cout << u.name << " " << u.age << endl; // C 3
cout << endl << "f3(u)" << endl;
f3(u);
cout << u.name << " " << u.age << endl; // D 4
cout << endl << "f4(u)" << endl;
User v1 = f4(u);
cout << endl << endl << "f5(u)" << endl;
User v2 = f5(u);
cout << endl << endl << "f6(u)" << endl;
User v3 = f6(u);
v3 = f6(u);
94:16
INS


PassObject - NetBeans IDE 6.0
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
PassObject.cc
cout << endl << "f3(u)" << endl;
f3(u);
cout << u.name << " " << u.age << endl; // D 4
cout << endl << "f4(u)" << endl;
User v1 = f4(u);
cout << endl << endl << "f5(u)" << endl;
User v2 = f5(u);
cout << endl << endl << "f6(u)" << endl;
User v3 = f6(u);
v3 = f6(u);
cout << endl << endl << "f7()" << endl;
User v4 = f7();
cout << v4.name << " " << v4.age << endl; // X 11
cout << endl << endl << "f8()" << endl;
User v5 = f8();
cout << v5.name << " " << v5.age << endl; // program crash
return 0;
}
89:21
INS


PassObject - NetBeans IDE 6.0
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
PassObject.cc
User f7() {
User u("X", 10);
return u; // copy the local object
}
User& f8() {
User u("Y", 11);
return u; // illegal, local object destroyed outside the function
}
int main() {
User u("A", 1);
67:1
INS
Output - PassObject (Build)
Running "C:\yunglu\cygwin\bin\make.exe -f Makefile CONF=Debug" in C:\yunglu\java\CopyAssignX
/usr/bin/make -f nbproject/Makefile-Debug.mk SUBPROJECTS= .build-conf
make[1]: Entering directory `/cygdrive/c/yunglu/java/CopyAssignX'
mkdir -p build/Debug/Cygwin-Windows
g++.exe -c -g -o build/Debug/Cygwin-Windows/PassObject.o PassObject.cc
PassObject.cc: In function `User& f8()':
PassObject.cc:67: warning: reference to local variable `u' returned
mkdir -p dist/Debug/Cygwin-Windows
g++.exe -o dist/Debug/Cygwin-Windows/copyassignx build/Debug/Cygwin-Windows/PassObject.o
make[1]: Leaving directory `/cygdrive/c/yunglu/java/CopyAssignX'
Build successful. Exit value 0.


C:\yunglu\cygwin\bin\sh.exe
User(string nam, int yy)
f1(u)
User(const User & u)
A 1
f2(u)
C 3
f3(u)
D 4
f4(u)
User(const User & u)
User(const User & u)
f5(u)
User(const User & u)
f6(u)
User(const User & u)
User & operator = (const User & u)
f7()
User(string nam, int yy)
X 10
f8()
User(string nam, int yy)
User(const User & u)
/cygdrive/c/yunglu/Netbeans60/cnd1/bin/dorun.sh: line 82: 1896 Ab

Self Test

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

Overload Resolution

Yung-Hsiang Lu
yunglu@purdue.edu

Overloading (by Parameters Types)

Overloading: same function name, same return type, different input parameters (numbers, types, or both)

Overload cannot be resolved by the return type.

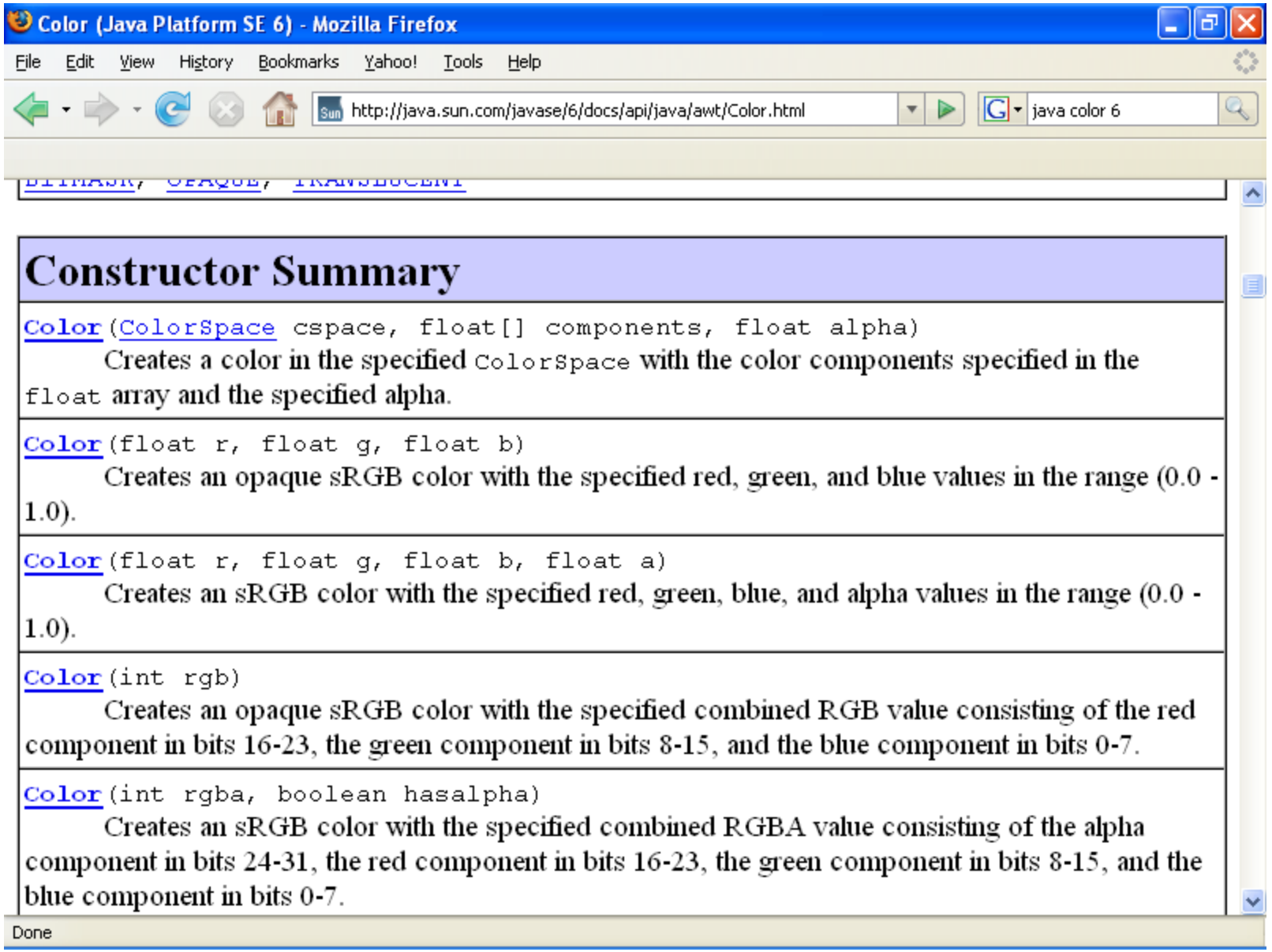
example: Java Color class has 7 constructors

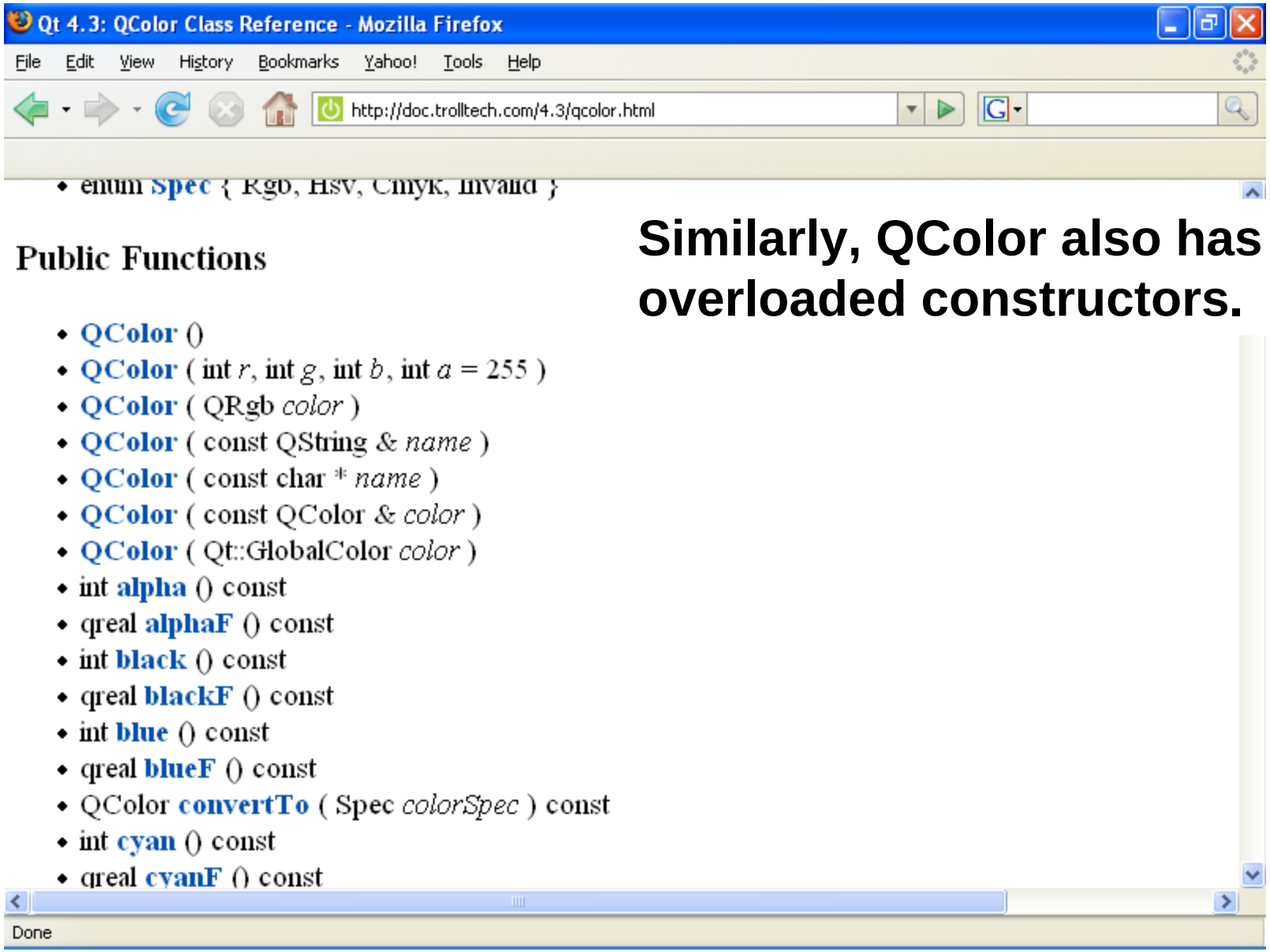
- **Color** (ColorSpace cspace, float[] components, float alpha)
  Creates a color in the specified ColorSpace with the color components specified in the float array and the specified alpha. (alpha: transparency)
- **Color**(float r, float g, float b)
  Creates an opaque sRGB color with the specified red, green, and blue values in the range (0.0 - 1.0).

- **Color**(float r, float g, float b, float a)
  Creates an sRGB color with the specified red, green, blue, and alpha values in the range (0.0 - 1.0).
- **Color**(int rgb)
  Creates an opaque sRGB color with the specified combined RGB value consisting of the red component in bits 16-23, the green component in bits 8-15, and the blue component in bits 0-7.
- **Color**(int r, int g, int b)
  Creates an opaque sRGB color with the specified red, green, and blue values in the range (0 - 255).

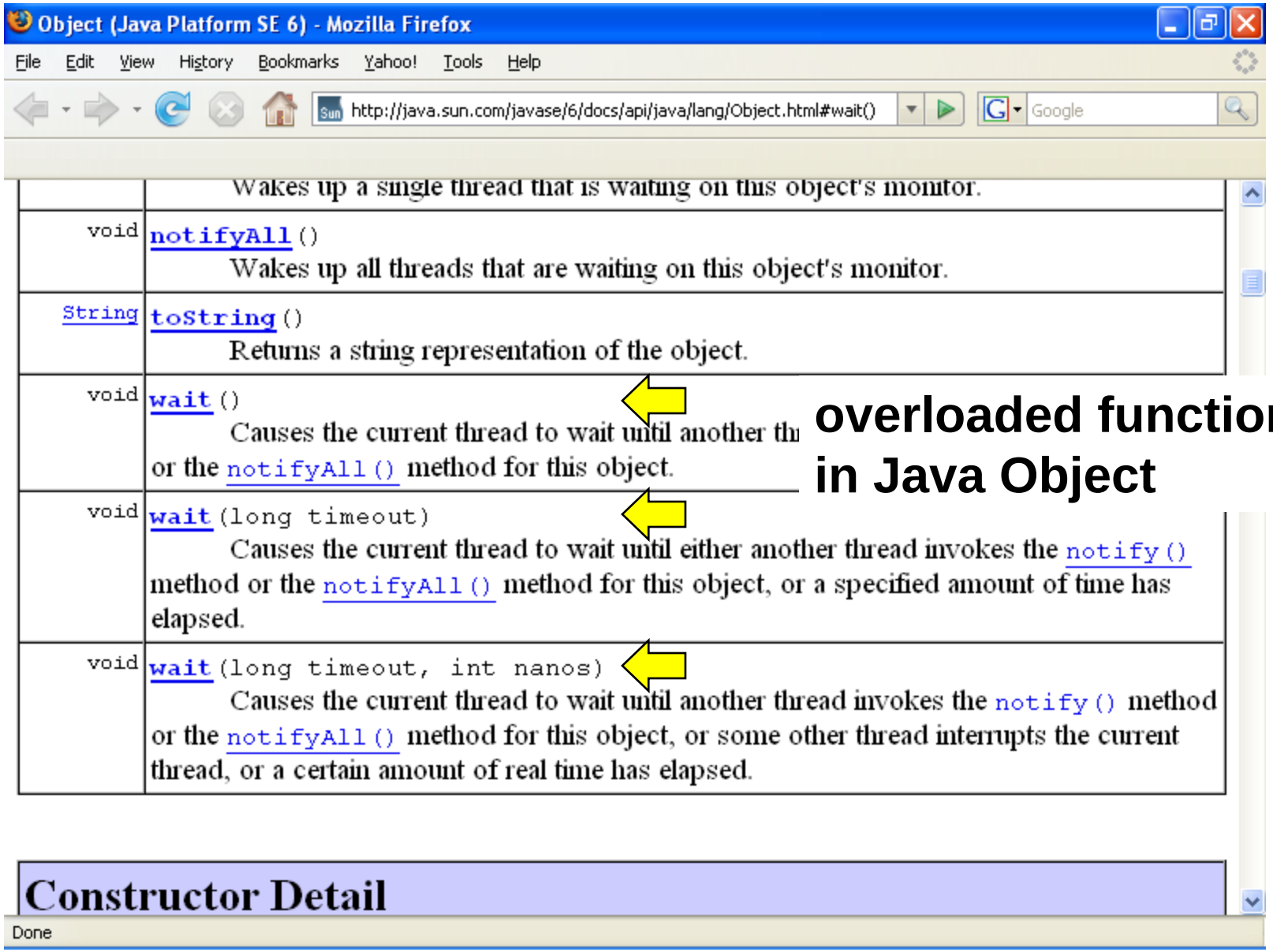
  ... 2 more constructors ... check

Why to overload? convenience: you may have different types of information
frequently used in constructors

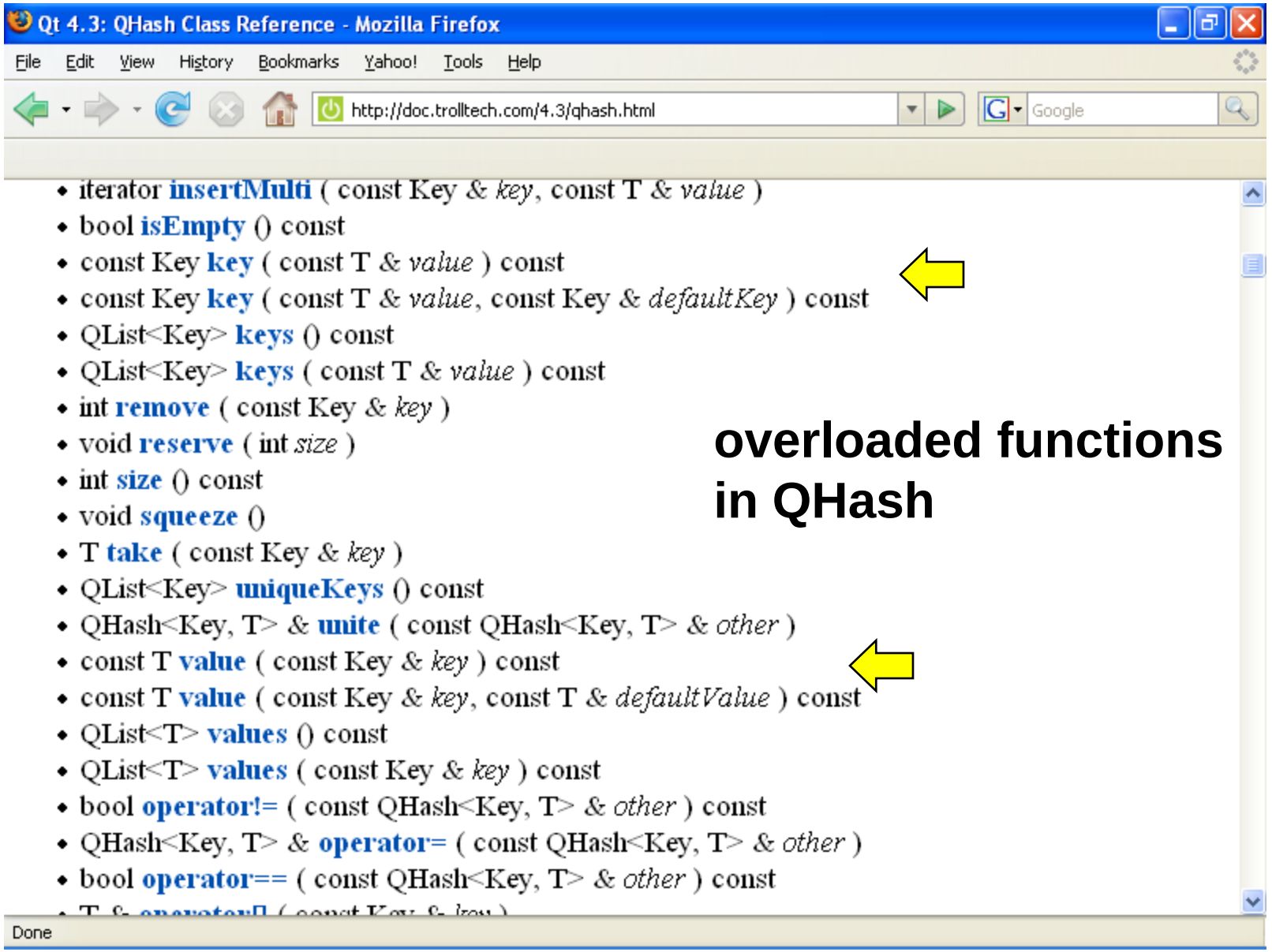

Color (Java Platform SE 6) - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://java.sun.com/javase/6/docs/api/java/awt/Color.html
java color 6
BITMASK, OPAQUE, TRANSLUCENT
Constructor Summary
Color(ColorSpace cspace, float[] components, float alpha)
Creates a color in the specified ColorSpace with the color components specified in the float array and the specified alpha.
Color(float r, float g, float b)
Creates an opaque sRGB color with the specified red, green, and blue values in the range (0.0 - 1.0).
Color(float r, float g, float b, float a)
Creates an sRGB color with the specified red, green, blue, and alpha values in the range (0.0 - 1.0).
Color(int rgb)
Creates an opaque sRGB color with the specified combined RGB value consisting of the red component in bits 16-23, the green component in bits 8-15, and the blue component in bits 0-7.
Color(int rgba, boolean hasalpha)
Creates an sRGB color with the specified combined RGBA value consisting of the alpha component in bits 24-31, the red component in bits 16-23, the green component in bits 8-15, and the blue component in bits 0-7.
Done



Similarly, QColor also has overloaded constructors.



overloaded functions in Java Object



overloaded functions in QHash

Overload Resolution
(how to find the right function to call)

- If there is an exact match, the function is called.
  - (int, double, User) → f(int, double, User)
  - (String, User, Color) → f(String, User, Color)
- If there is no exact match, the compiler will find the "closest" match by
  - type conversion through promotion (no information loss), eg. short → int, char → int, float → double
  - not const → const
  - C++: standard conversion (information may be lost), eg . double → int, int → double
  - derived class → base class

Overload Resolution

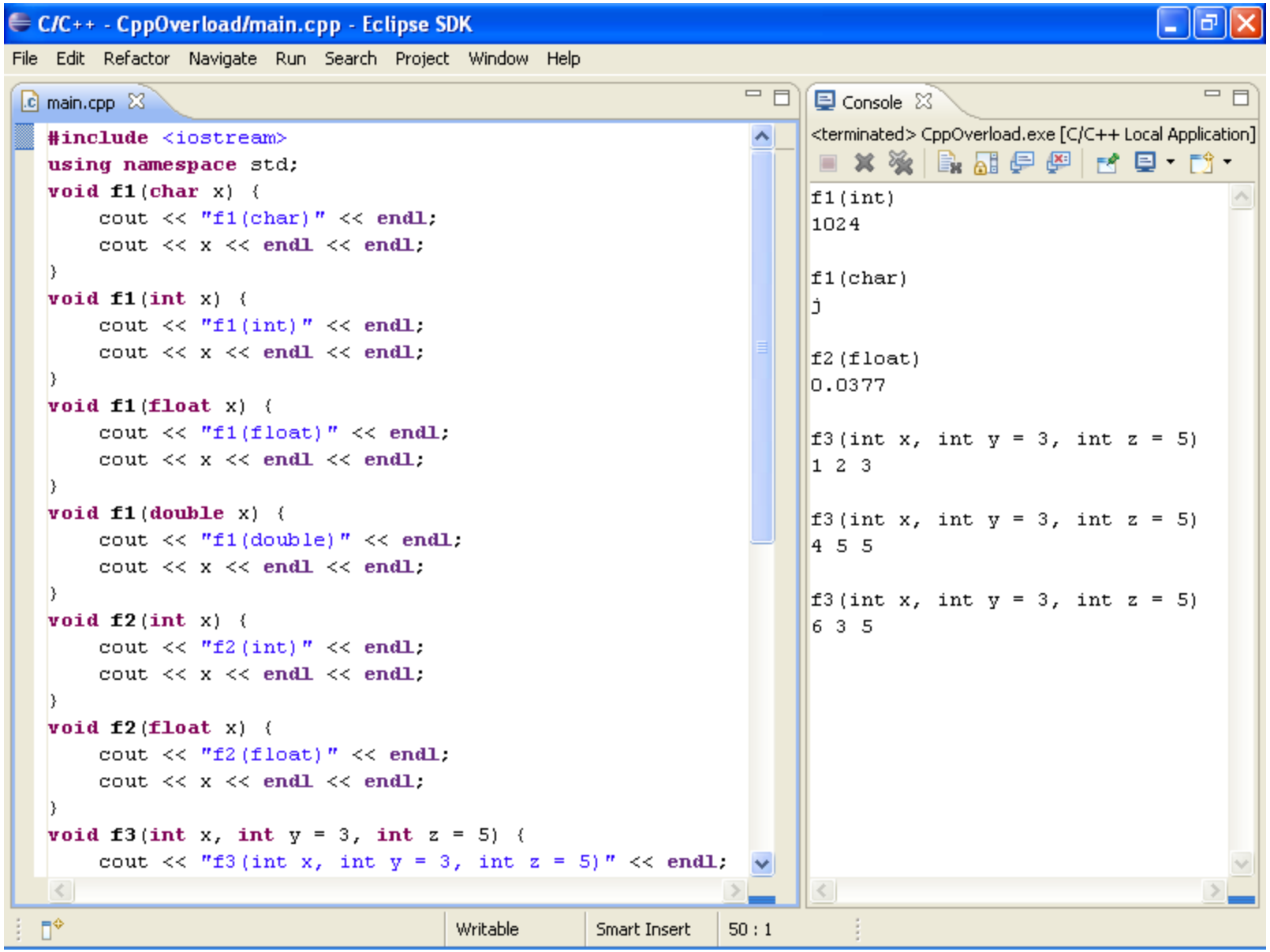
- If two (or more) matches are equally close $\Rightarrow$ error
- determined **at compilation time**, not run-time
- C++ allows default values, as one format of overloading

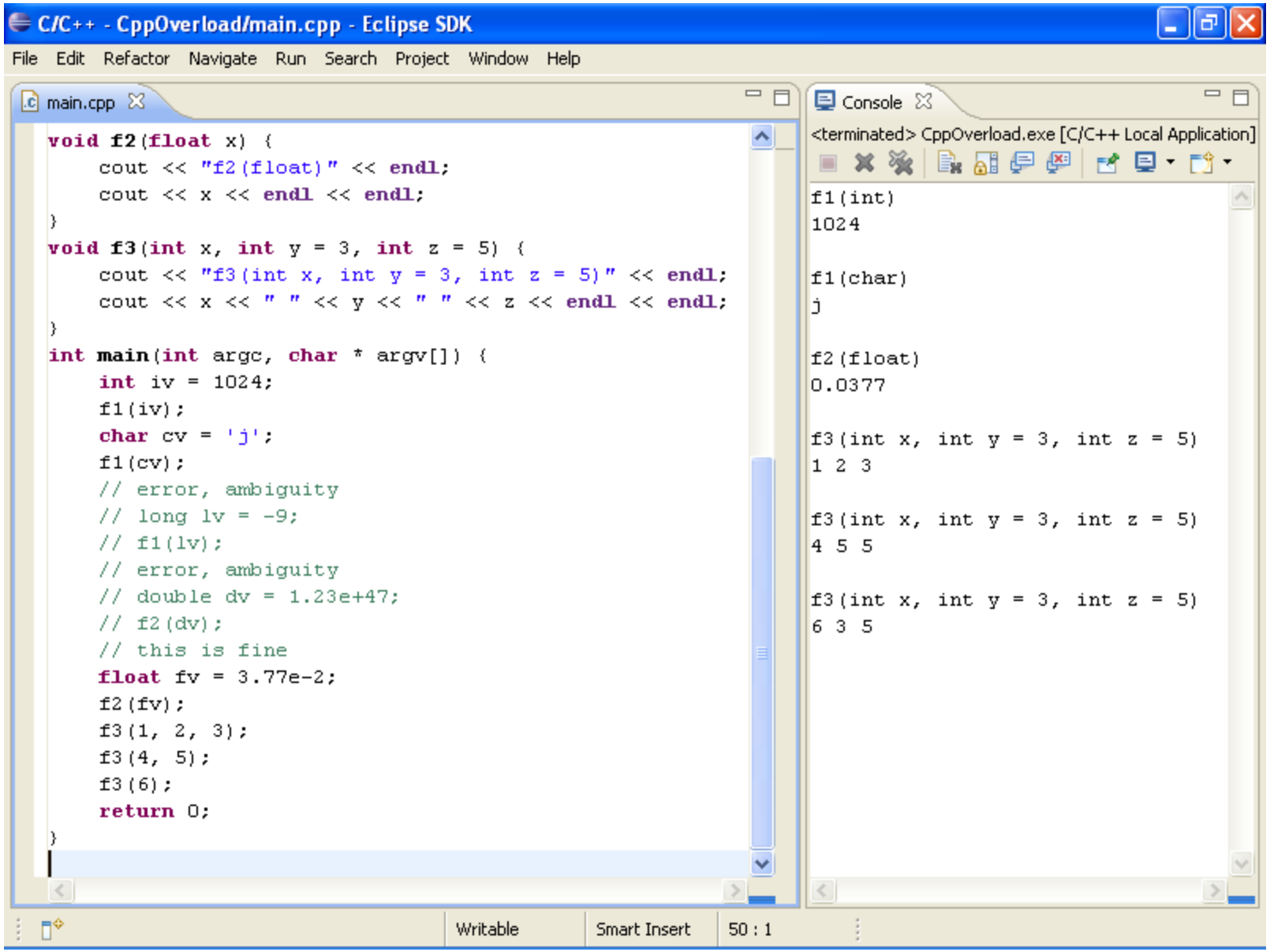
```
func(int x, int y = 8) { ... }
func(5); // same as calling func(5,8);
func(5, -11); // x = 5 and y = -11
// always start from the last parameter
func(int x = 3, int y) { ...  } // error
func(int x, double y = 0.0, int z = -10) { ... } // OK
```

Differences between Java and C++

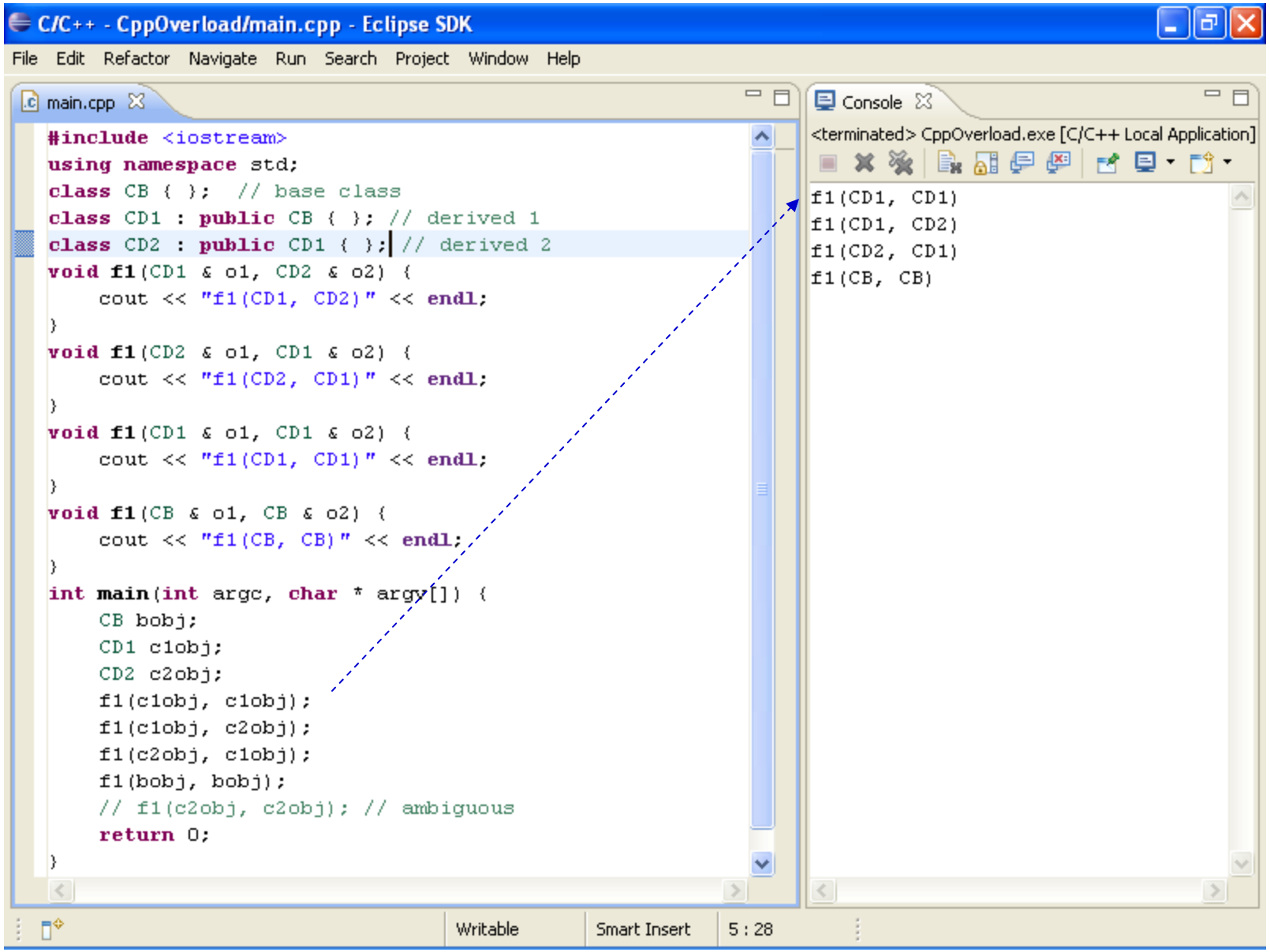
- Java allows conversions without losing information (6.7.2):
  - byte $\rightarrow$ short, int, long, float, or double
  - int $\rightarrow$ long, float, or double
  - long $\rightarrow$ float or double
  - ...
- Java does not allow programmer-defined conversion.
- Java does not allow ellipsis (undefined number of parameters, should also be avoided in C++)

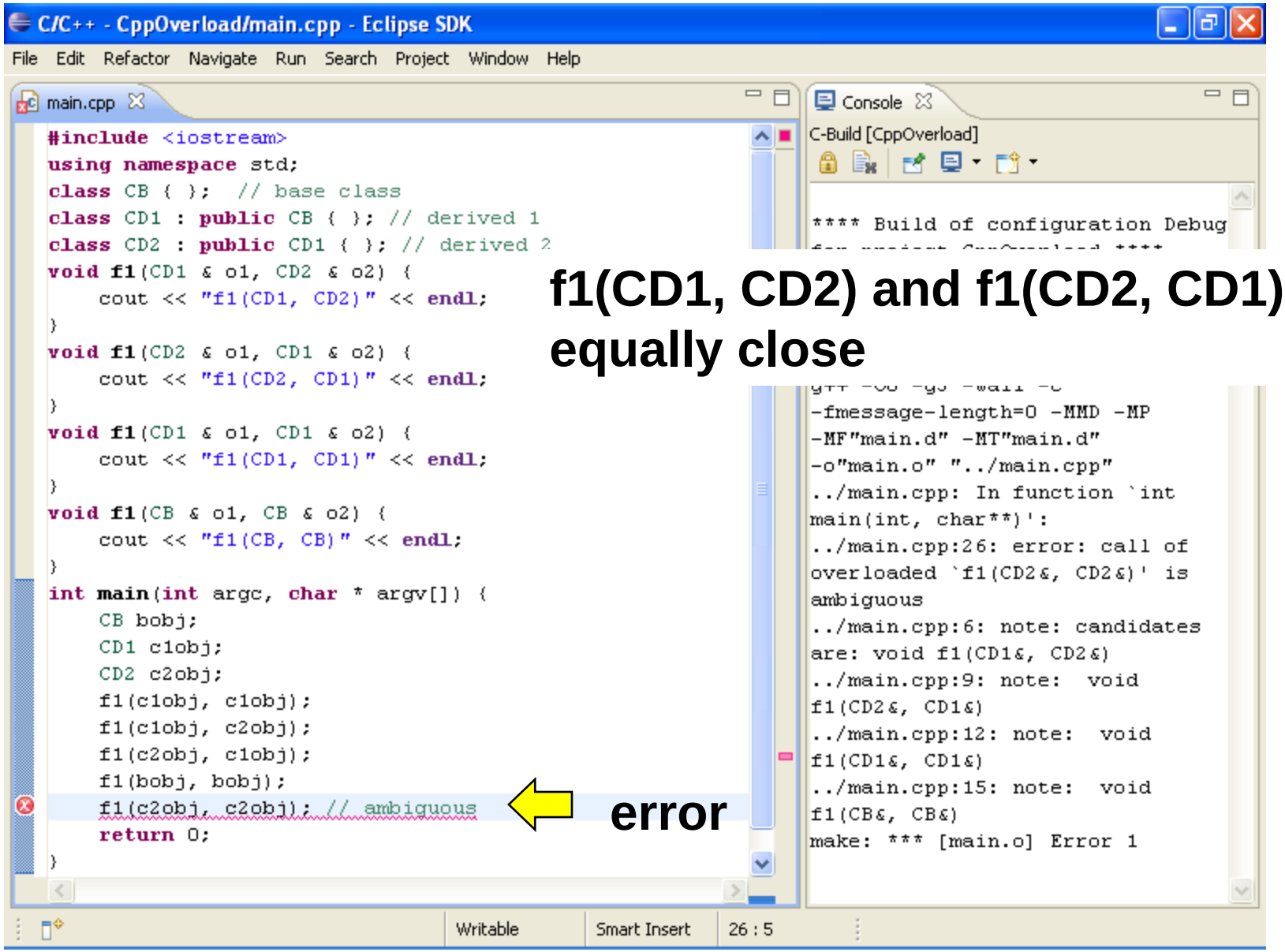
C++ Overload using Primitive Types


C/C++ - CppOverload/main.cpp - Eclipse SDK
File Edit Refactor Navigate Run Search Project Window Help
main.cpp
#include <iostream>
using namespace std;
void f1(char x) {
cout << "f1(char)" << endl;
cout << x << endl << endl;
}
void f1(int x) {
cout << "f1(int)" << endl;
cout << x << endl << endl;
}
void f1(float x) {
cout << "f1(float)" << endl;
cout << x << endl << endl;
}
void f1(double x) {
cout << "f1(double)" << endl;
cout << x << endl << endl;
}
void f2(int x) {
cout << "f2(int)" << endl;
cout << x << endl << endl;
}
void f2(float x) {
cout << "f2(float)" << endl;
cout << x << endl << endl;
}
void f3(int x, int y = 3, int z = 5) {
cout << "f3(int x, int y = 3, int z = 5)" << endl;
Console
<terminated> CppOverload.exe [C/C++ Local Application]
f1(int)
1024
f1(char)
j
f2(float)
0.0377
f3(int x, int y = 3, int z = 5)
1 2 3
f3(int x, int y = 3, int z = 5)
4 5 5
f3(int x, int y = 3, int z = 5)
6 3 5
Writable
Smart Insert
50 : 1


C/C++ - CppOverload/main.cpp - Eclipse SDK
File Edit Refactor Navigate Run Search Project Window Help
main.cpp
void f2(float x) {
cout << "f2(float)" << endl;
cout << x << endl << endl;
}
void f3(int x, int y = 3, int z = 5) {
cout << "f3(int x, int y = 3, int z = 5)" << endl;
cout << x << " " << y << " " << z << endl << endl;
}
int main(int argc, char * argv[]) {
int iv = 1024;
f1(iv);
char cv = 'j';
f1(cv);
// error, ambiguity
// long lv = -9;
// f1(lv);
// error, ambiguity
// double dv = 1.23e+47;
// f2(dv);
// this is fine
float fv = 3.77e-2;
f2(fv);
f3(1, 2, 3);
f3(4, 5);
f3(6);
return 0;
}
Console
<terminated> CppOverload.exe [C/C++ Local Application]
f1(int)
1024
f1(char)
j
f2(float)
0.0377
f3(int x, int y = 3, int z = 5)
1 2 3
f3(int x, int y = 3, int z = 5)
4 5 5
f3(int x, int y = 3, int z = 5)
6 3 5
Writable
Smart Insert
50 : 1

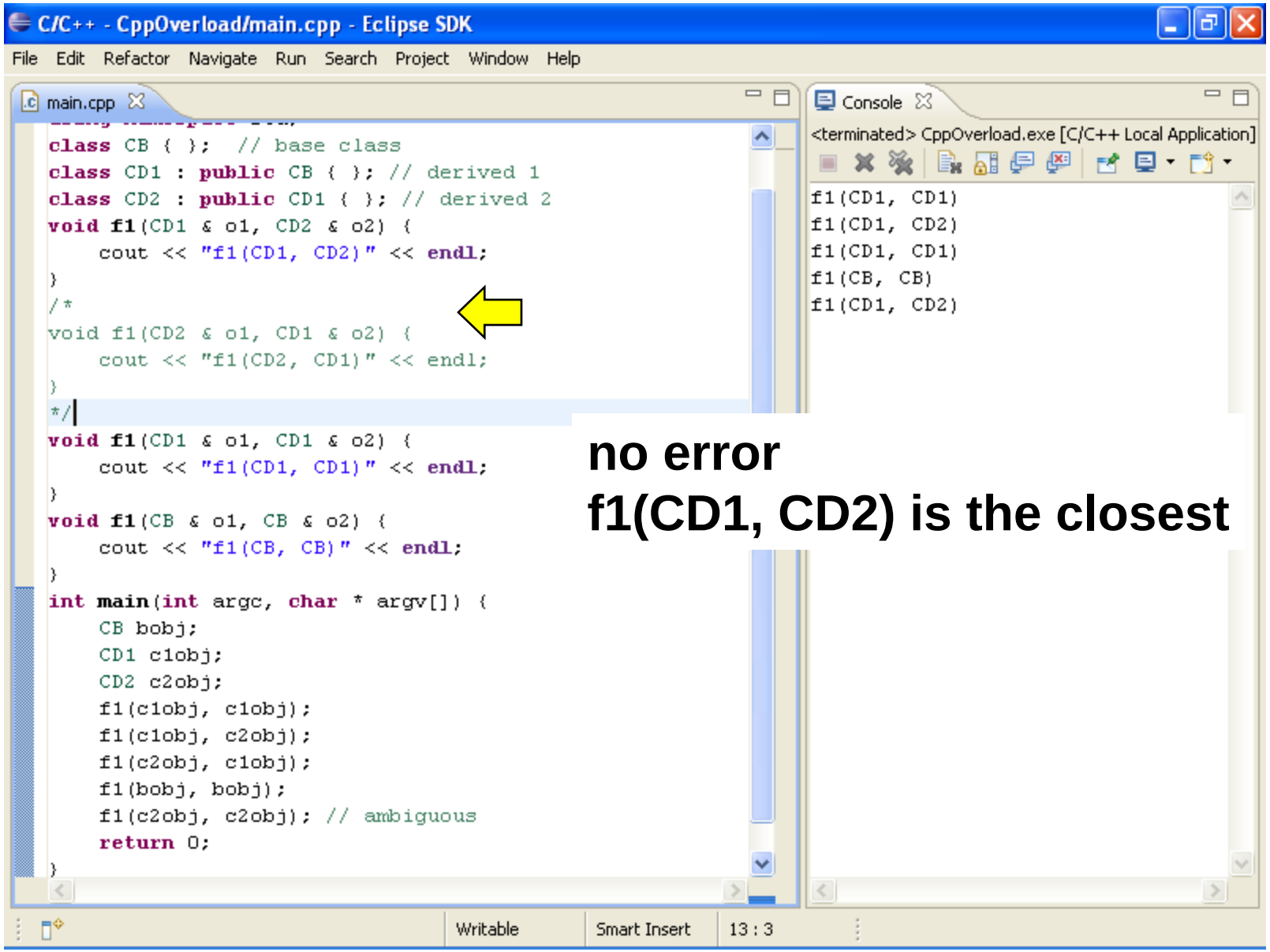
C++ Overload with Objects


C/C++ - CppOverload/main.cpp - Eclipse SDK
File Edit Refactor Navigate Run Search Project Window Help
main.cpp
#include <iostream>
using namespace std;
class CB { }; // base class
class CD1 : public CB { }; // derived 1
class CD2 : public CD1 { }; // derived 2
void f1(CD1 & o1, CD2 & o2) {
cout << "f1(CD1, CD2)" << endl;
}
void f1(CD2 & o1, CD1 & o2) {
cout << "f1(CD2, CD1)" << endl;
}
void f1(CD1 & o1, CD1 & o2) {
cout << "f1(CD1, CD1)" << endl;
}
void f1(CB & o1, CB & o2) {
cout << "f1(CB, CB)" << endl;
}
int main(int argc, char * argv[]) {
CB bobj;
CD1 c1obj;
CD2 c2obj;
f1(c1obj, c1obj);
f1(c1obj, c2obj);
f1(c2obj, c1obj);
f1(bobj, bobj);
// f1(c2obj, c2obj); // ambiguous
return 0;
}
Console
<terminated> CppOverload.exe [C/C++ Local Application]
f1(CD1, CD1)
f1(CD1, CD2)
f1(CD2, CD1)
f1(CB, CB)
Writable
Smart Insert
5 : 28



f1(CD1, CD2) and f1(CD2, CD1) equally close

error



no error
f1(CD1, CD2) is the closest

Self Test