

ECE 462

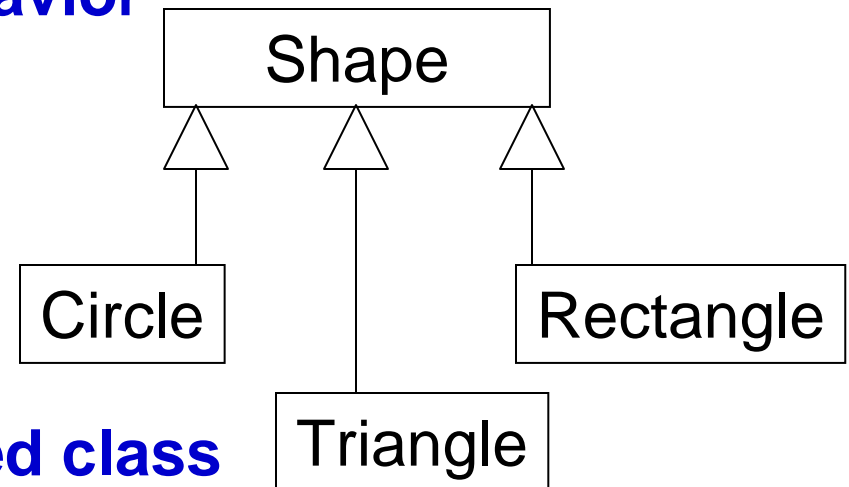
**Object-Oriented Programming
using C++ and Java**

Program Structure

Yung-Hsiang Lu
yunглу@purdue.edu

Objects and Classes

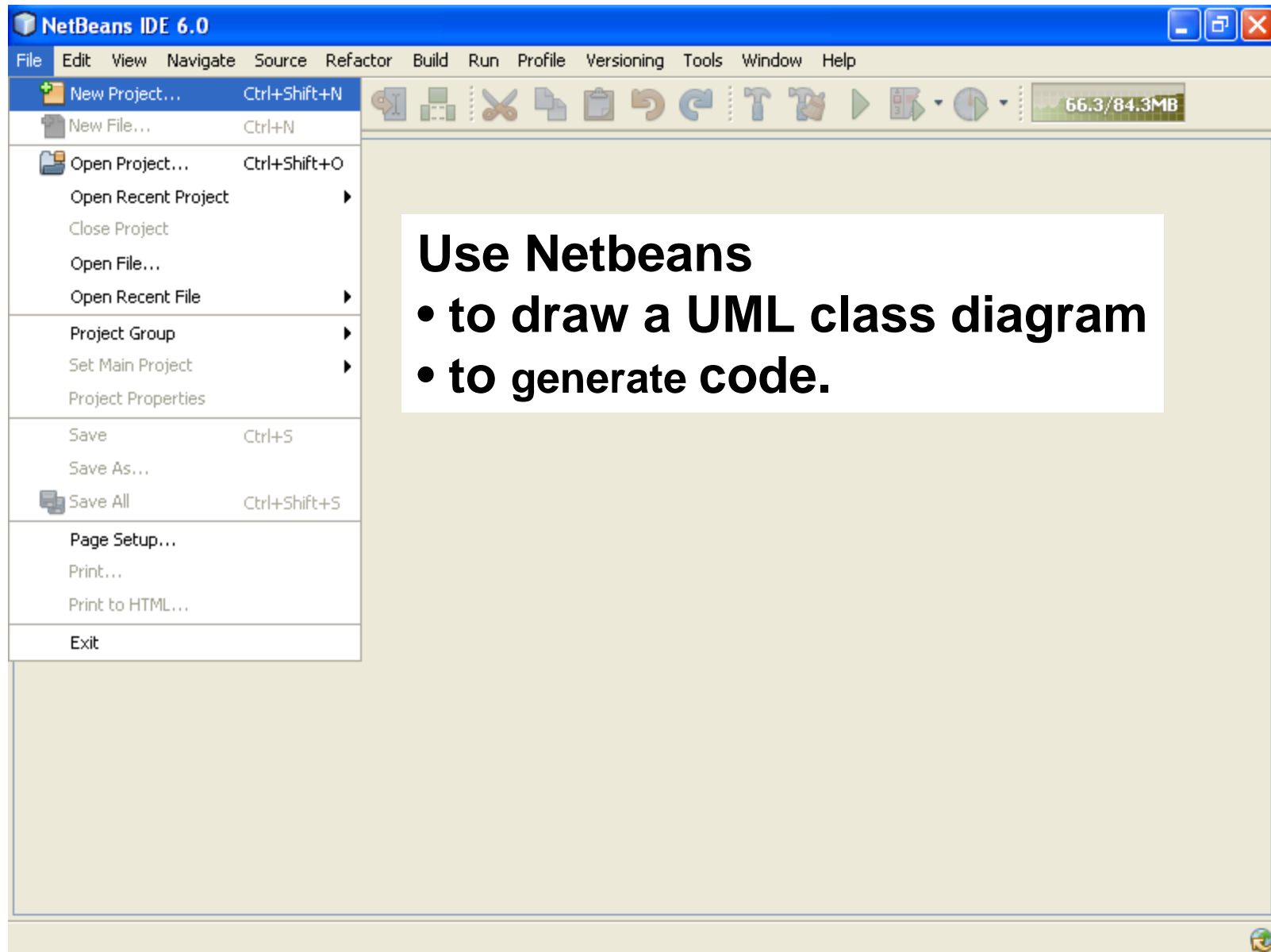
- Organize objects with similar properties (attributes and behaviors) into classes: human, car, computer, phone, window, circle, rectangle, triangle, bridge, skyscraper ...
- Inheritance: find **commonality** among **classes** (not objects) and create a base class that represents the common **interfaces** and **behavior**
- Common interface of Shape:
 - color
 - line style and thickness
 - area but it is **computed differently in each derived class**

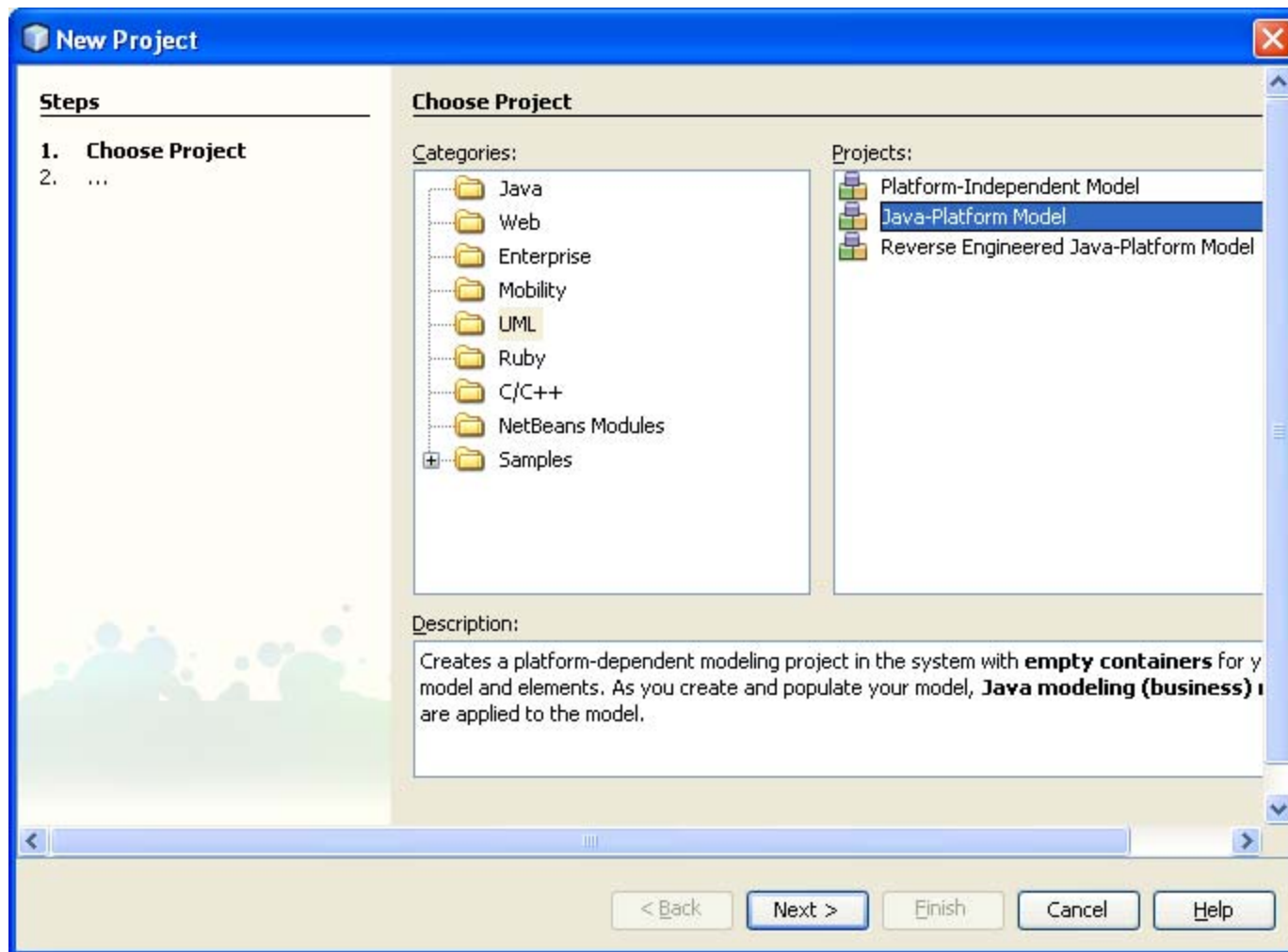


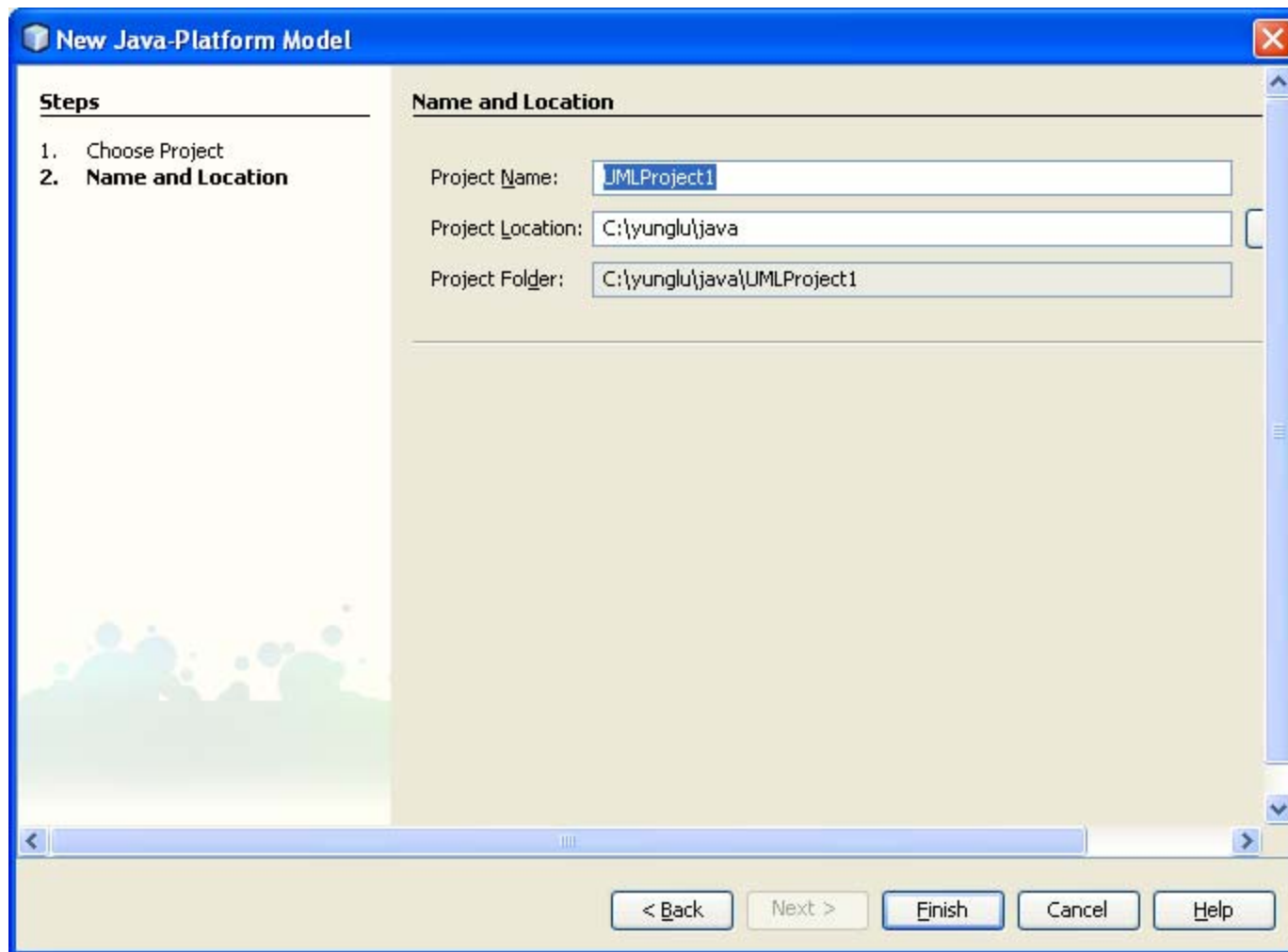
Reuse (Base or Derived?)

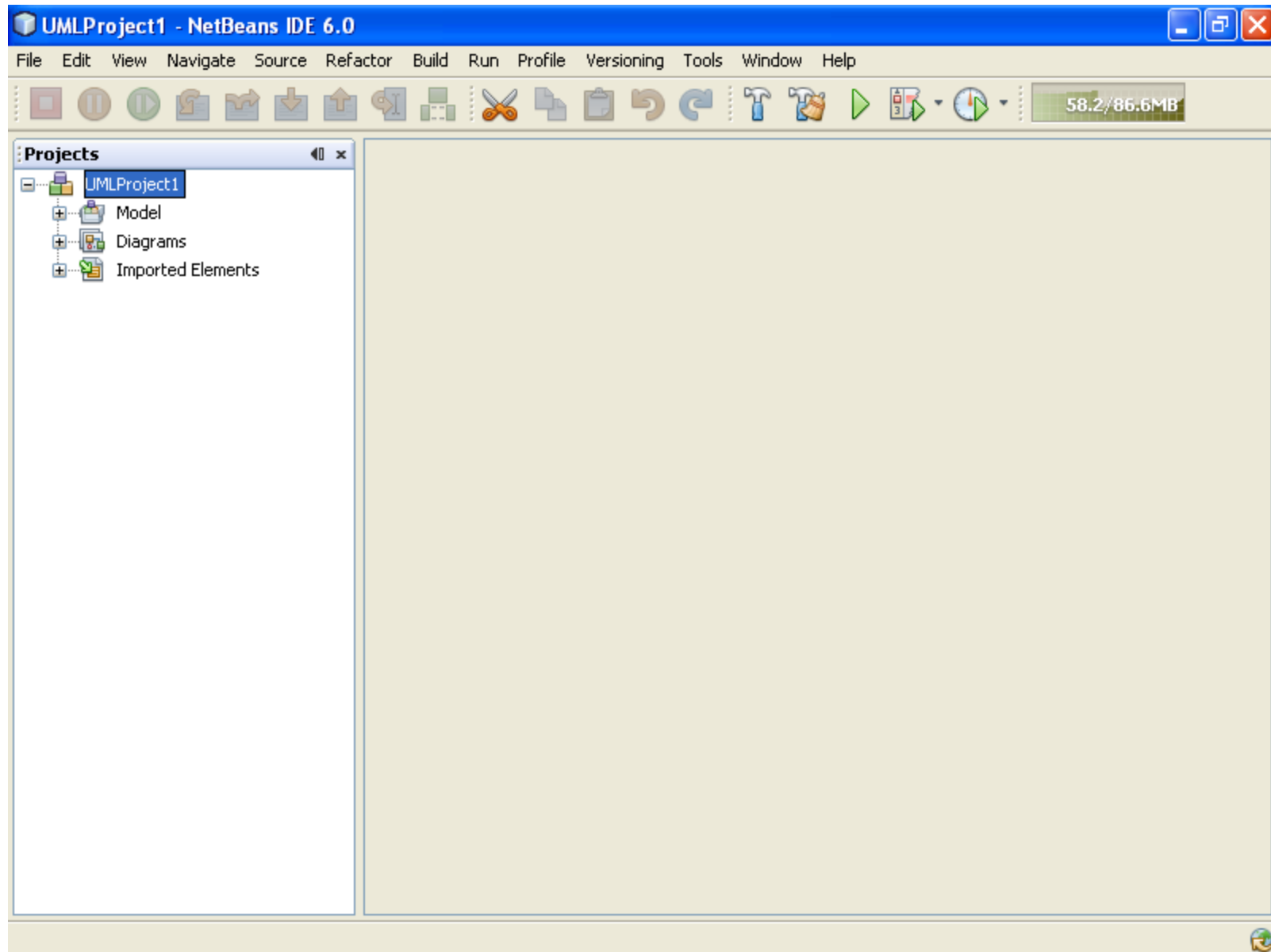
- Attribute (member data)
 - If an attribute is shared by all derived classes, the attribute should be declared in the base class. example: color, line style, thickness.
 - If an attribute is unique to a class, it should be declared in this derived class, example: radius for circle.
- Behavior (member function, method):
 - If a method is available to all derived classes, such as getArea and setLineStyle, it should be **declared** in the base class.
 - If the method's implementation is applicable to all derived classes, it should be **implemented** in the base class.
 - If the implementation is **unique to each derived class**, it should be implemented in the derived classes.

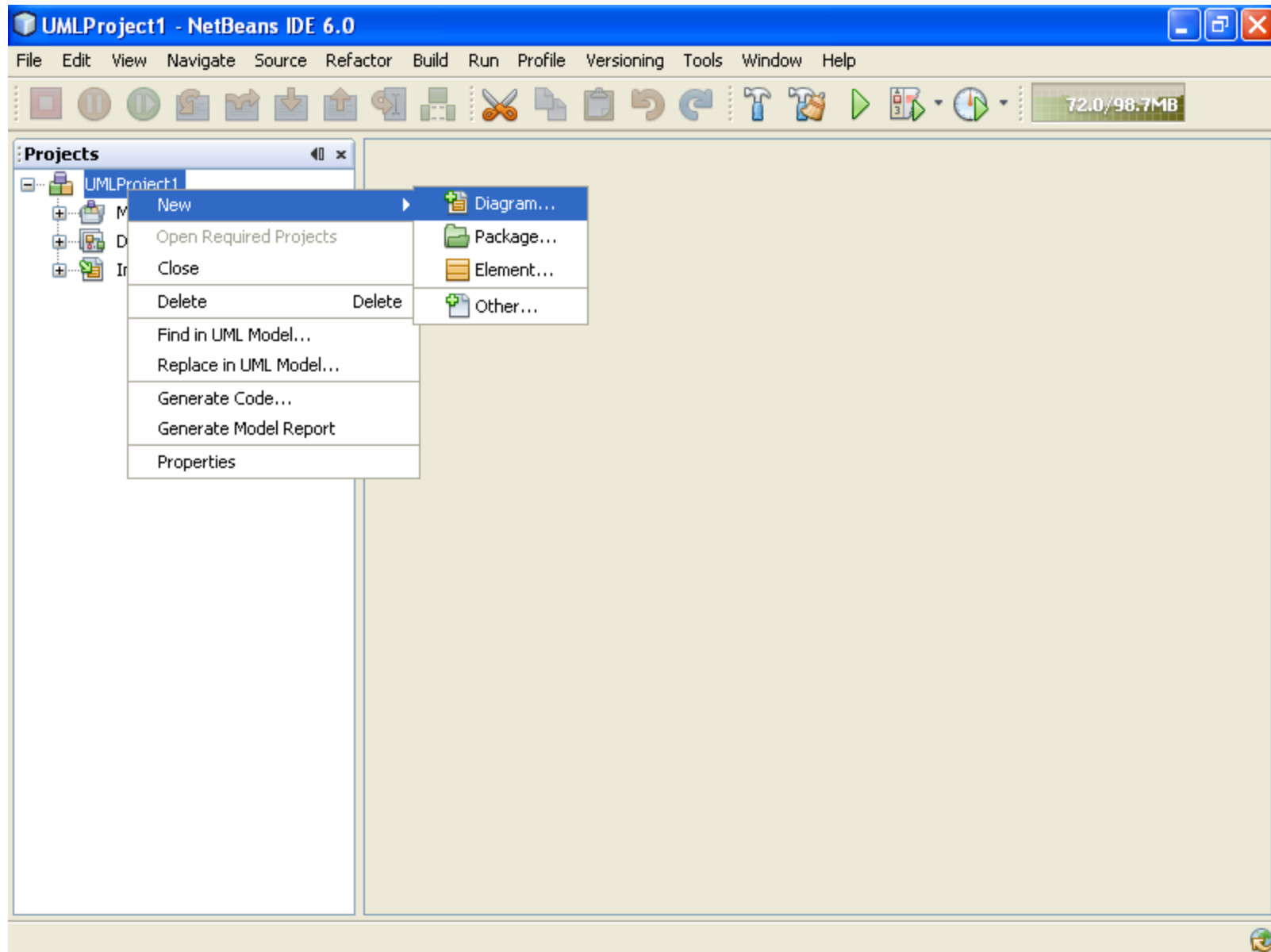
Generate Code from UML Class Diagram

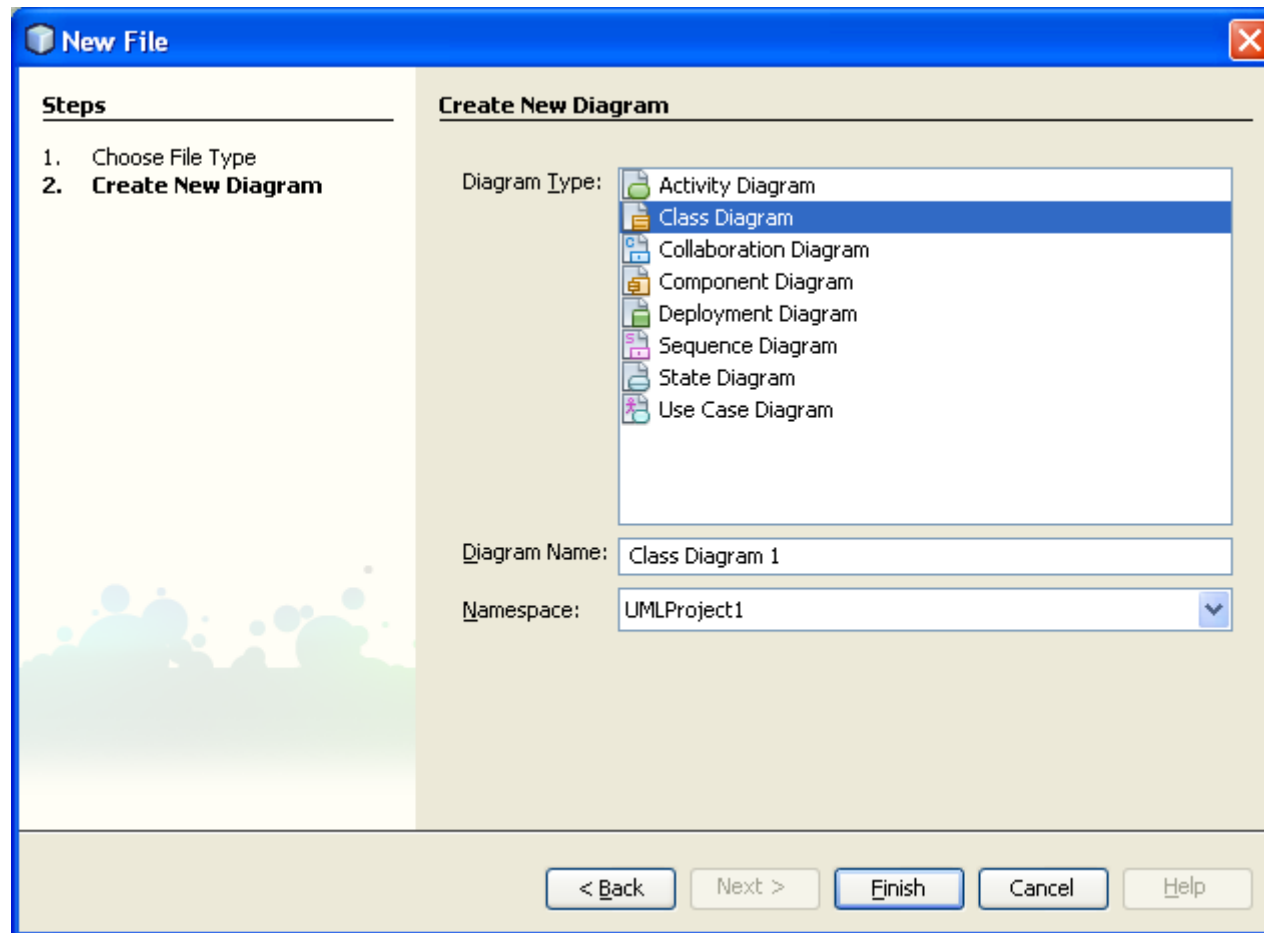


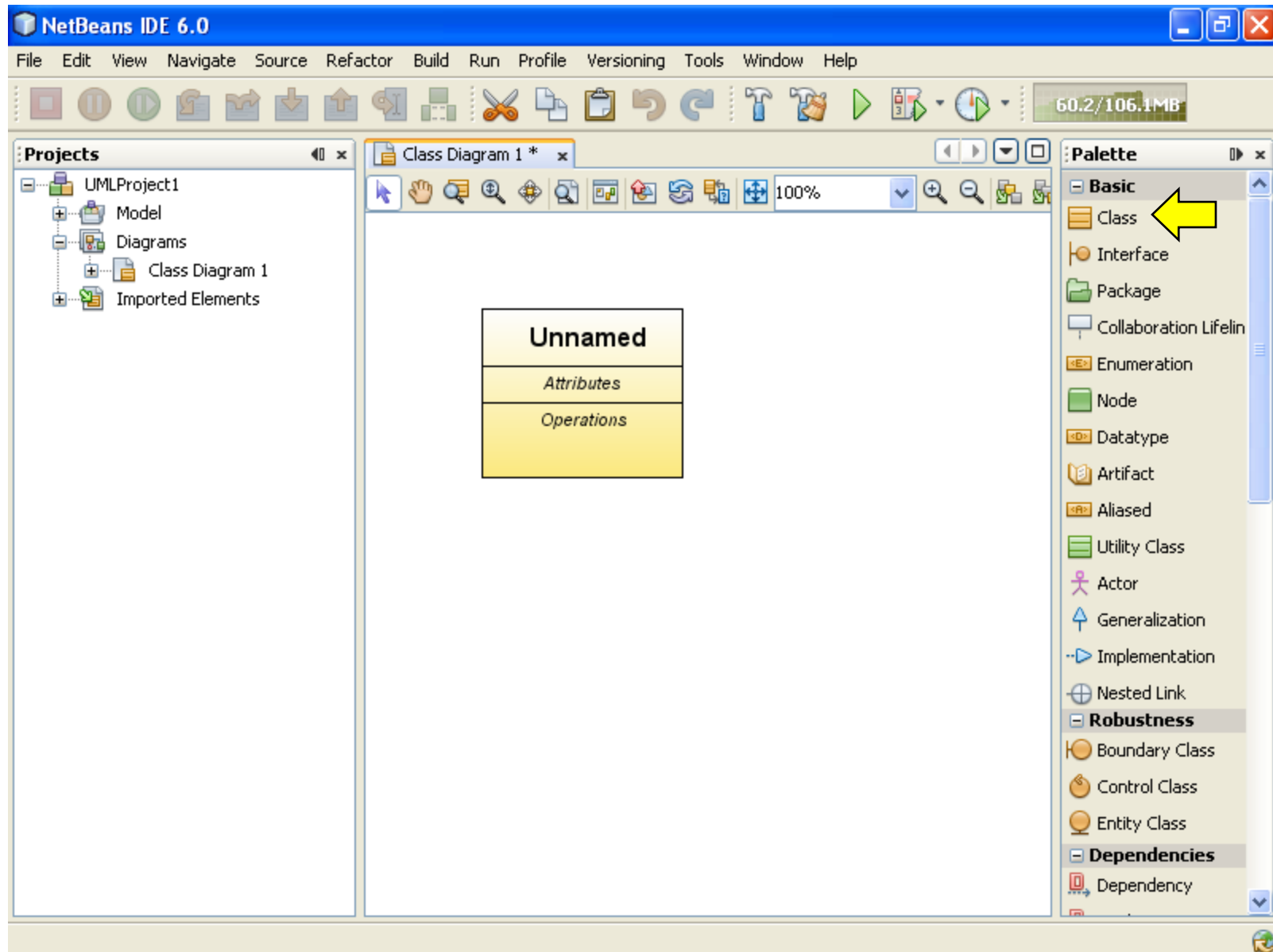


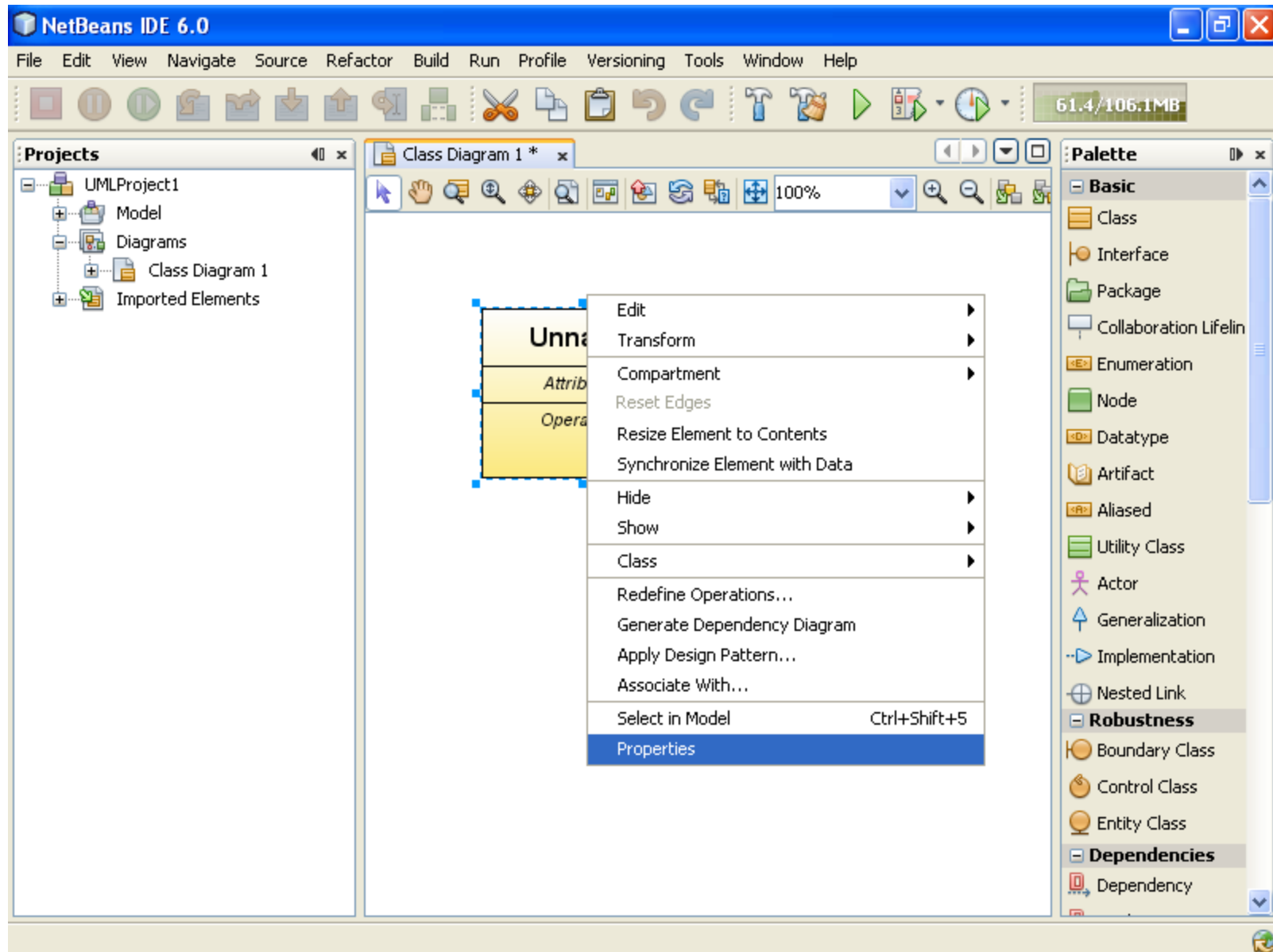













Unnamed - Properties [X]

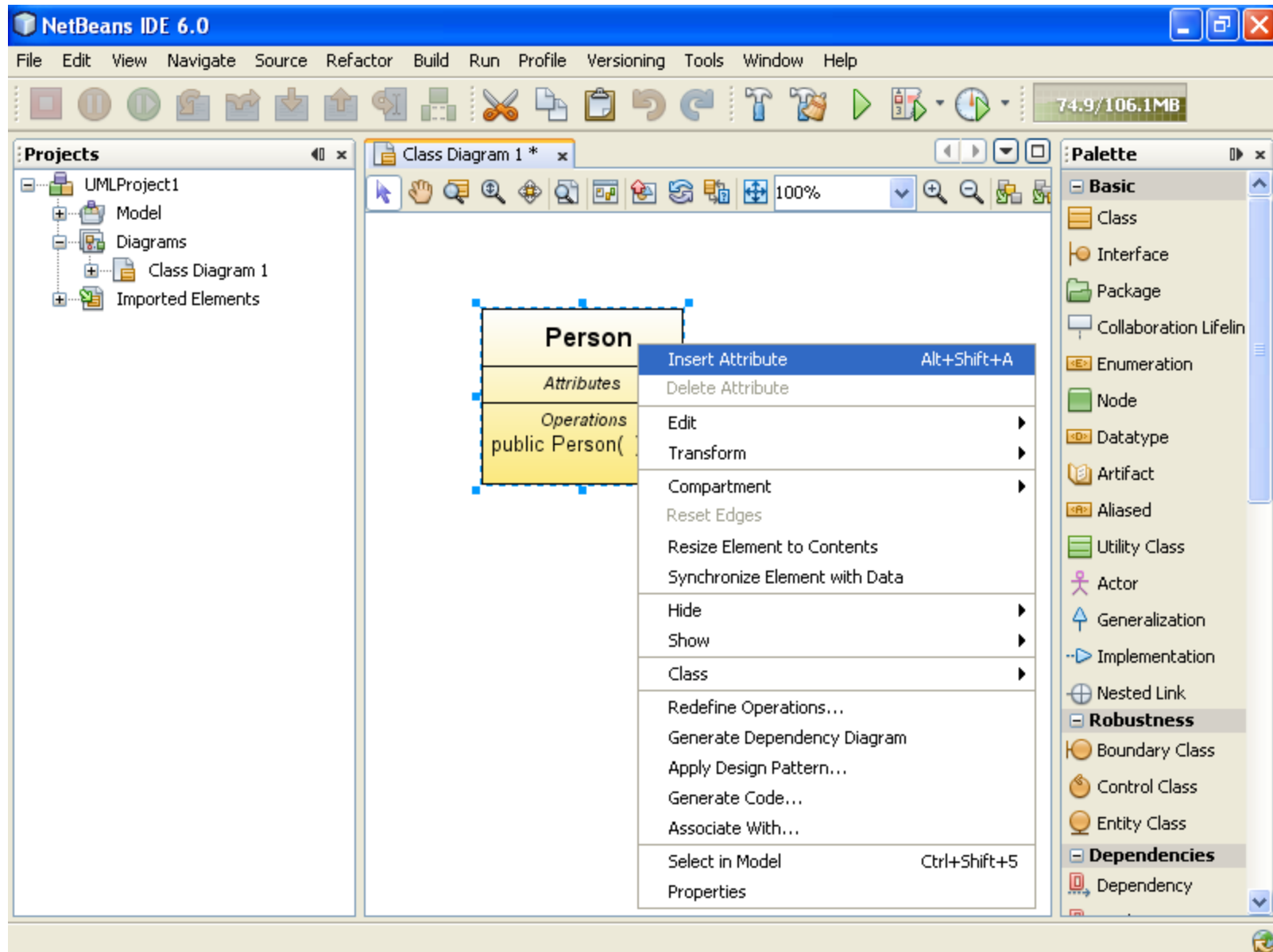
[-] Class

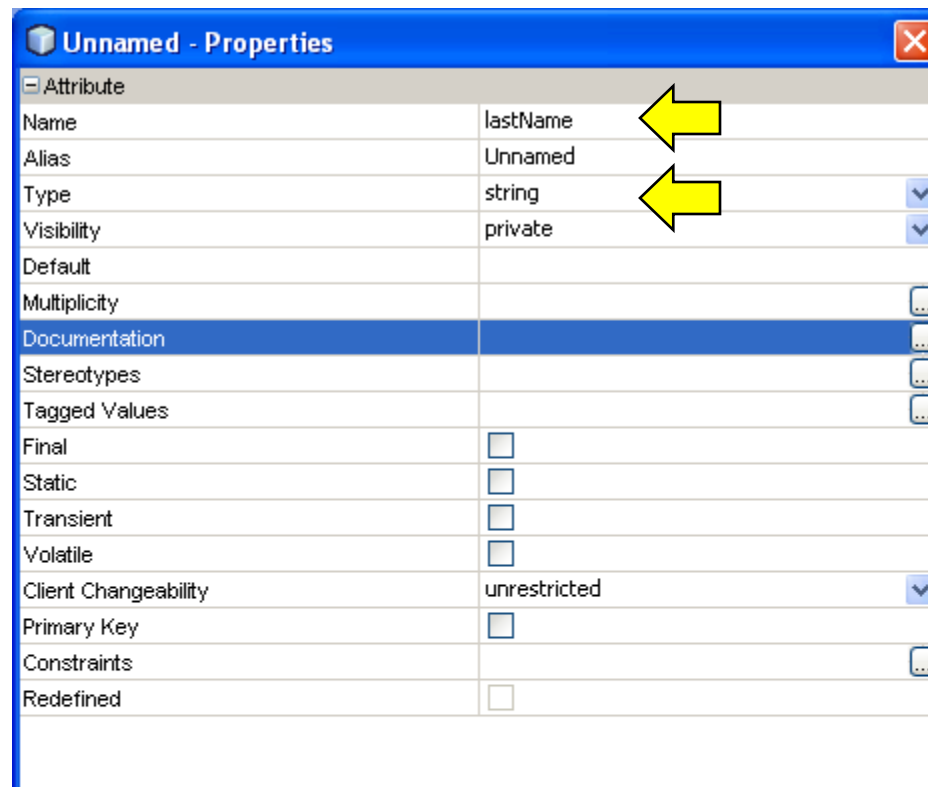
Name	Person	
Alias	Unnamed	
Visibility	public	▼
Stereotypes		...
Tagged Values		...
Documentation		...
Constraints		...
Template Parameters		...
Transient	<input checked="" type="checkbox"/>	
Abstract	<input type="checkbox"/>	
Leaf	<input type="checkbox"/>	
Active	<input type="checkbox"/>	

Name [?]

Close







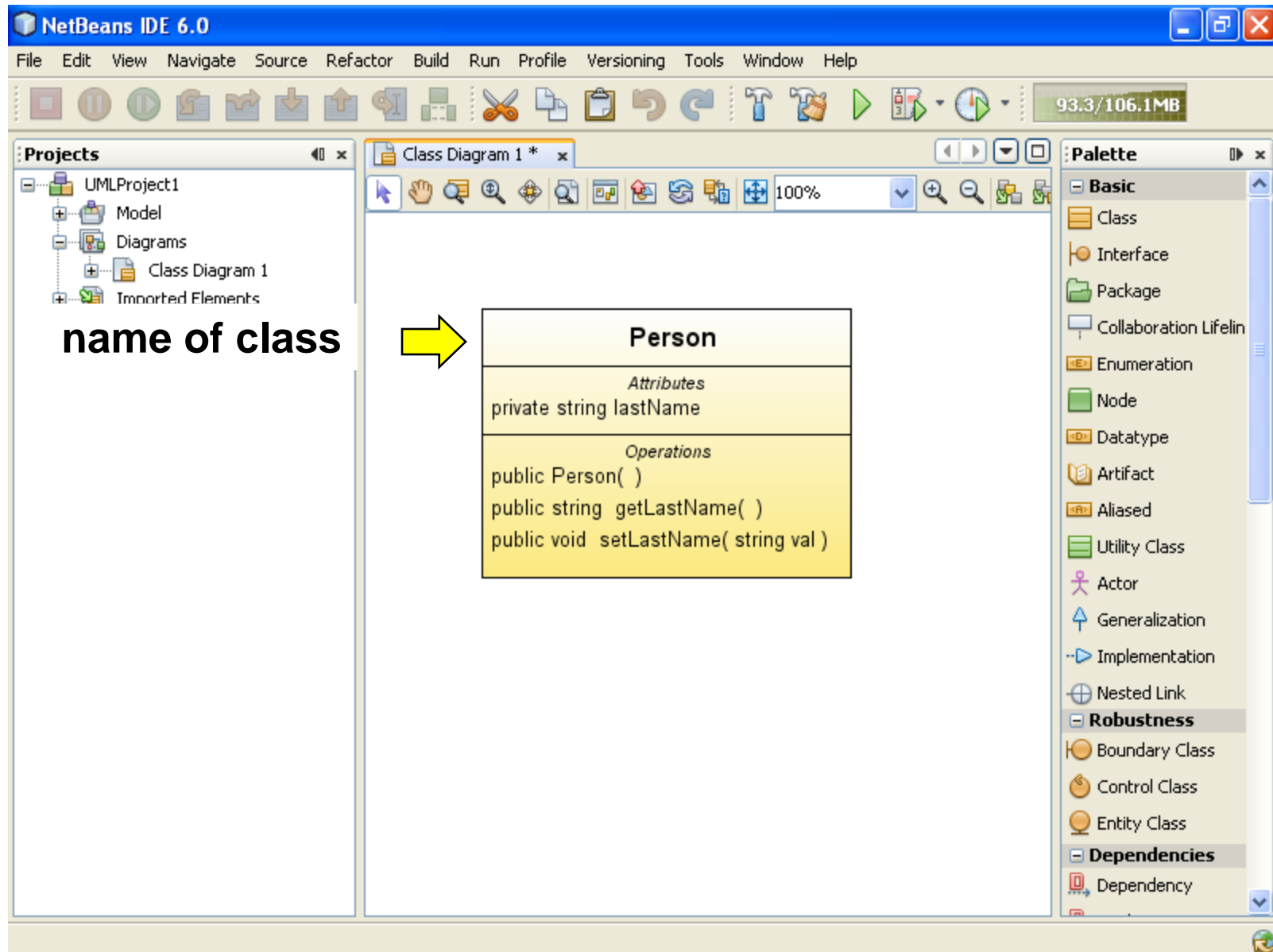
The image shows a 'Properties' dialog box for an unnamed attribute. The dialog has a blue title bar with the text 'Unnamed - Properties' and a close button. The main area is a table with two columns: the left column lists property names, and the right column shows their current values. Two yellow arrows point to the 'lastName' value in the 'Name' row and the 'string' value in the 'Type' row.

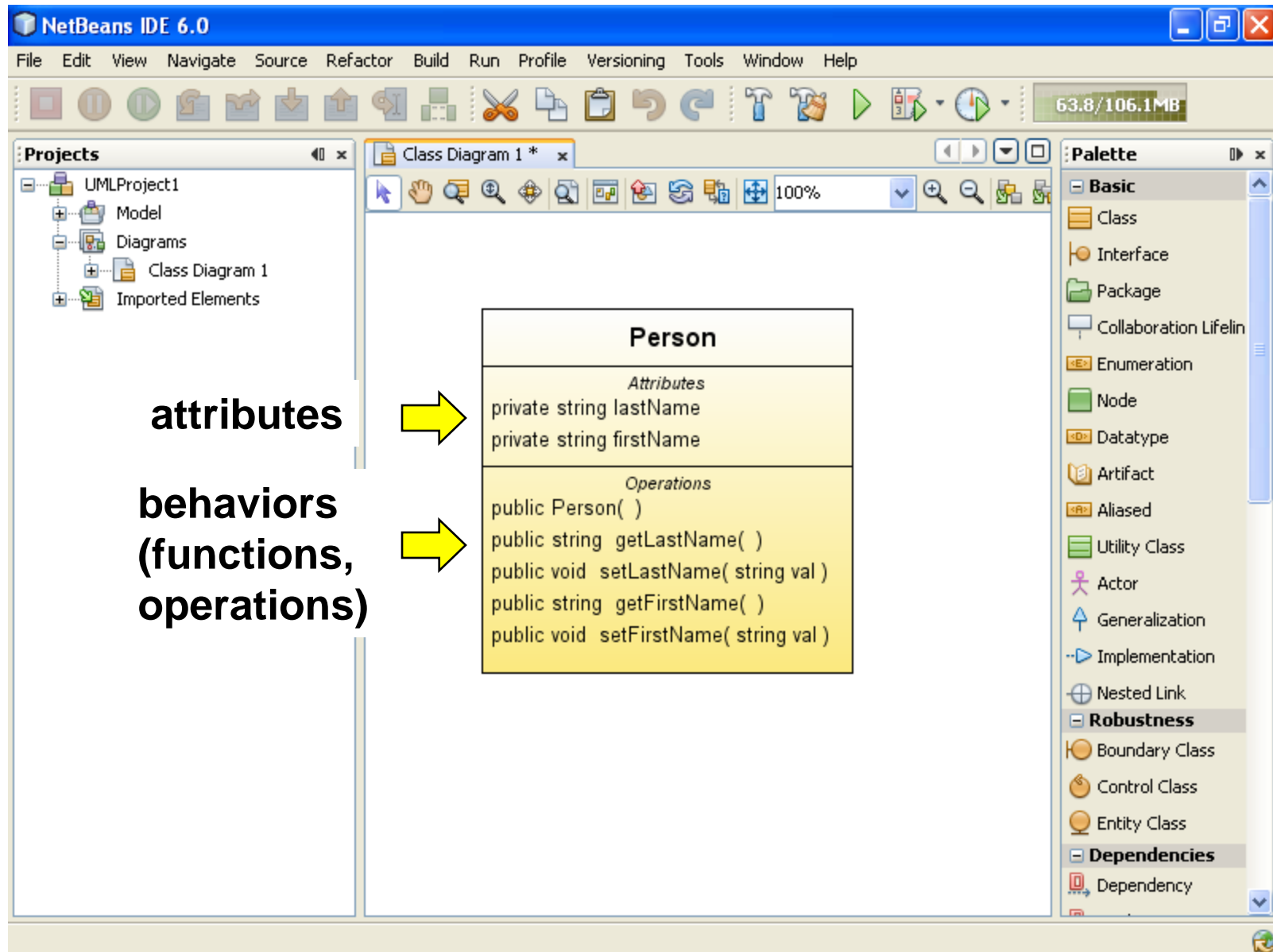
Attribute	
Name	lastName
Alias	Unnamed
Type	string
Visibility	private
Default	
Multiplicity	
Documentation	
Stereotypes	
Tagged Values	
Final	<input type="checkbox"/>
Static	<input type="checkbox"/>
Transient	<input type="checkbox"/>
Volatile	<input type="checkbox"/>
Client Changeability	unrestricted
Primary Key	<input type="checkbox"/>
Constraints	
Redefined	<input type="checkbox"/>

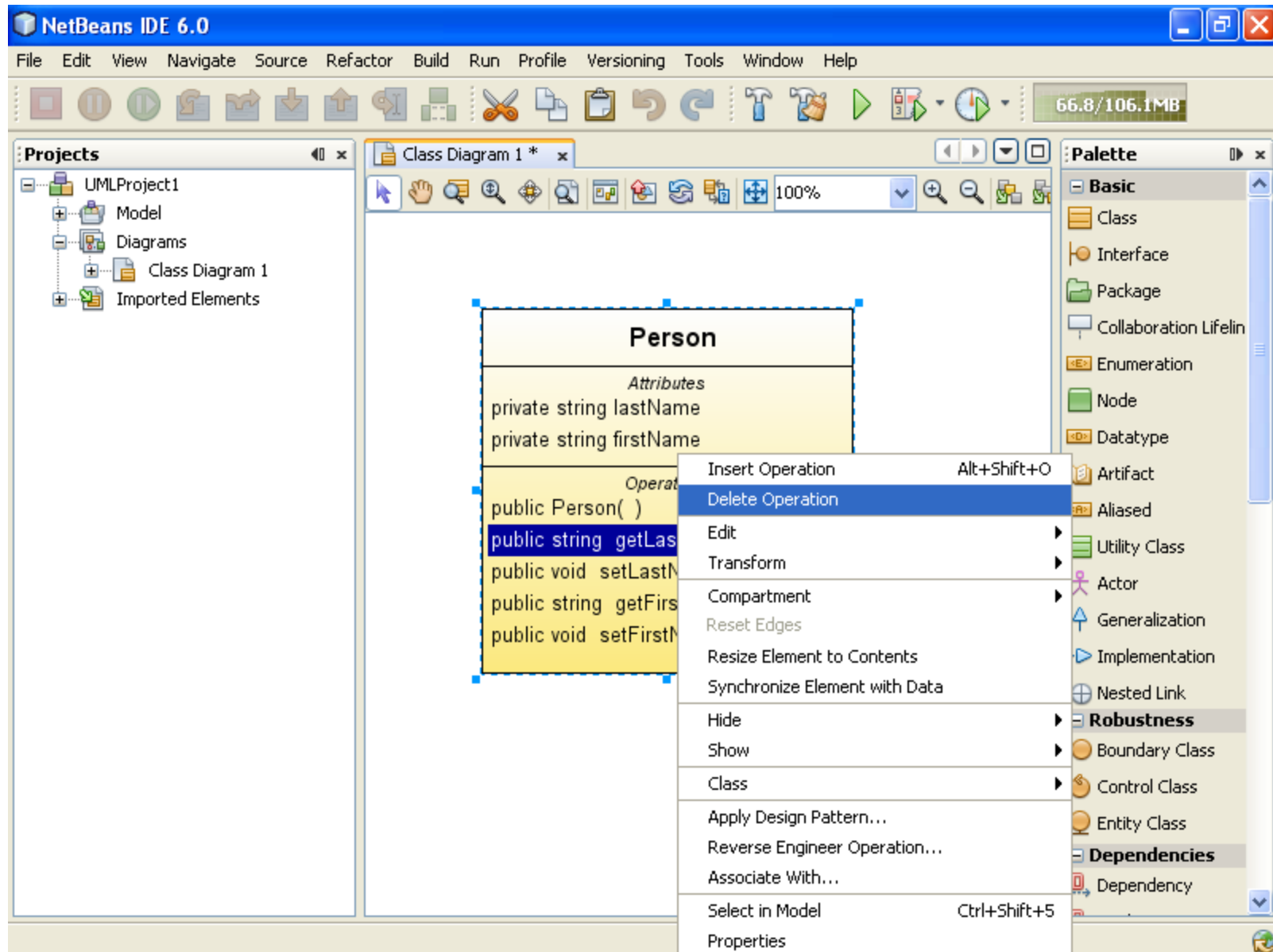
C++: string (lower case)

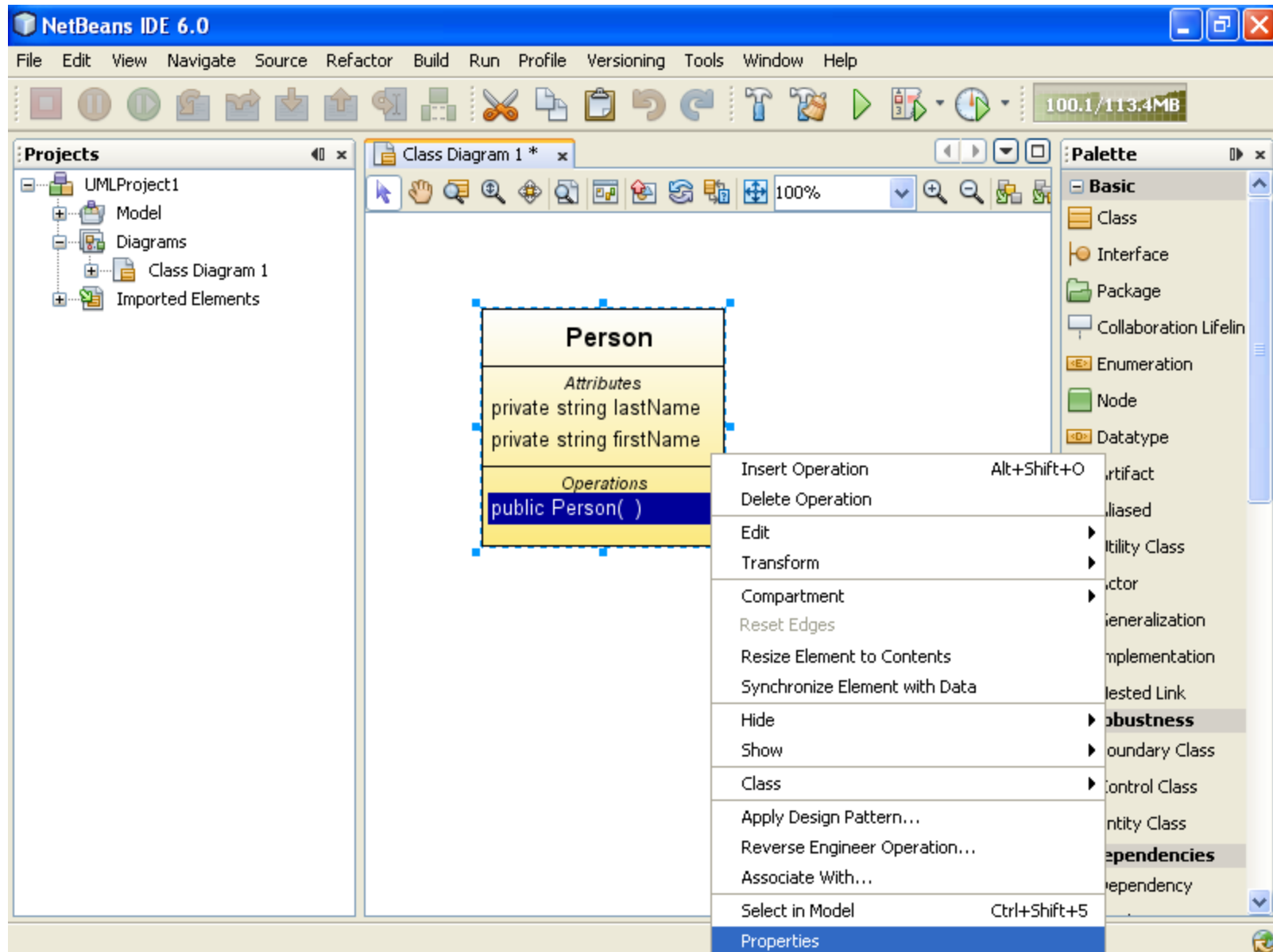
Java: String (upper case)

Both languages are case sensitive









Person - Properties

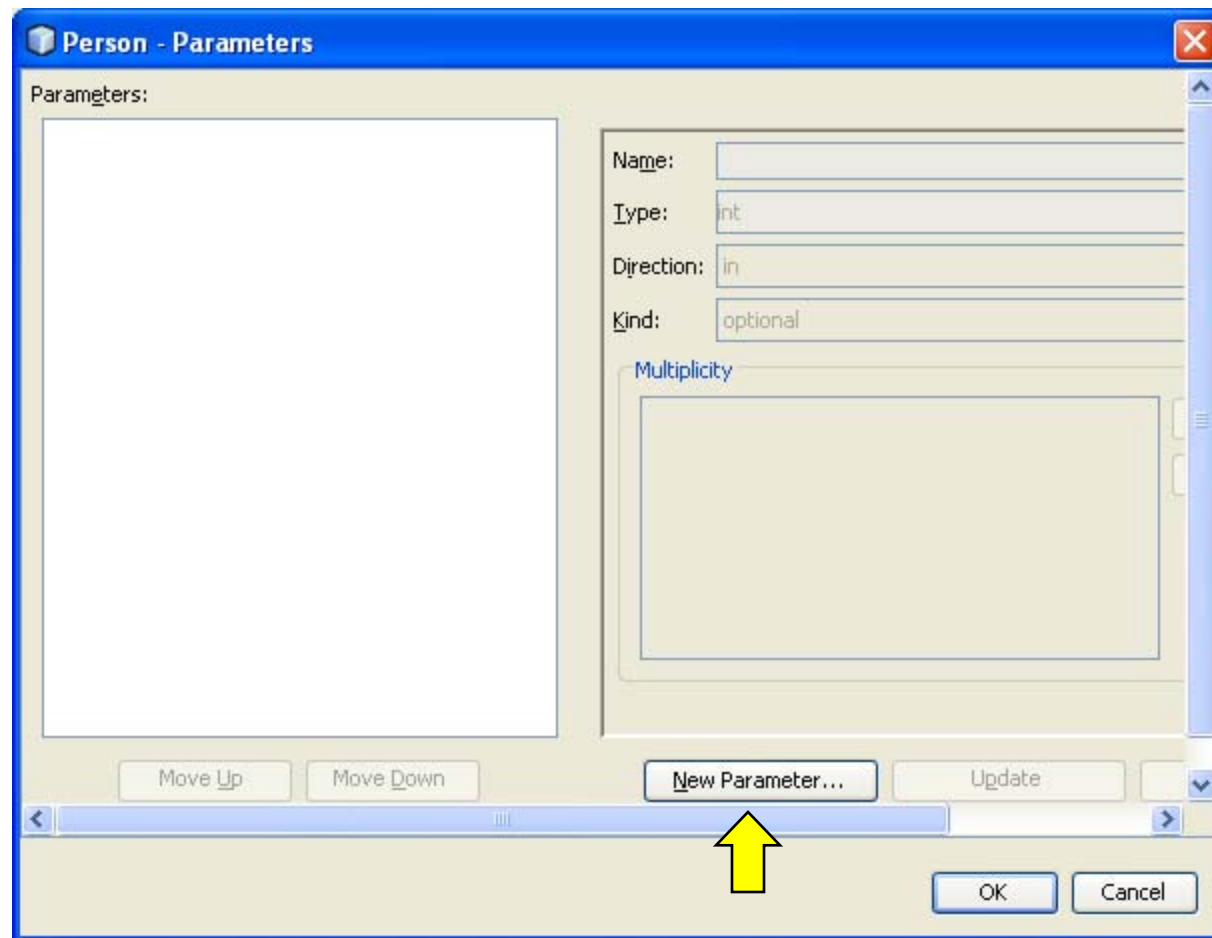
Operation

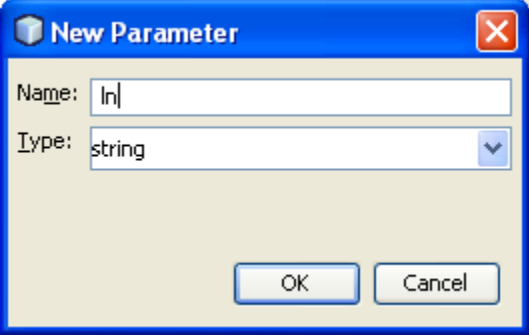
Name	Person
Alias	Person
Return Type	public Person()
Parameters	public Person()
Visibility	public
Documentation	
Stereotypes	
Tagged Values	
Abstract	<input type="checkbox"/>
Final	<input type="checkbox"/>
Static	<input type="checkbox"/>
Native	<input type="checkbox"/>
Strict FP	<input type="checkbox"/>
Concurrency	sequential
Query	<input type="checkbox"/>
Raised Exceptions	
Constraints	
Redefined	<input type="checkbox"/>

Parameters

Close







A screenshot of a 'New Parameter' dialog box. The dialog has a blue title bar with a cube icon and a close button. It contains two input fields: 'Name' with the text 'ln' and 'Type' with a dropdown menu showing 'string'. At the bottom are 'OK' and 'Cancel' buttons.

New Parameter

Name:

Type:

Person - Parameters

Parameters:

string In

Name: In

Type: string

Direction: in

Kind: byValue

Multiplicity

Lower	Upper	Collection Type
-------	-------	-----------------

Add Range

Remove Range

Move Up Move Down New Parameter... Update Remove

OK Cancel

Person - Parameters

Parameters:

string In	Name: fn	Type: string	Direction: in	Kind: byValue
string fn				

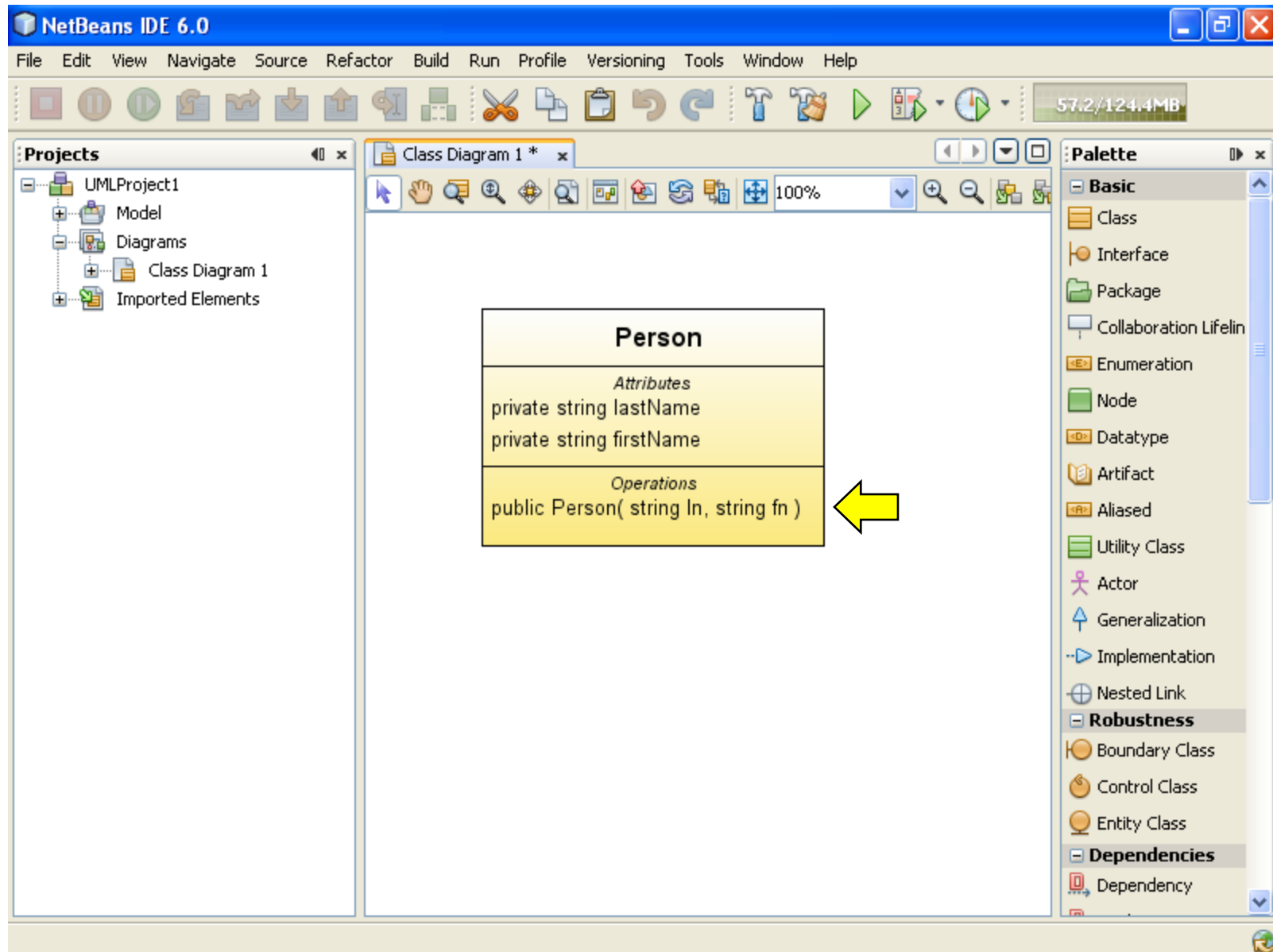
Multiplicity

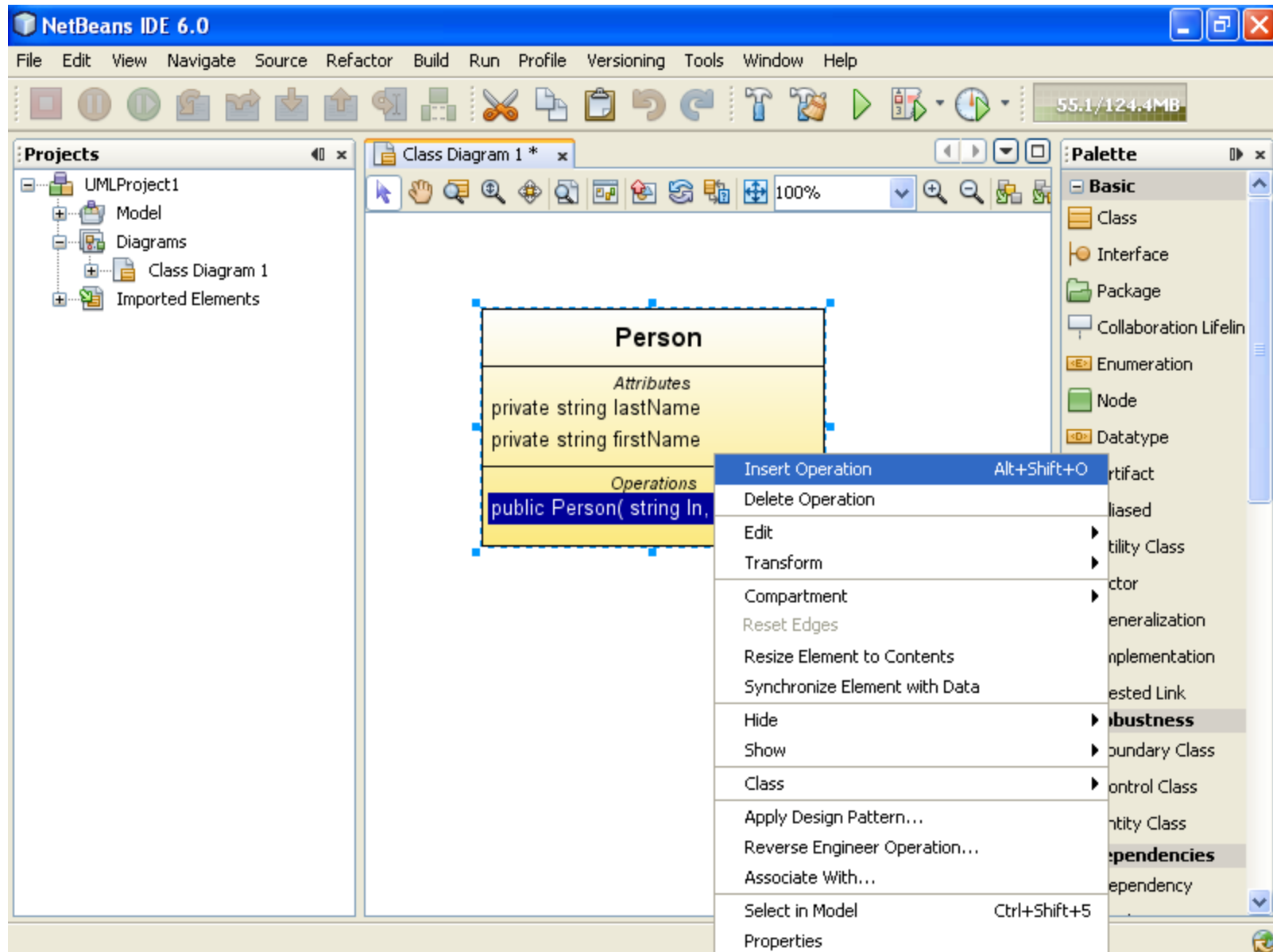
Lower	Upper	Collection Type

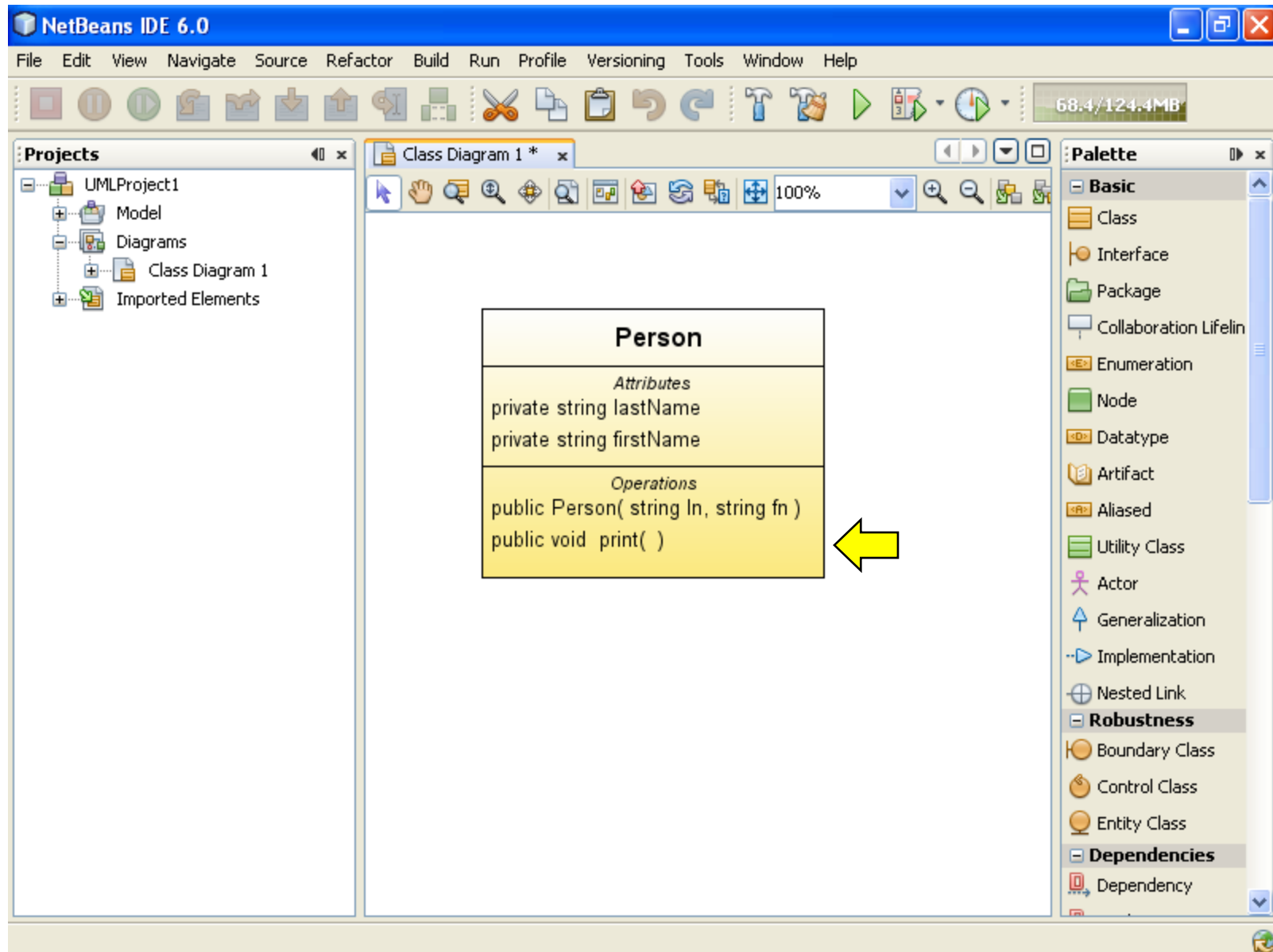
Add Range
Remove Range

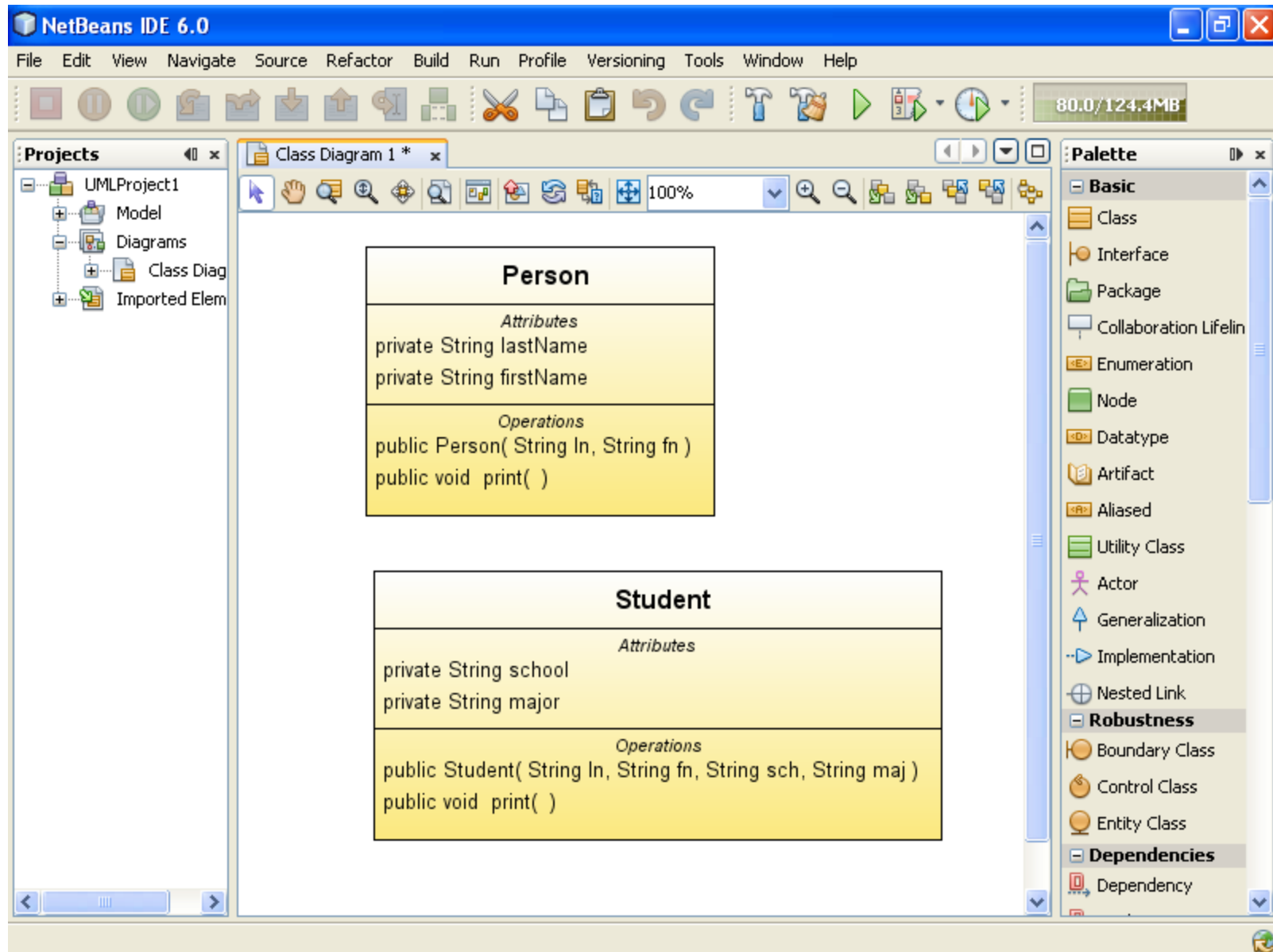
Move Up Move Down New Parameter... Update Remove

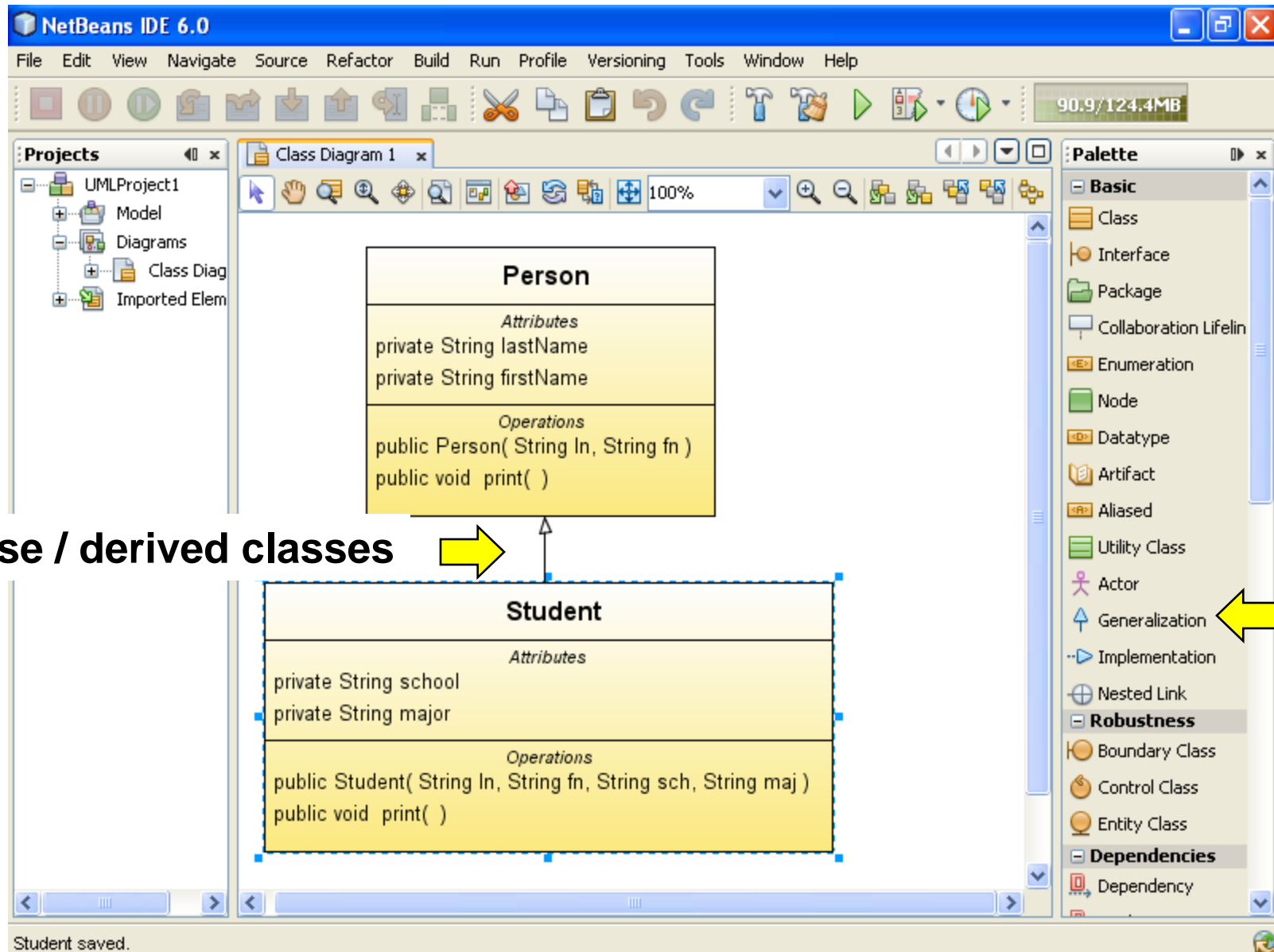
OK Cancel



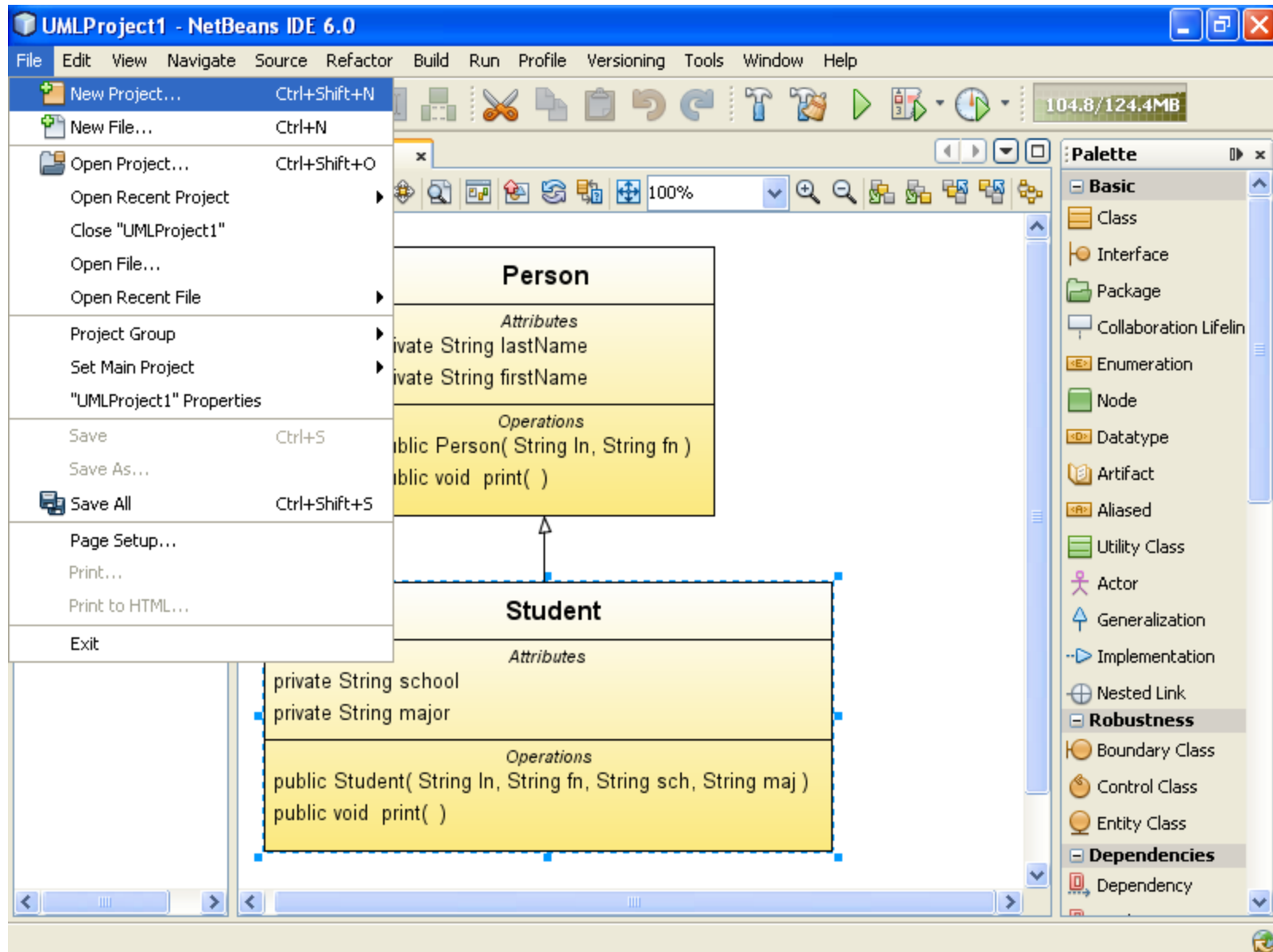




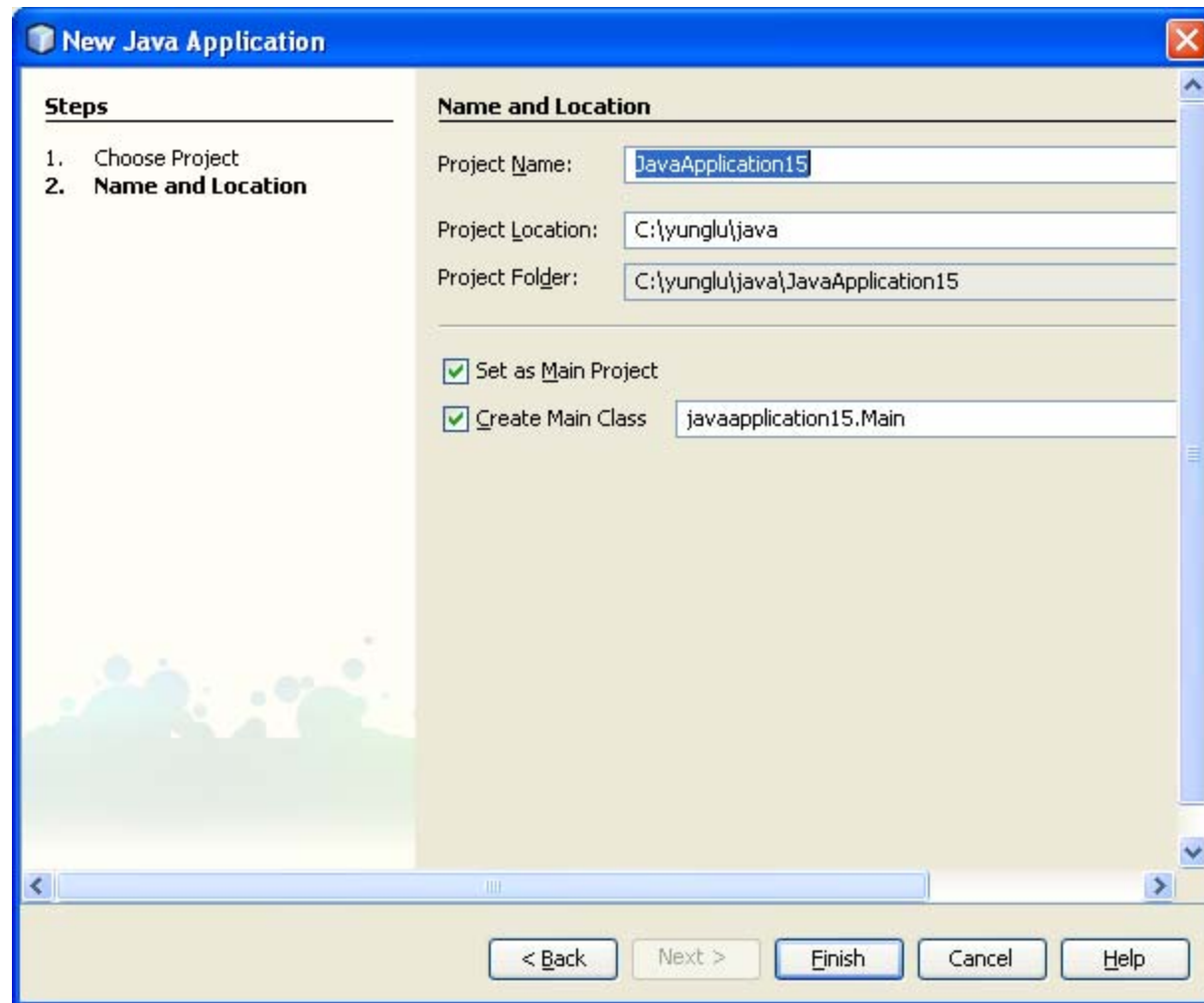


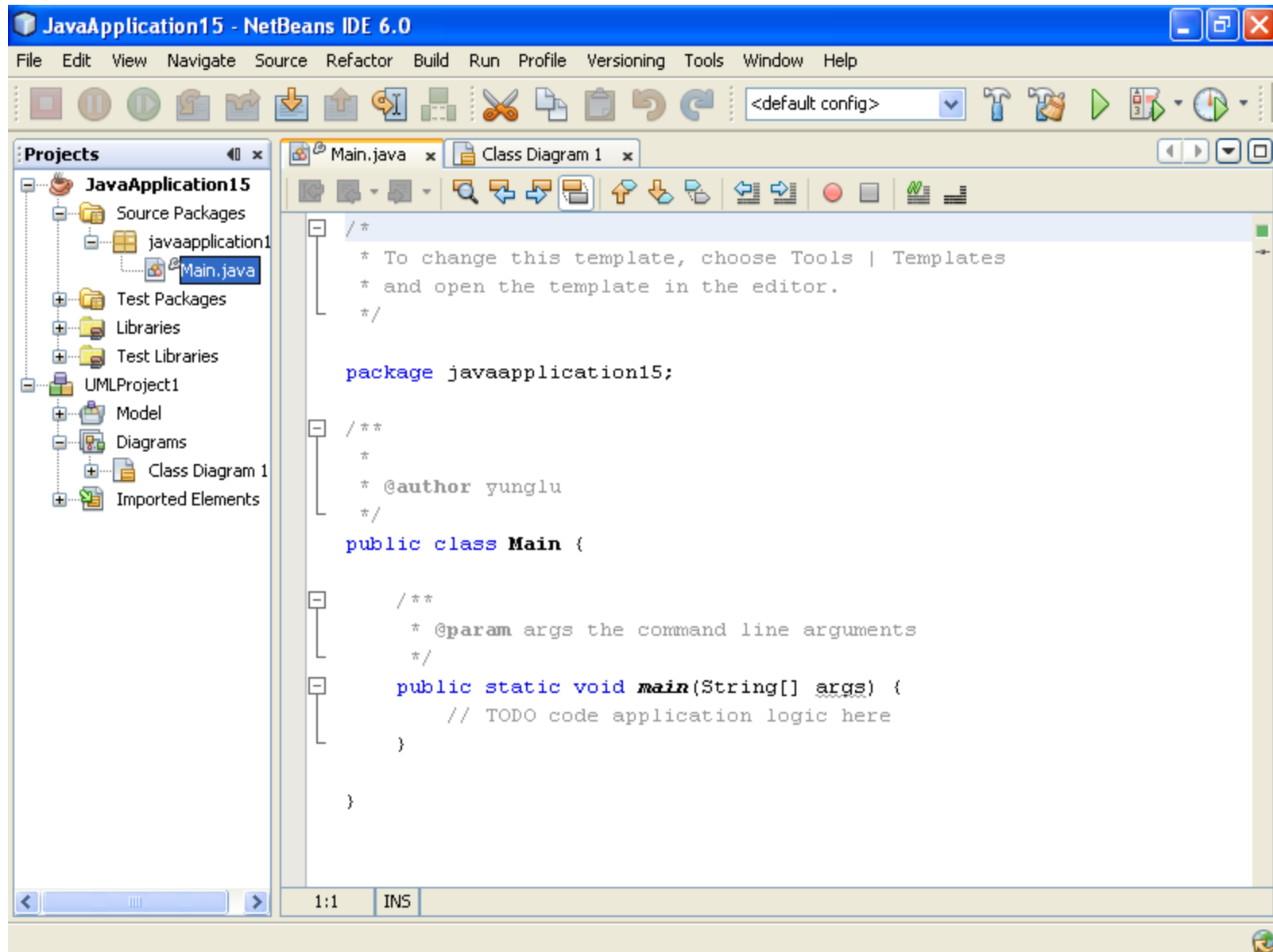


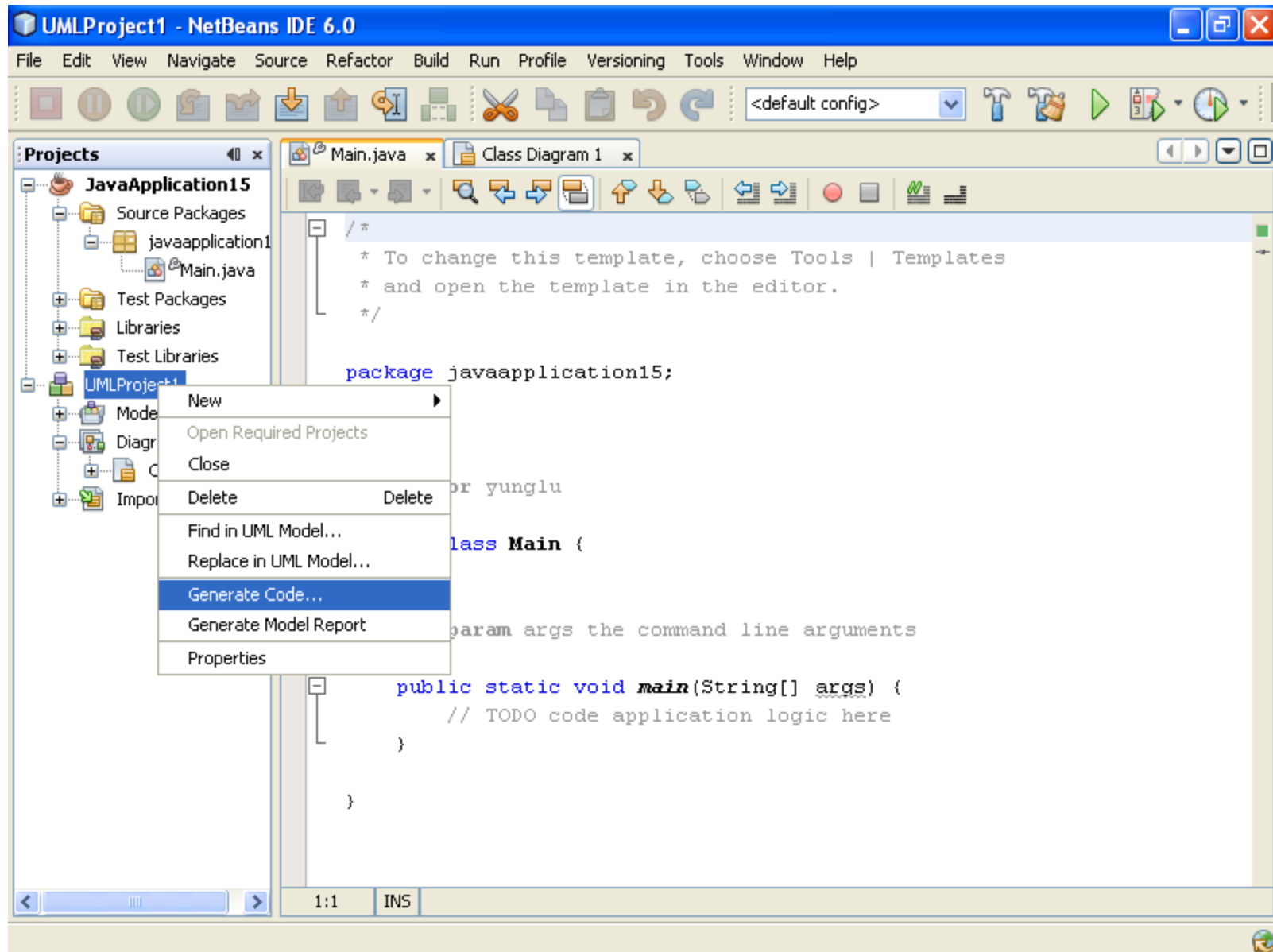
Generate Java Code from Class Diagram

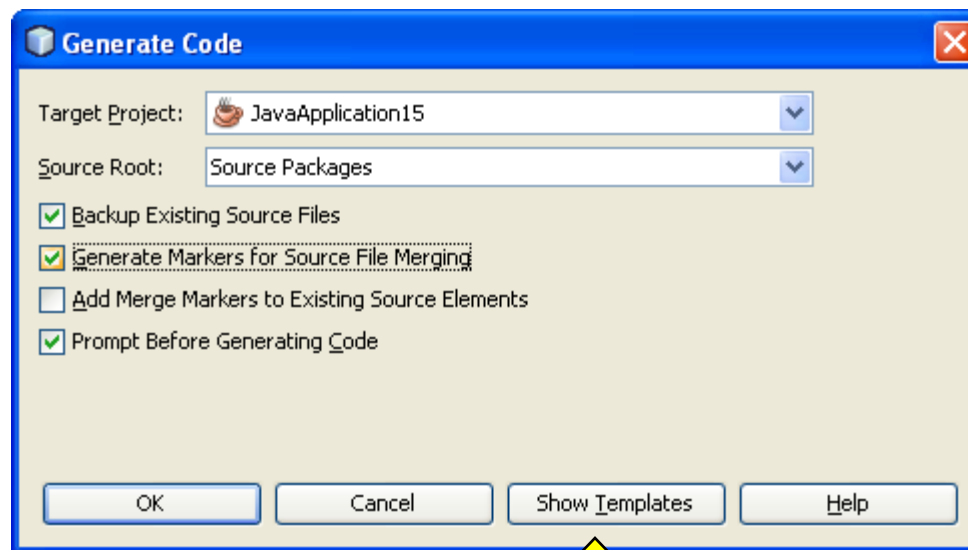


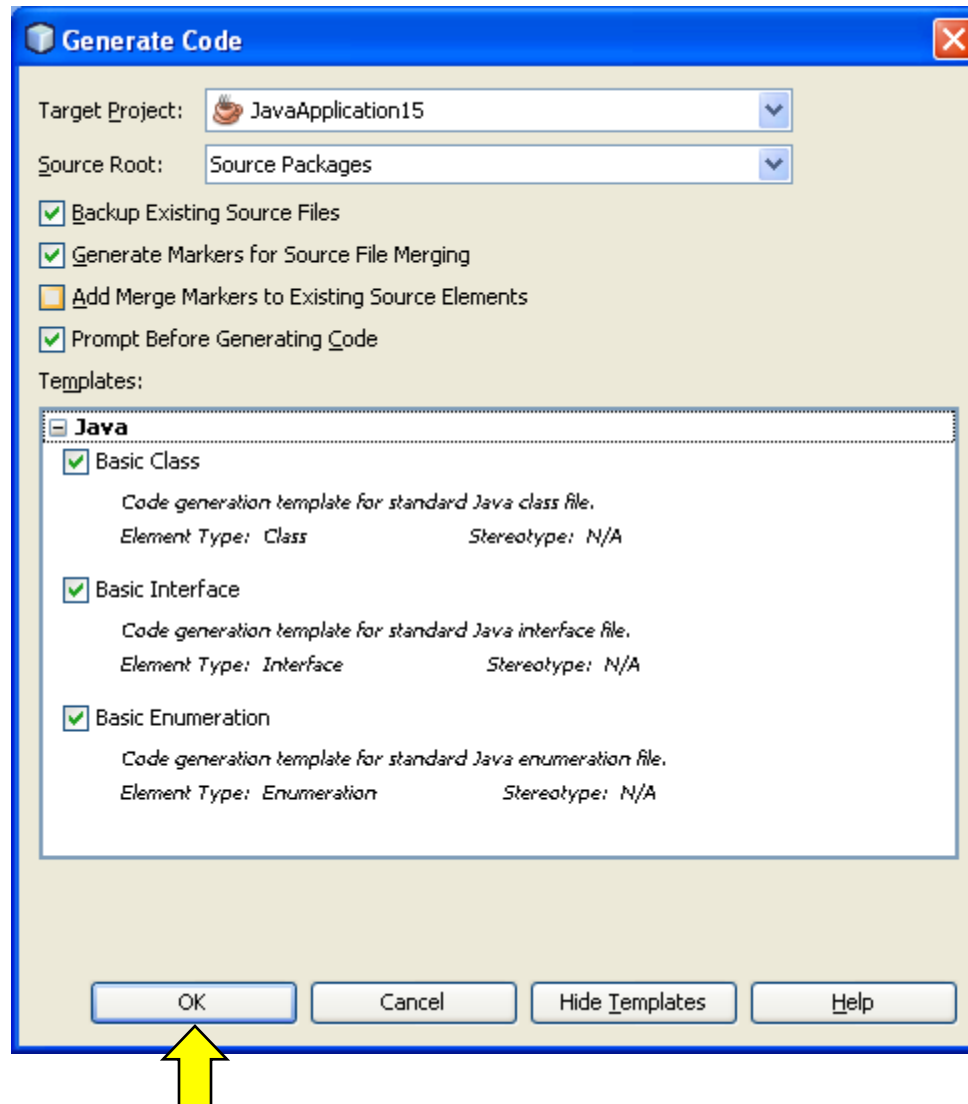


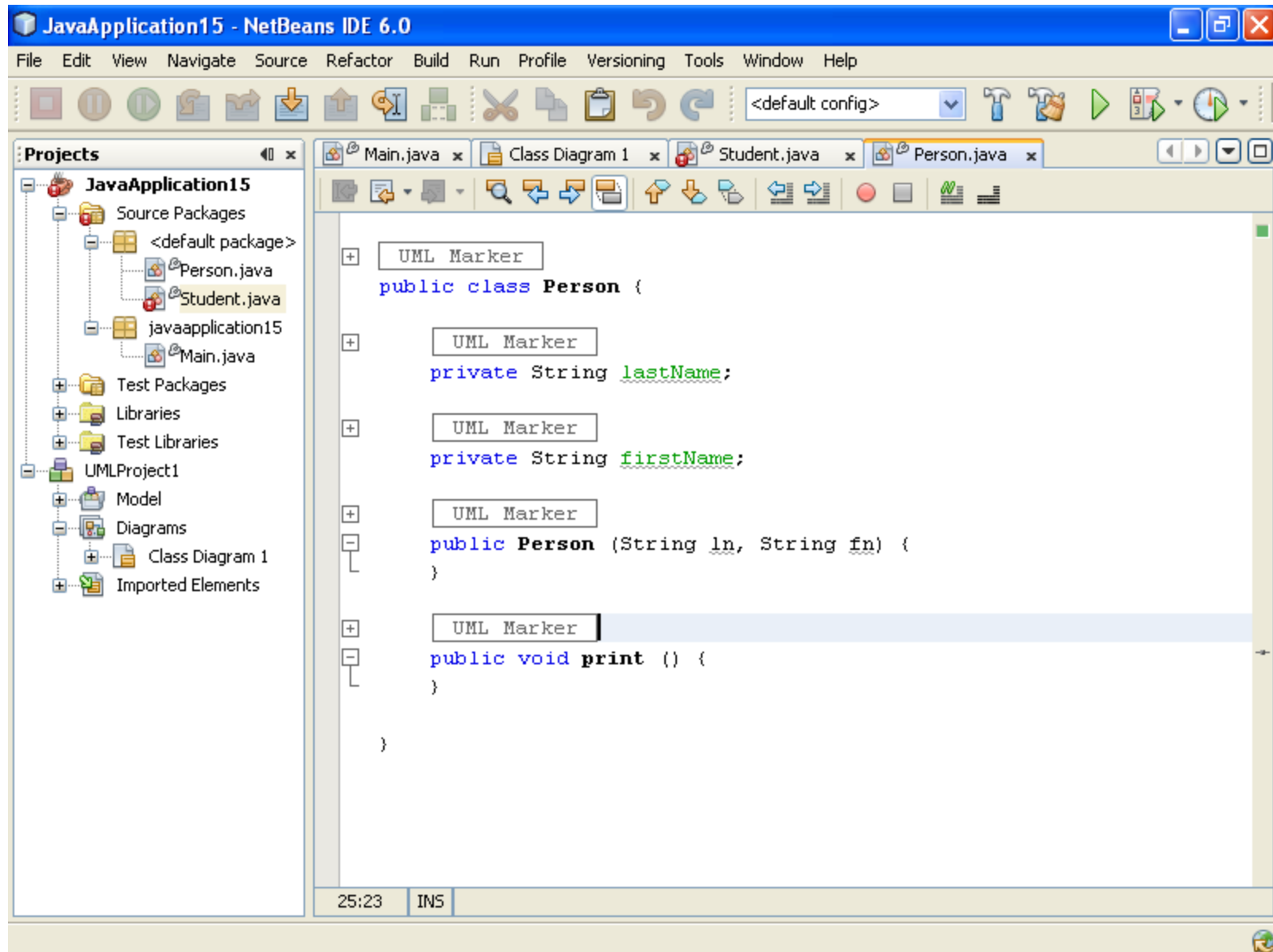


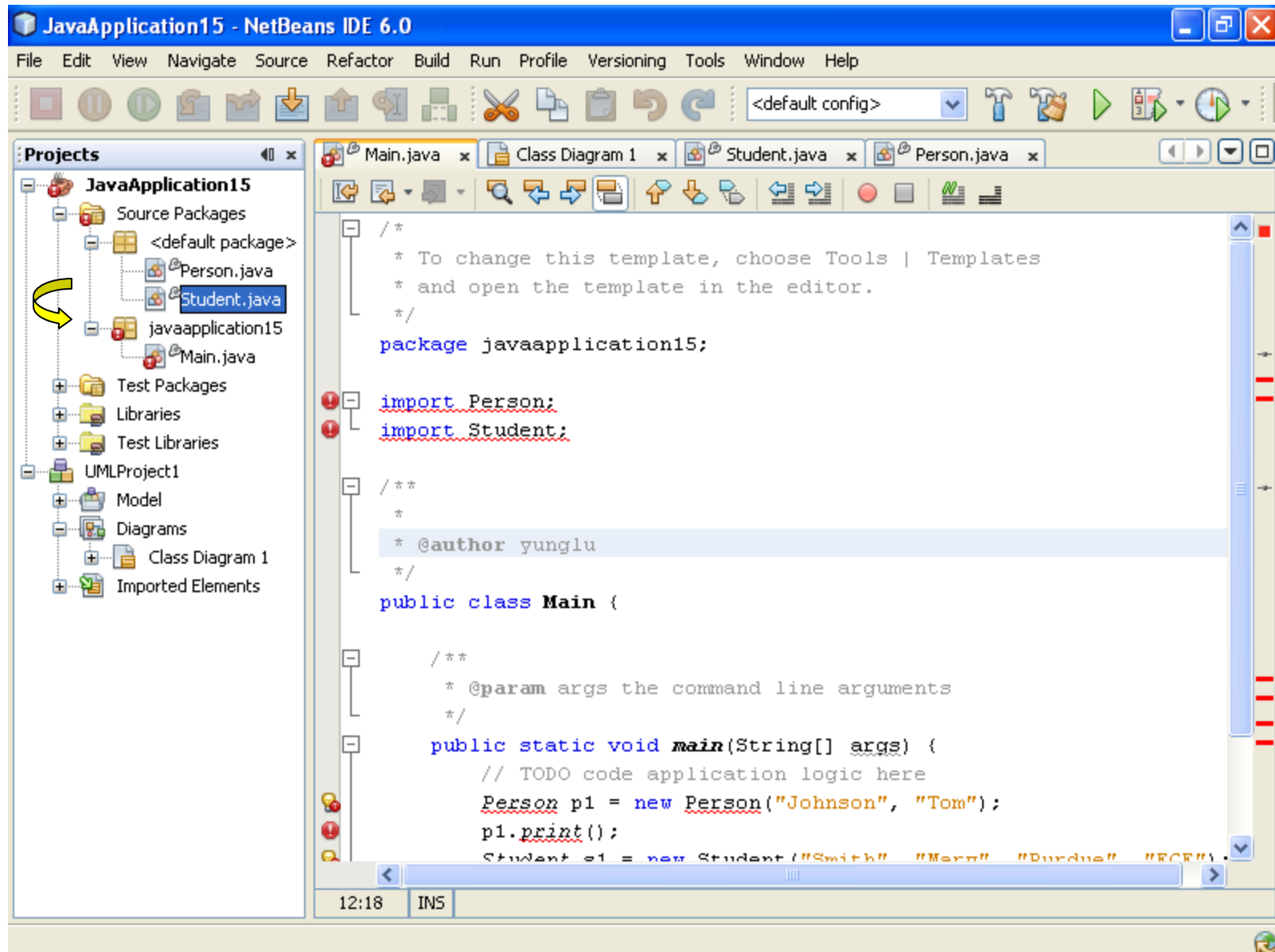


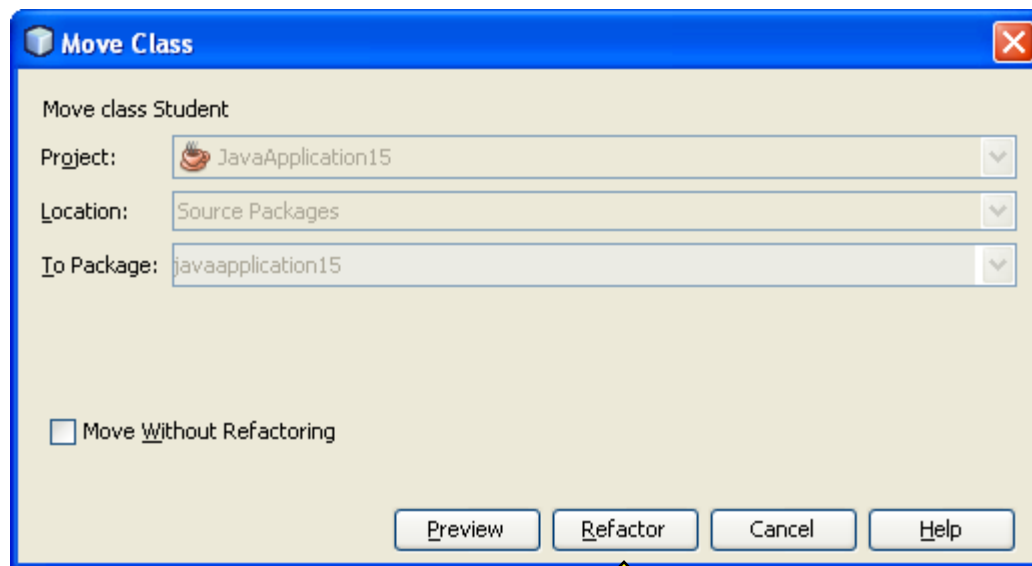


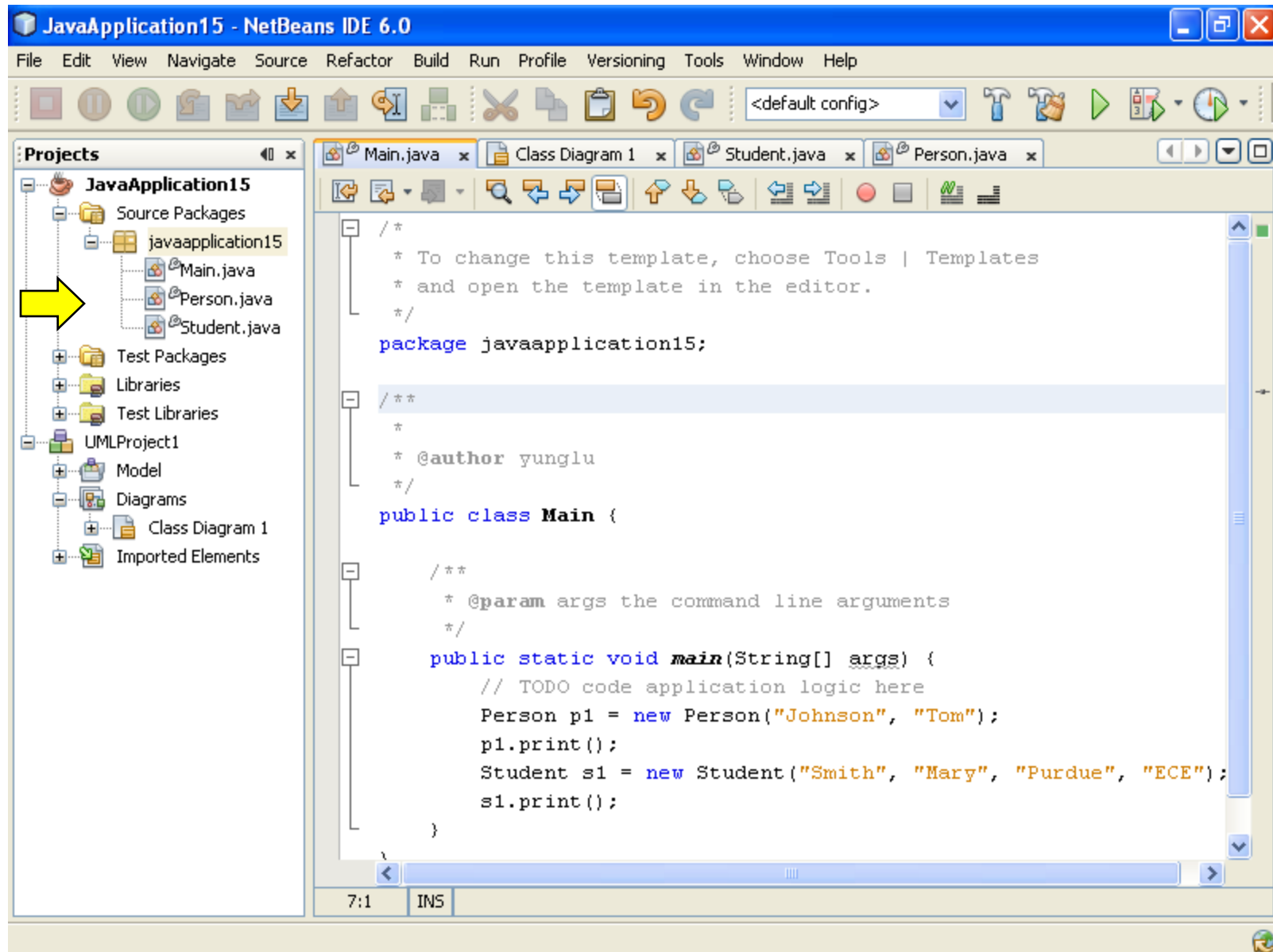


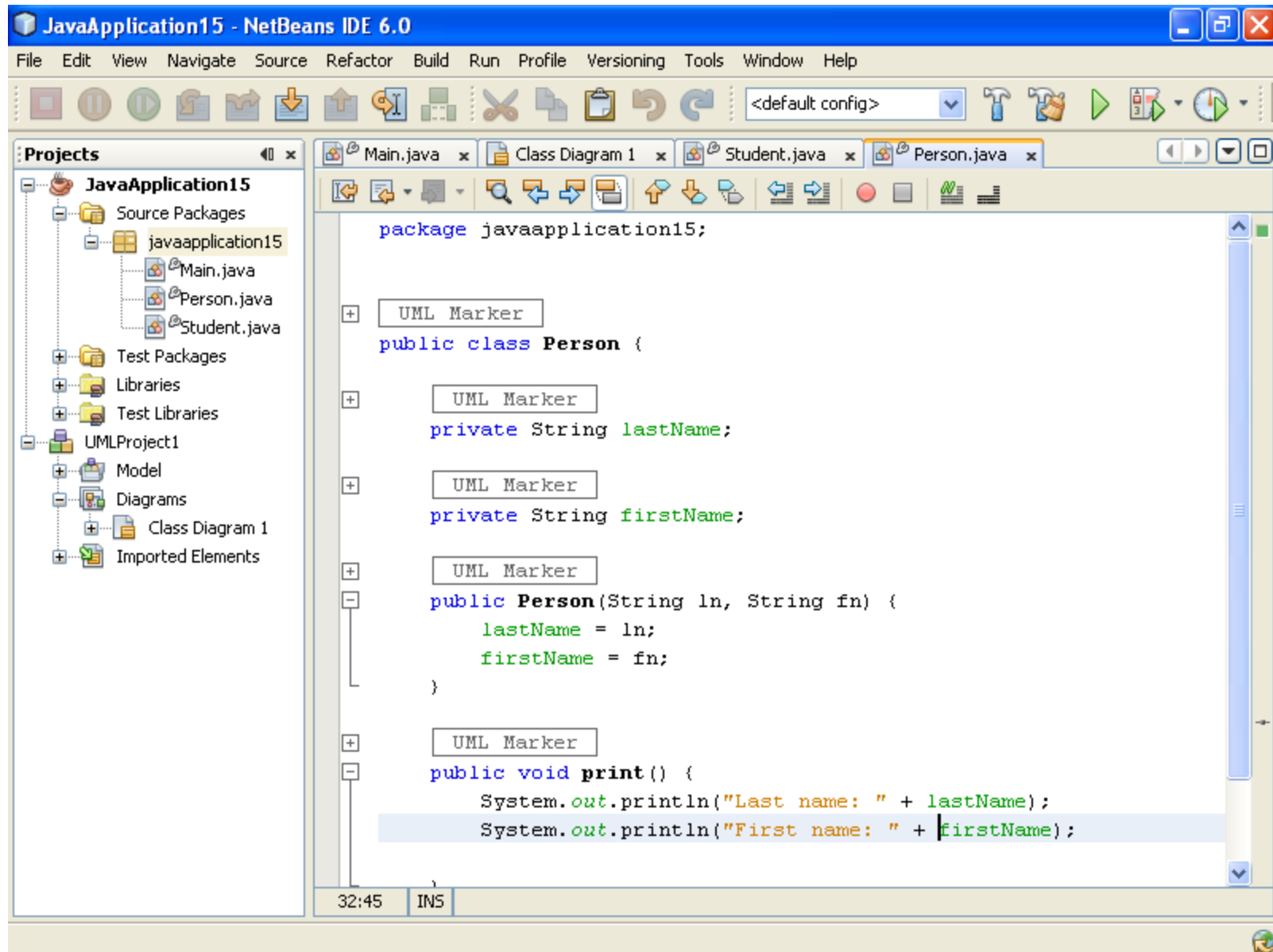


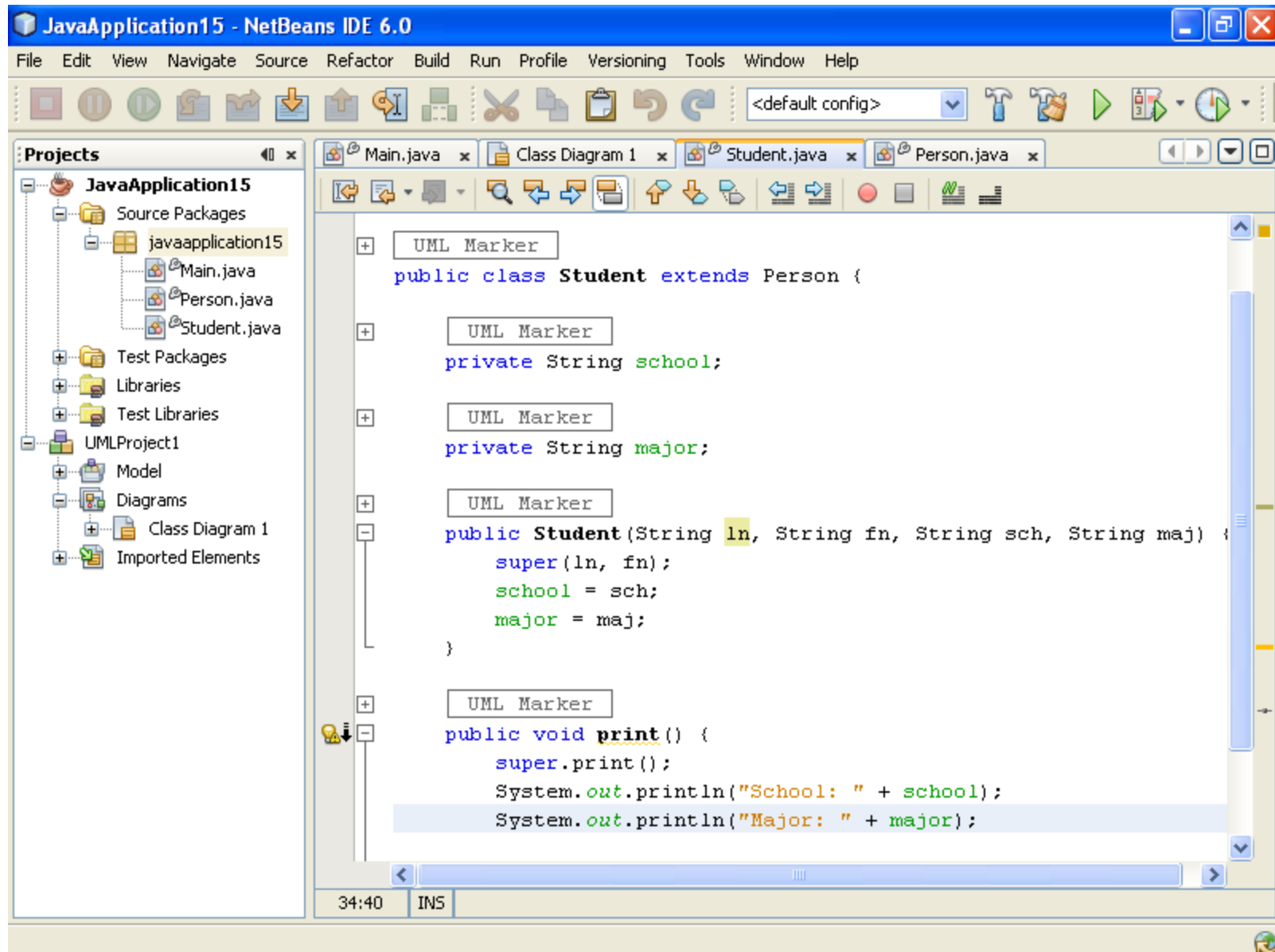


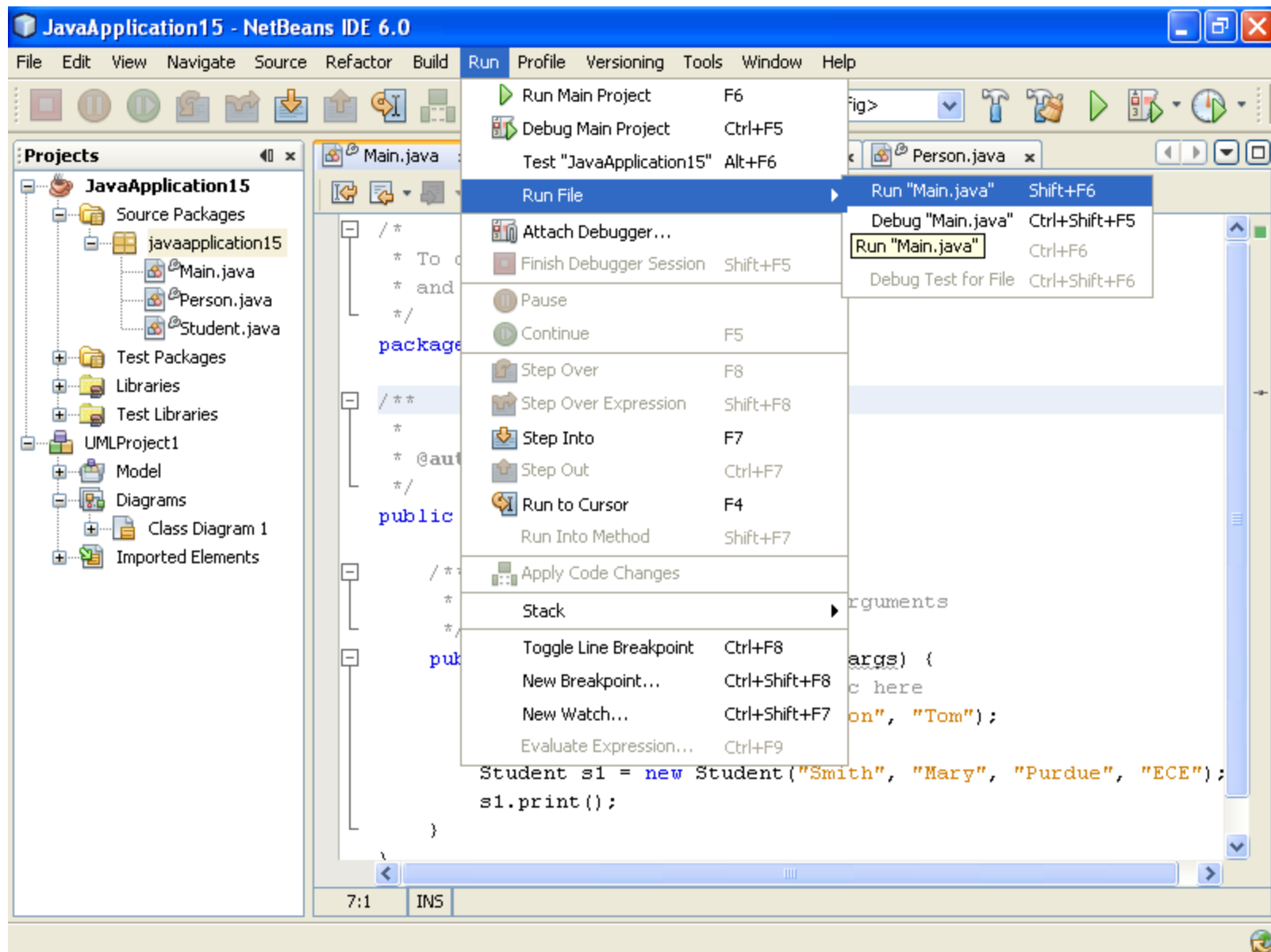


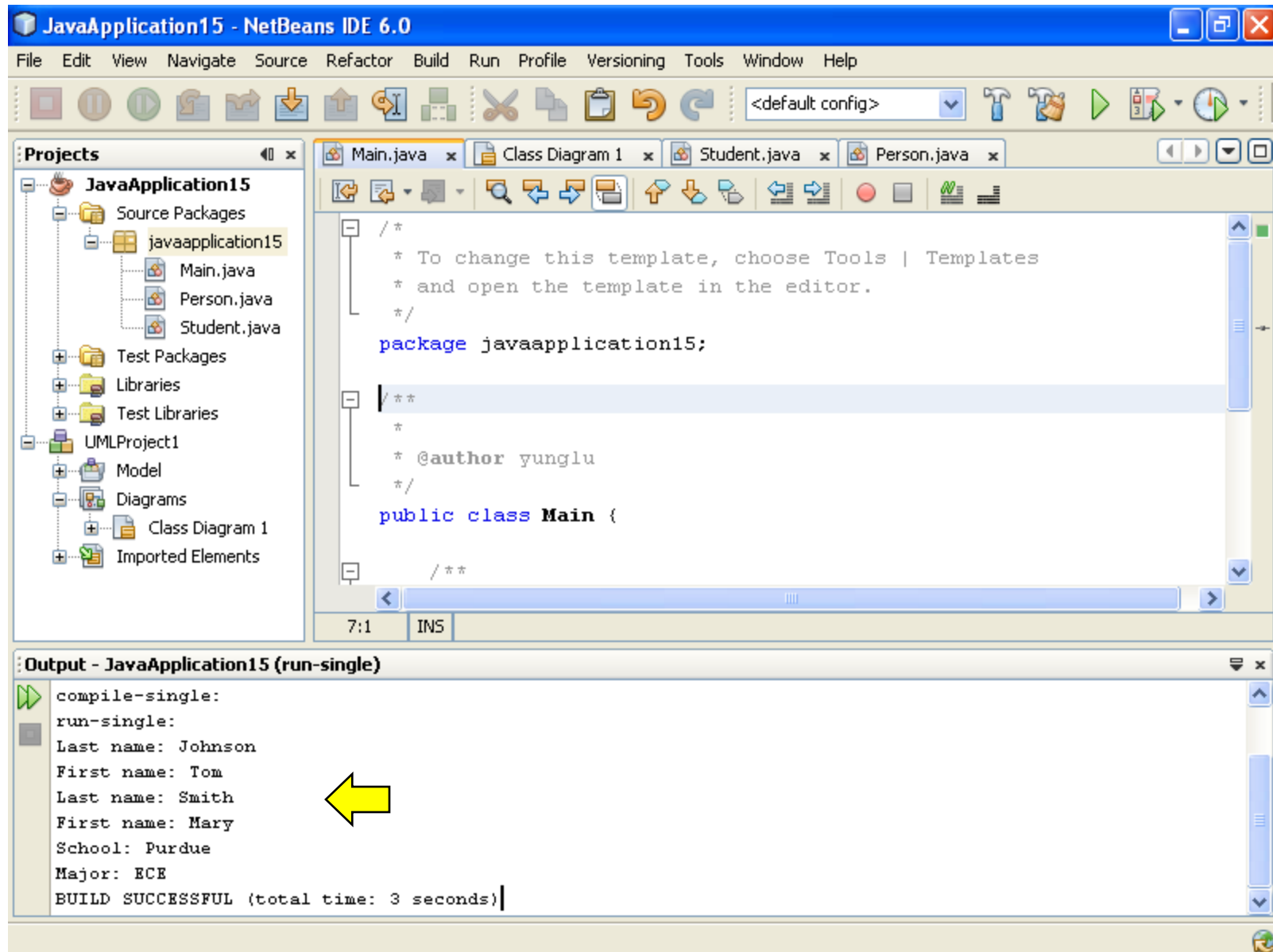






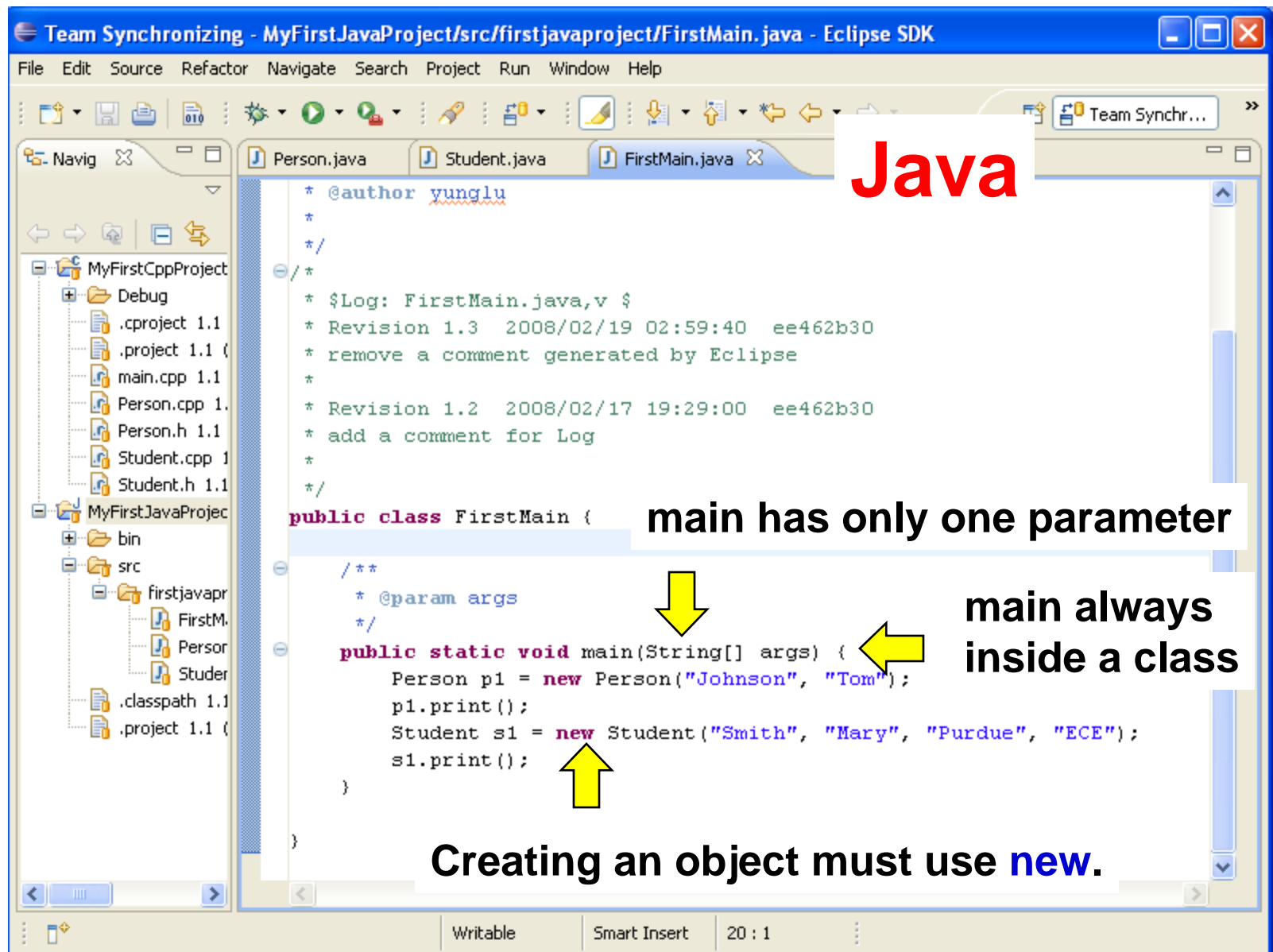






C++ and Java Syntax

C++	Java
<code>int main(int argc, char * argv[])</code>	<code>public static void main(String[] args) {</code>
<code>Person p1("Johnson", "Tom");</code>	<code>Person p1 = new Person("Johnson", "Tom");</code>
<code>p1.print();</code>	<code>p1.print();</code>
<code>class Person { public: Person(string ln, string fn);</code>	<code>public class Person { public Person(String ln, String fn) {</code>
<code>const string p_lastName;</code>	<code>final String p_lastName;</code>
<code>class Student: public Person</code>	<code>class Student extends Person</code>



Team Synchronizing - MyFirstCppProject/main.cpp - Eclipse SDK

File Edit Refactor Navigate Search Project Run Window Help

C++

main has two parameters

main always outside a class

```
#include "Person.h"
#include "Student.h"
#include <iostream>
using namespace std;
int main(int argc, char * argv[]) {
    Person p1("Johnson", "Tom");
    p1.print();
    Student s1("Smith", "Mary", "Purdue", "ECE");
    s1.print();
    return 0;
}
```

Creating an object does not use **new**.
(C++ can also use new to create an object; this will be discussed later.)

MyFirstCppProject

- Debug
- .cproject 1.1
- .project 1.1
- main.cpp 1.1
- Person.cpp 1.1
- Person.h 1.1
- Student.cpp 1.1
- Student.h 1.1

MyFirstJavaProject

- bin
- src
 - firstjavapr
 - FirstM.
 - Person
 - Studer
- .classpath 1.1
- .project 1.1

Writable Smart Insert 12 : 1

Encapsulation

- Both C++ and Java provide three levels of protection:
 - private (default), accessible to only the class
 - protected, accessible to the class and its derived classes
 - public, no restriction
- The three protection levels apply to both attributes (also called member data) and functions (also called member function or methods)
- In general, **keep** as many attributes and functions **private** as possible. Allow only limited accesses.

Team Synchronizing - MyFirstJavaProject/src/firstjavaproject/Person.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Java

```
package firstjavaproject;

public class Person {
    final String p_lastName;
    final String p_firstName;

    public Person(String ln, String fn) {
        p_lastName = ln;
        p_firstName = fn;
    }

    public void print() {
        System.out.println("Last name: " + p_lastName);
        System.out.println("First name: " + p_firstName);
    }
}
```

String (uppercase)

public function

public return-type function

System.out.println to print

no return-type for constructor

no header file for Java

no destructor

no virtual

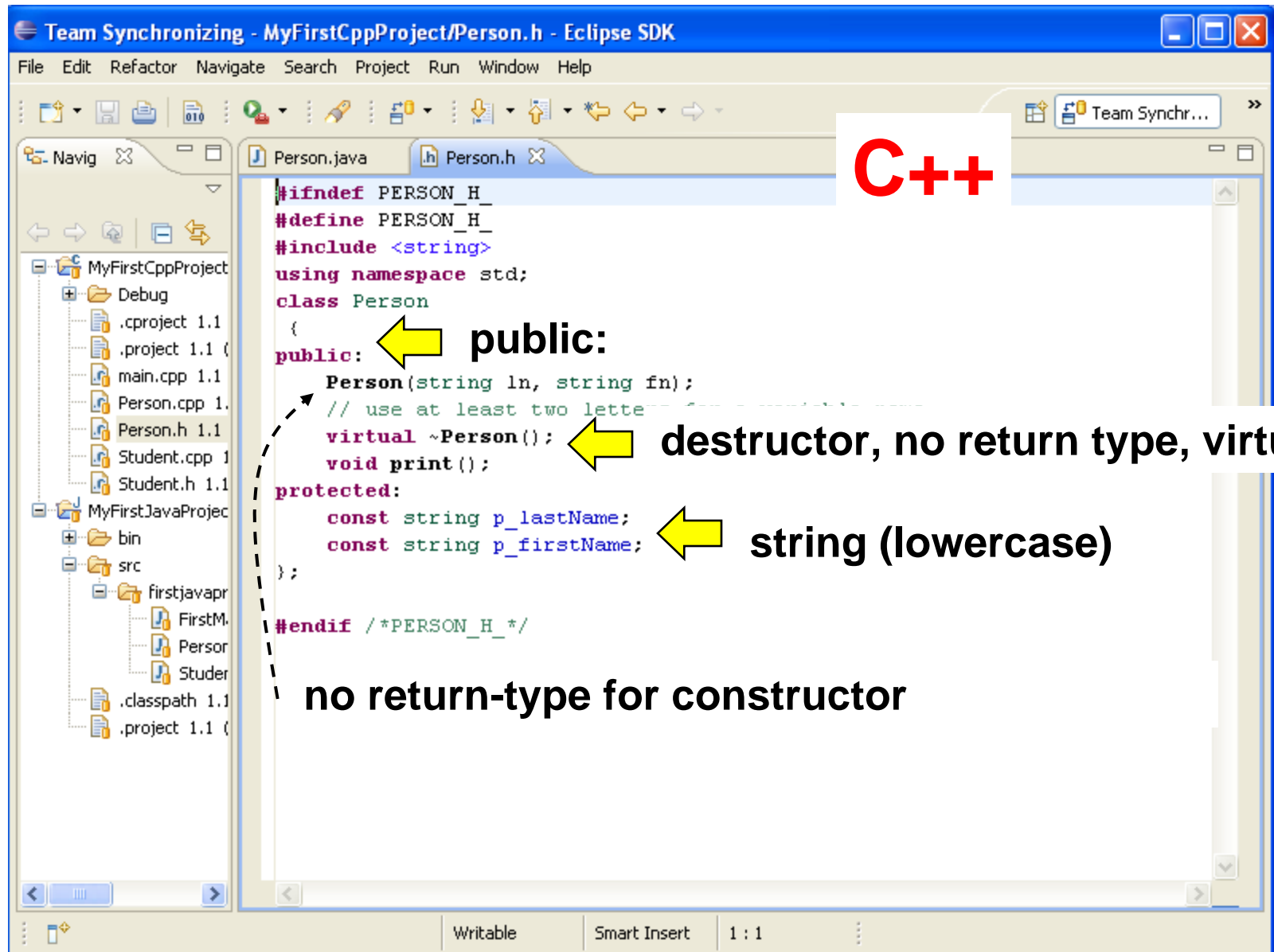
MyFirstCppProject

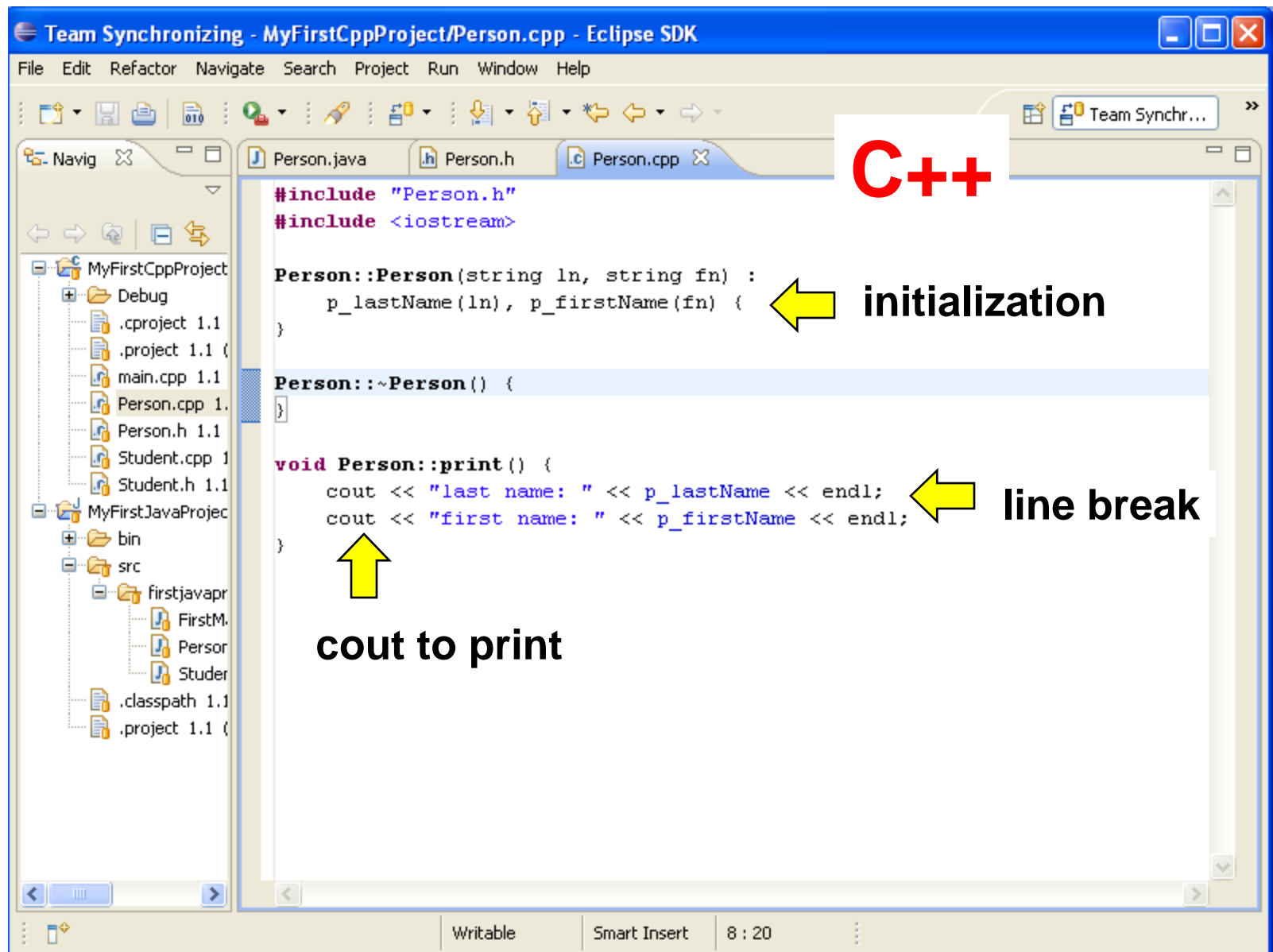
- Debug
- .cproject 1.1
- .project 1.1
- main.cpp 1.1
- Person.cpp 1.1
- Person.h 1.1
- Student.cpp 1.1
- Student.h 1.1

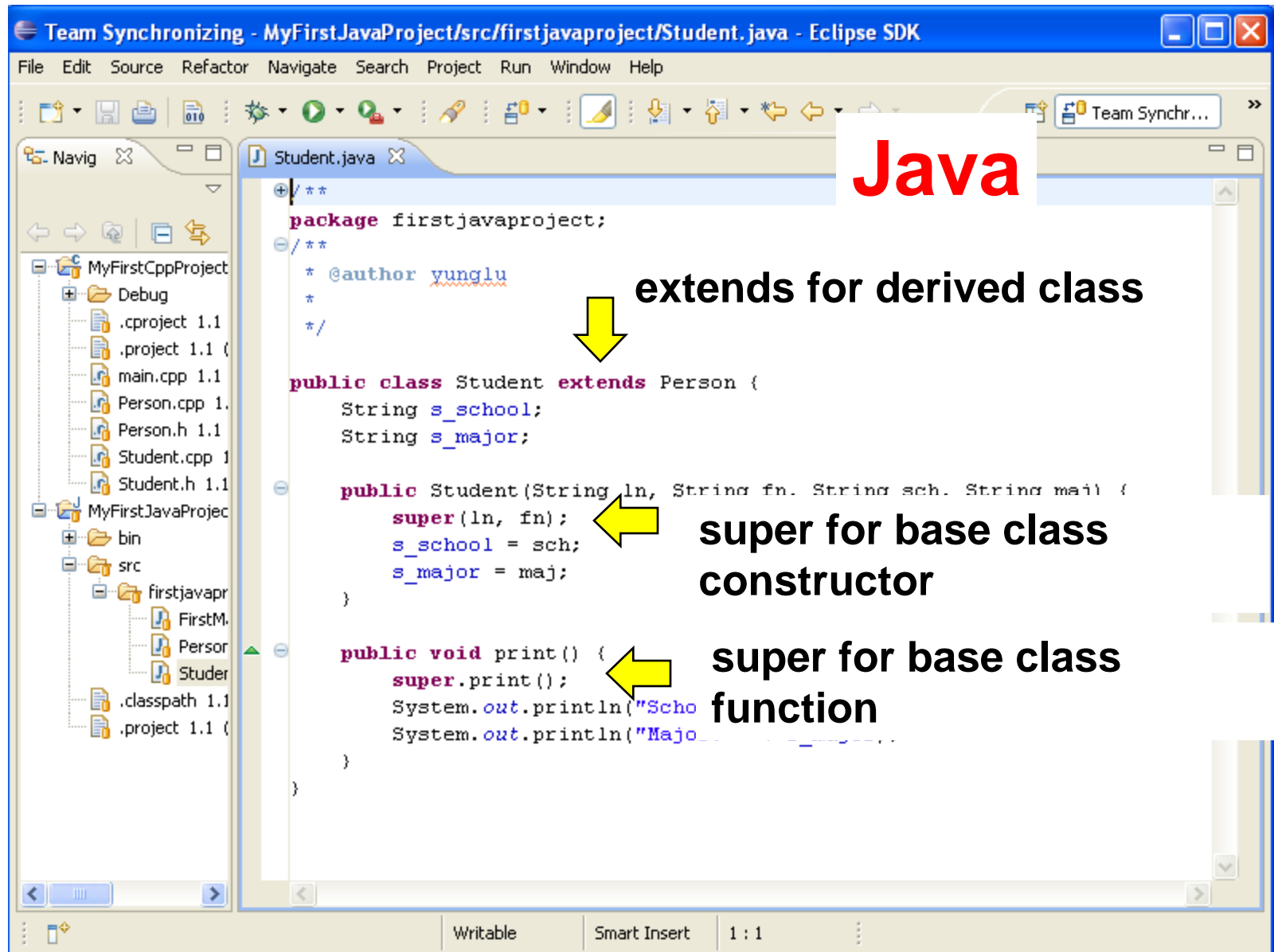
MyFirstJavaProject

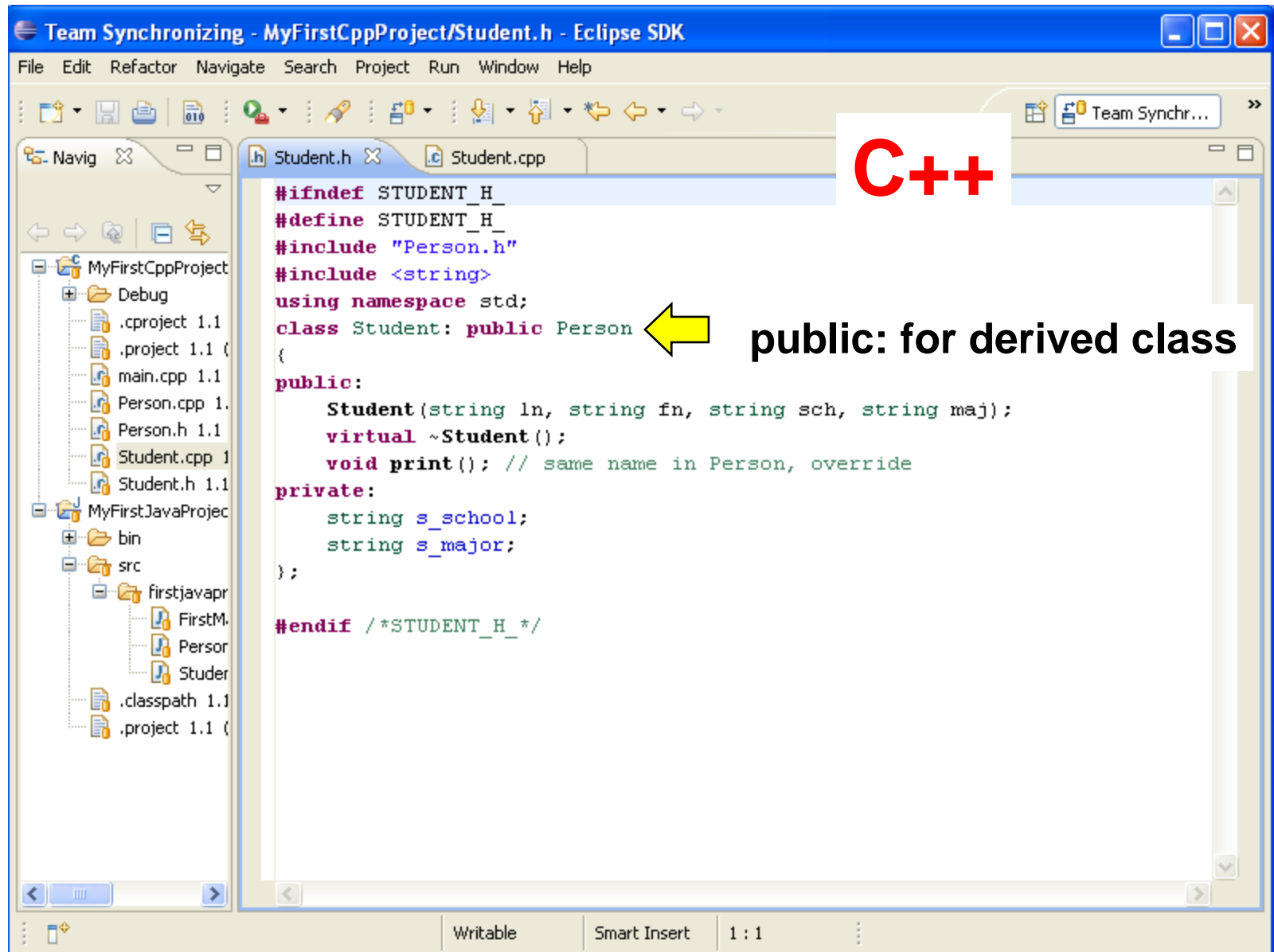
- bin
- src
 - firstjavapr
 - FirstM.
 - Person
 - Studer
- .classpath 1.1
- .project 1.1

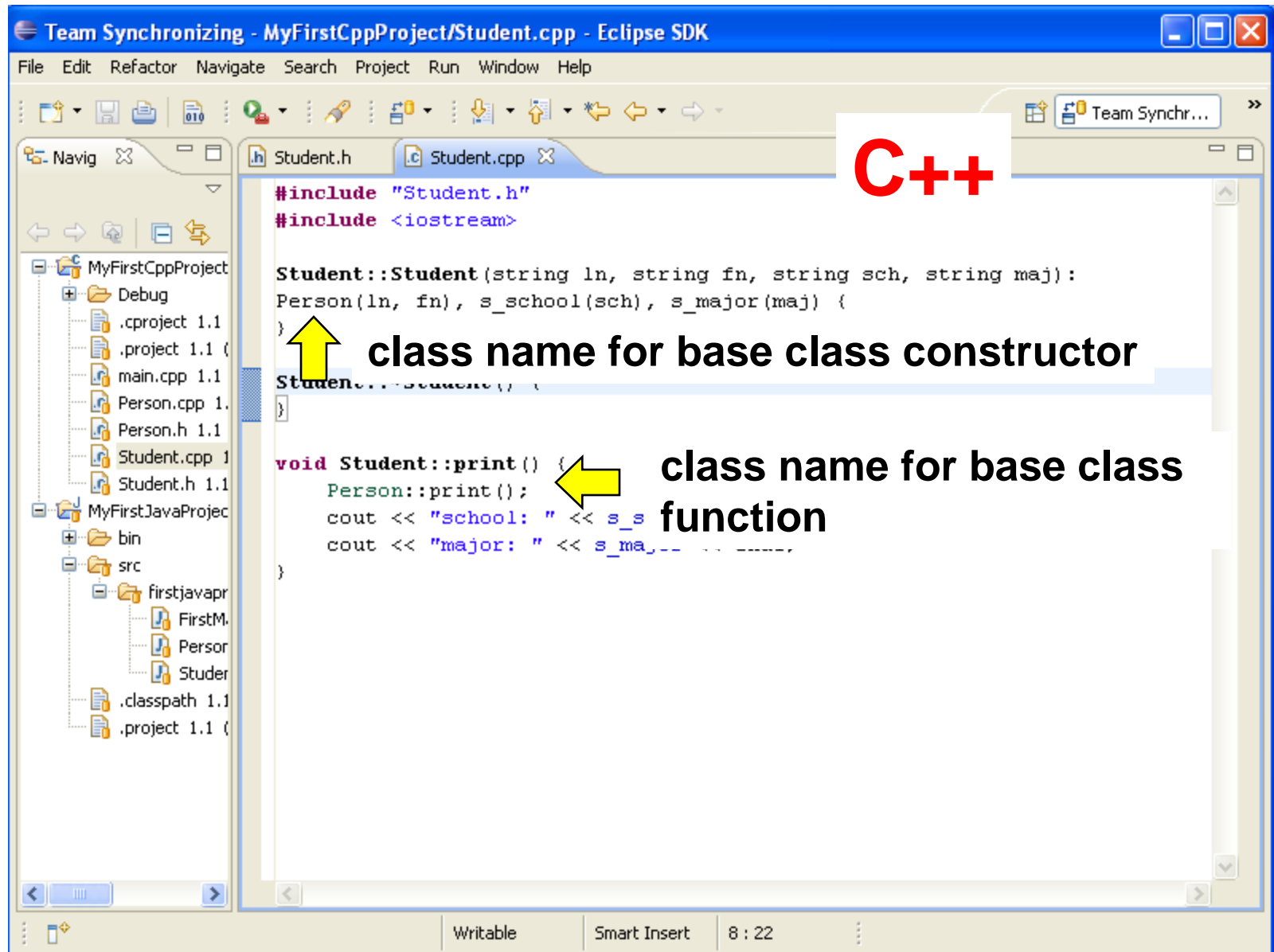
Writable Smart Insert 5 : 30











What is Method / Message

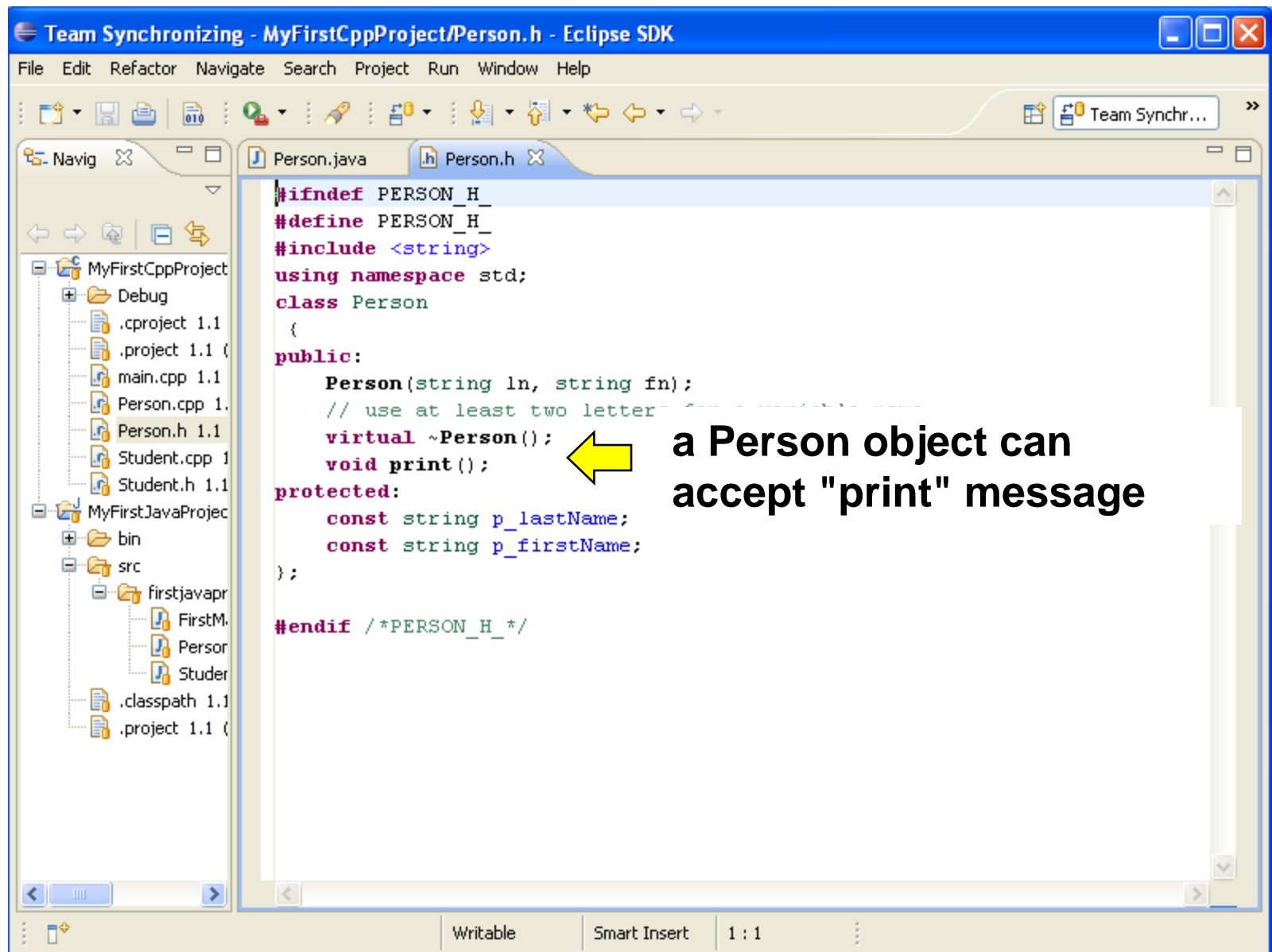
- A **message** is a request **to an object** to do something. An object can receive a message if it is declared as a method in a base class or the corresponding class.
- In OOP, many objects are created and they interact by “sending messages”, for example,
 - A Driver object sends a message “accelerate” to a MotorVehicle object.
 - An Instructor object sends a message “submitHomework” to a Student object.
 - A Caller object sends a message “callNumber” to a MobilePhone object.

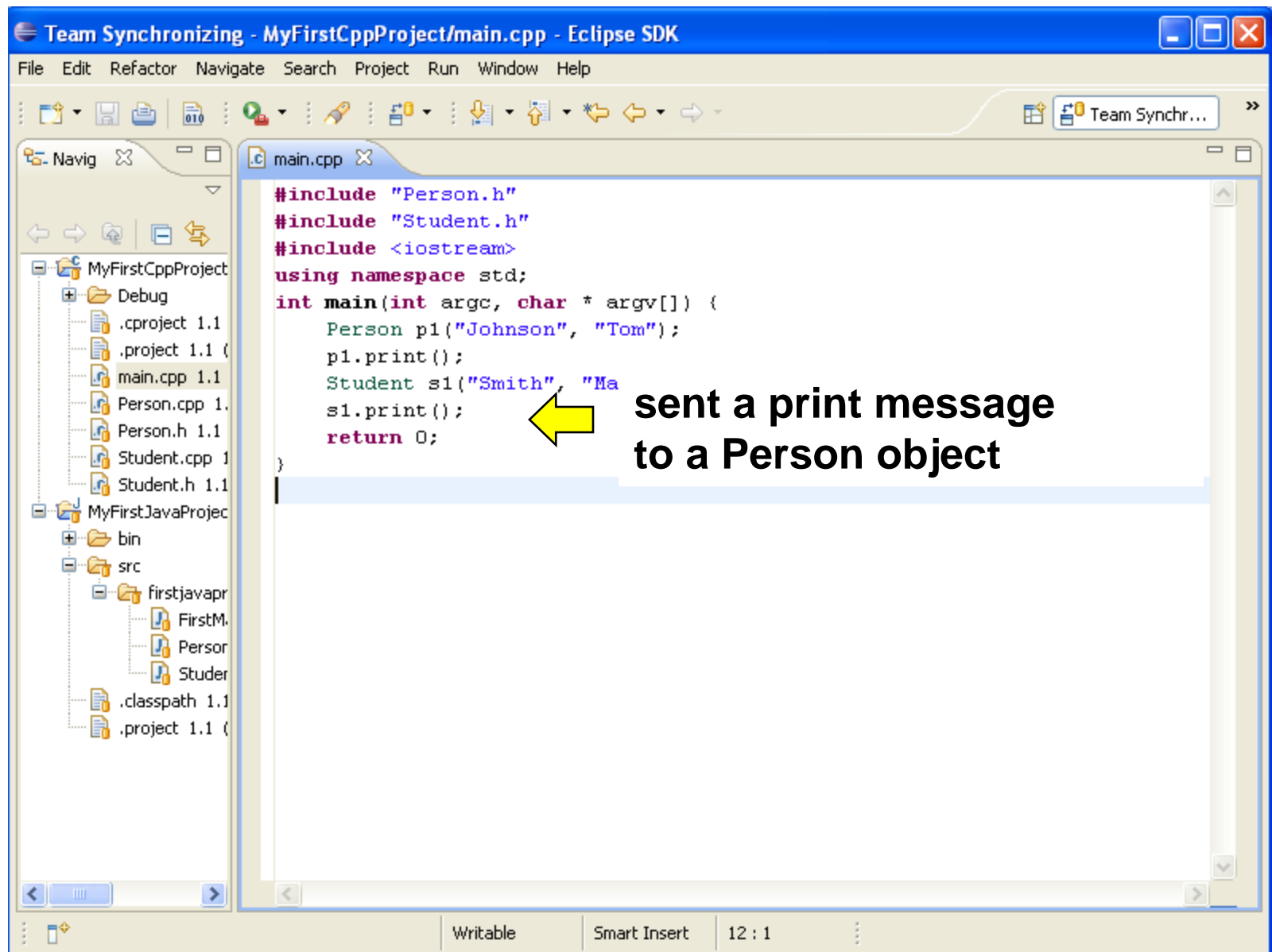
Message

- **not** a network concept
- the mechanism to interact with an object
 - ask a bridge about its length (no parameter)
 - turn on a light (no parameter)
 - accelerate a car (parameter = acceleration, m/s^2)
 - add a customer to a database (parameter = customer)
 - ask a customer of the credit number
- Disallowed messages are checked at **compile time**: compiler error if you ask a light bulb to accelerate.

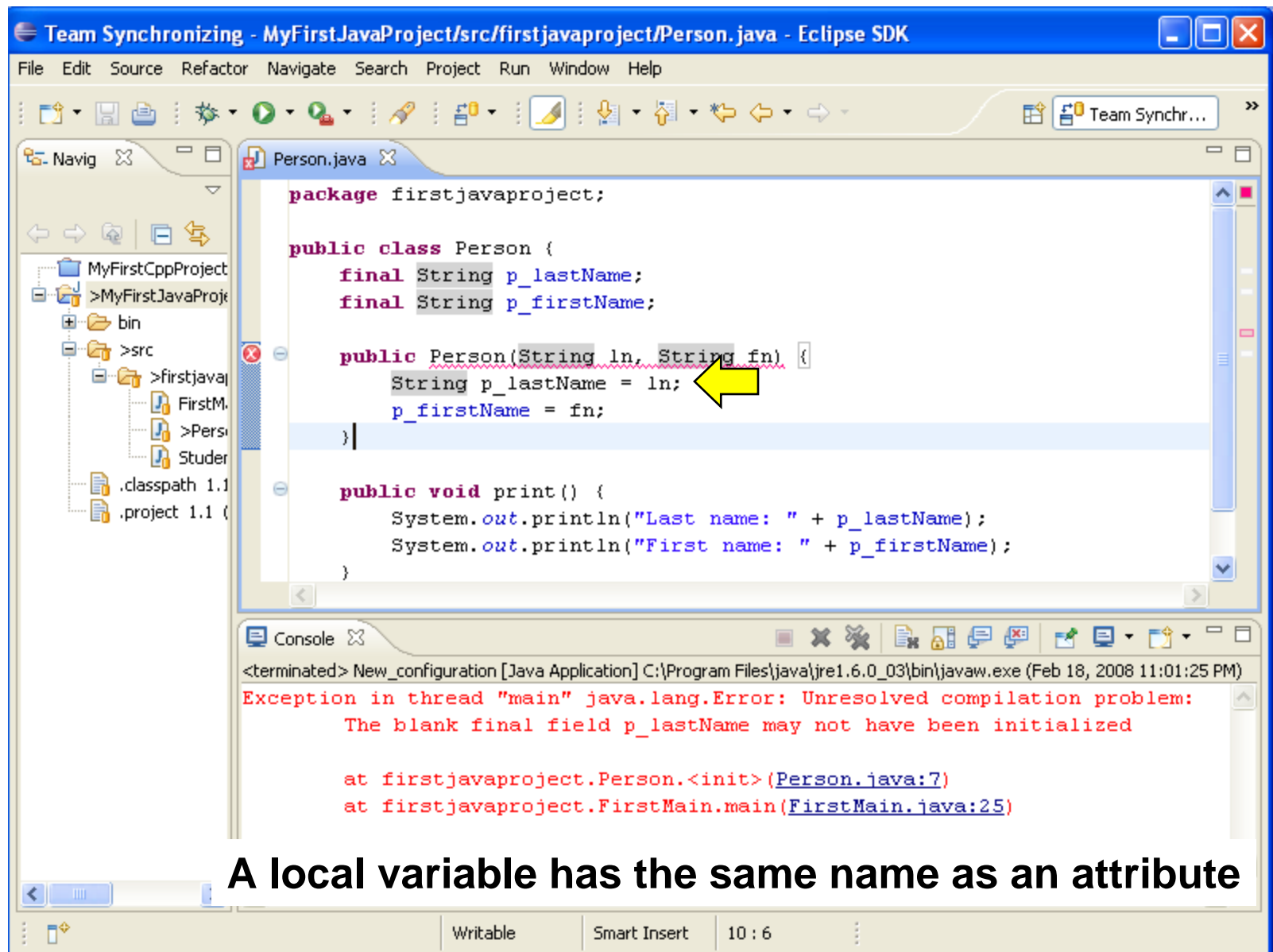
- A Driver object (dobj) sends an “accelerate” message to a Car object (cobj)
⇒ `cobj.accelerate();`
- A Teacher object (tobj) sends a message to a Student object (sobj) to submit a Homework object (hobj)
⇒ `sobj.submit(hobj);`
- A Person object (pobj) sends a Light Bulb object (bobj) message to turn on
⇒ `bobj.on();`

The object is the **recipient** of the message. Where is the sender? It is implicit by the location of the message.

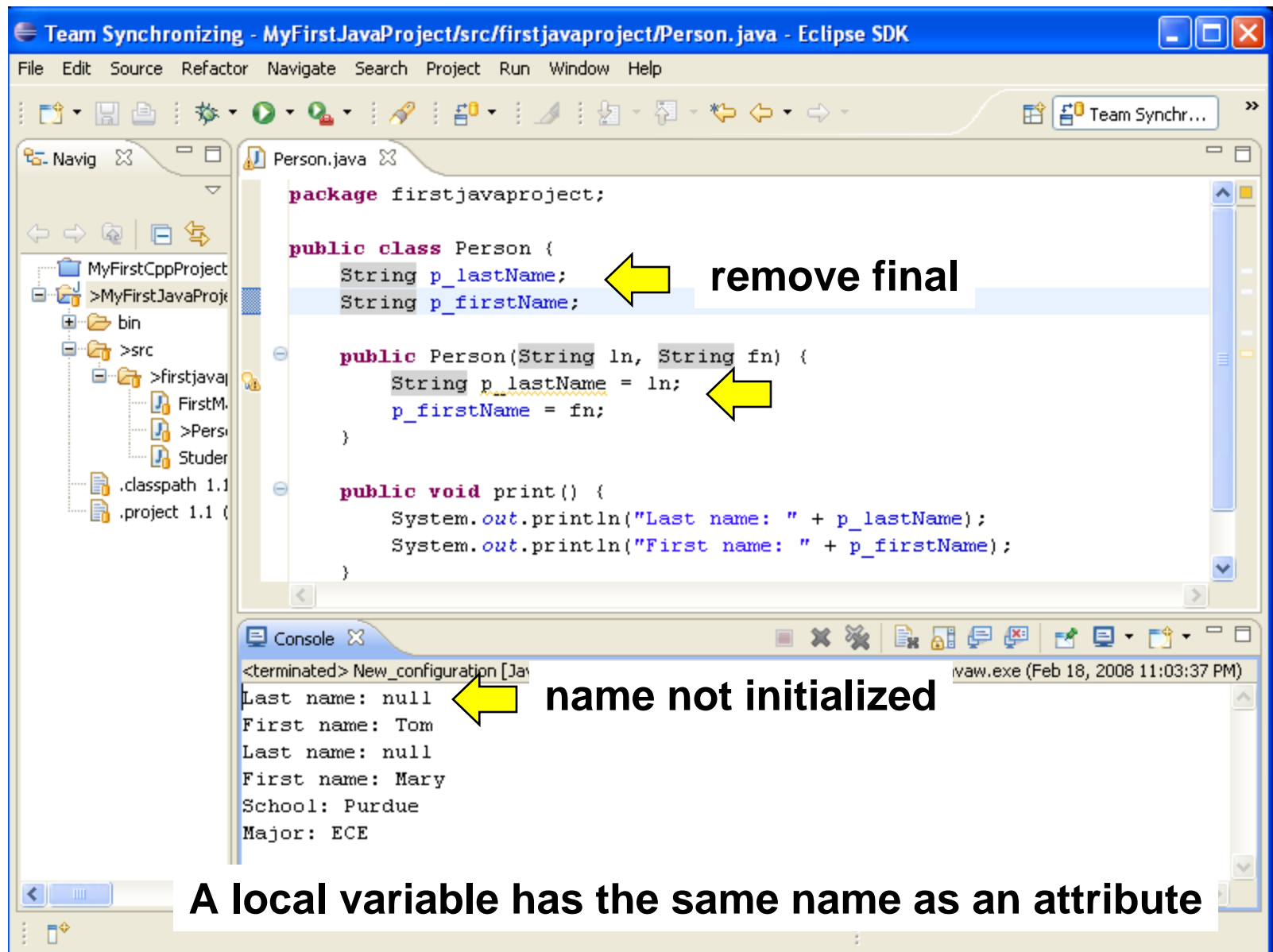


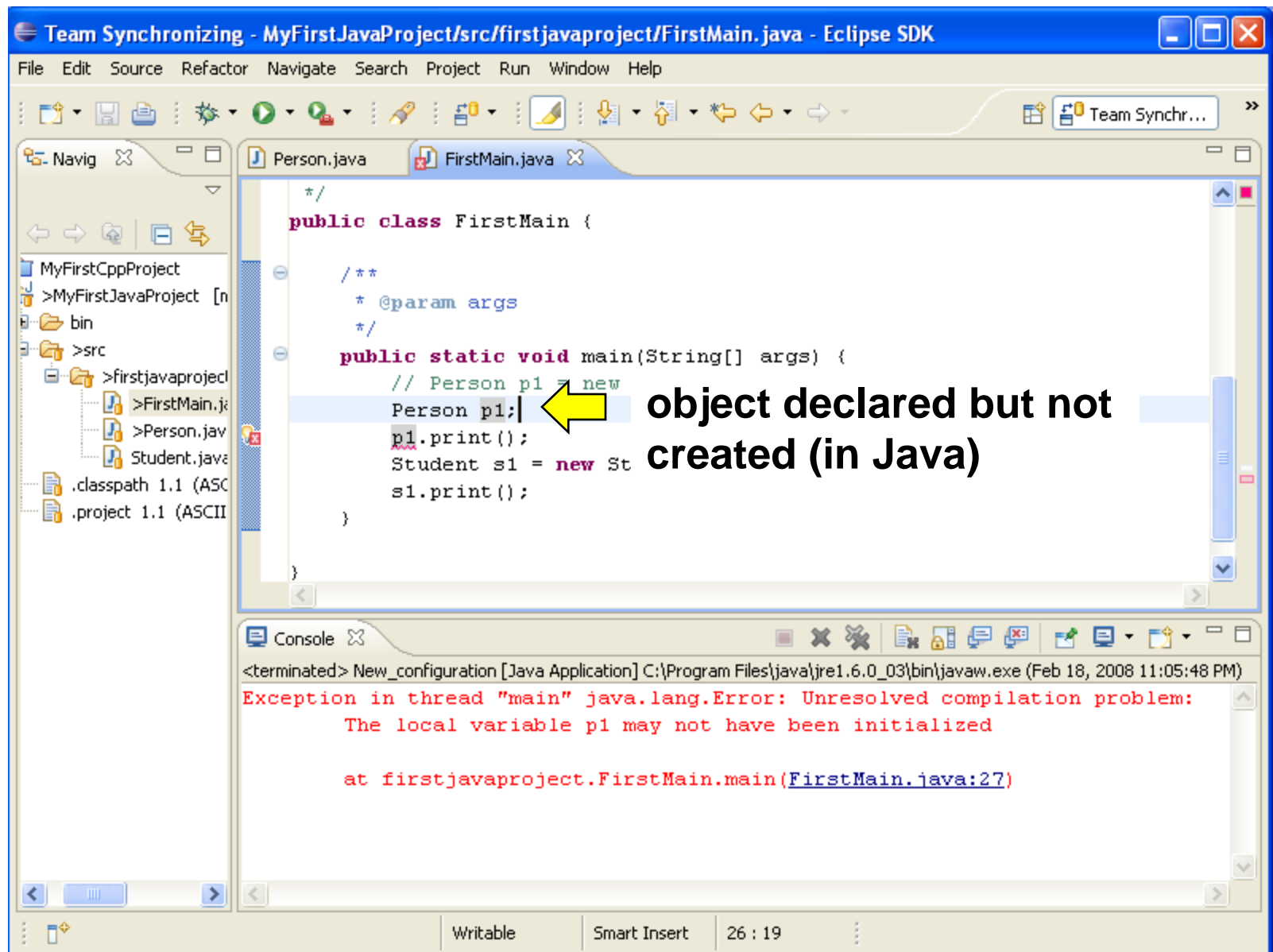


Common Mistakes



A local variable has the same name as an attribute





Self Test

ECE 462

Object-Oriented Programming
using C++ and Java

Class Examples

Yung-Hsiang Lu
yunглу@purdue.edu

JBButton (Java 2 Platform SE 5.0) - Mozilla Firefox

File Edit View History Bookmarks Yahoo! Tools Help

http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/JButton.html

Overview Package **Class** Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS

SUMMARY: NESTED | FIELD | CONSTR | METHOD

DETAIL: FIELD | CONSTR | METHOD

Java™ 2 Platform
Standard Ed. 5.0

javax.swing

Class JButton

[java.lang.Object](#)

- [java.awt.Component](#)
- [java.awt.Container](#)
- [javax.swing.JComponent](#)
- [javax.swing.AbstractButton](#)
- [javax.swing.JButton](#)

All Implemented Interfaces:
[ImageObserver](#), [ItemSelectable](#), [MenuItem](#)

Direct Known Subclasses:
[BasicArrowButton](#), [MetalComboBoxButton](#)

```
public class JButton
extends AbstractButton
implements Accessible
```

Done

← Class Hierarchy

↑ class name, usually noun

Constructor Summary ← **Overloaded Constructors**

overload = several functions with the same name but different parameters

constructor = function of the same name as the class

Method Summary

protected void	configurePropertiesFromAction (Action a) Factory method which sets the <code>AbstractButton</code> 's properties according to values from the <code>Action</code> instance.
AccessibleContext	getAccessibleContext ()

Done

JButton (Java 2 Platform SE 5.0) - Mozilla Firefox

File Edit View History Bookmarks Yahoo! Tools Help

http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/JButton.html

Google

Method Summary

protected void	configurePropertiesFromAction (Action a)	Factory method which sets the AbstractButton's properties according to values from the Action instance.
	getUIClassID ()	Returns a string that specifies the name of the L&F class that renders this component.
boolean	isDefaultButton ()	Gets the value of the defaultButton property, which if true means that this button is the current default button for its JRootPane.
	isDefaultCapable ()	Returns the value of the defaultCapable property.
protected String	 paramString ()	Returns a string representation of this JButton.
void	removeNotify ()	Overrides JComponent.removeNotify to check if this button is currently set as the default button on the RootPane, and if so, sets the RootPane's default button to null to ensure the RootPane

Done

encapsulation level

return type

methods usually verbs

JBButton (Java 2 Platform SE 5.0) - Mozilla Firefox

File Edit View History Bookmarks Yahoo! Tools Help

http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/JButton.html

Resets the UI property to a value from the current look and feel.

Methods inherited from class javax.swing.AbstractButton ← **inheritance**

[addActionListener](#), [addChangeListener](#), [addImpl](#), [addItemListener](#),
[checkHorizontalKey](#), [checkVerticalKey](#), [createActionListener](#),
[createActionPropertyChangeListener](#), [createChangeListener](#),
[createItemListener](#), [doClick](#), [doClick](#), [fireActionPerformed](#),
[fireItemStateChanged](#), [fireStateChanged](#), [getAction](#), [getActionCommand](#),
[getActionListeners](#), [getChangeListeners](#), [getDisabledIcon](#),
[getDisabledSelectedIcon](#), [getDisplayedMnemonicIndex](#), [getHorizontalAlignment](#),
[getHorizontalTextPosition](#), [getIcon](#), [getIconTextGap](#), [getItemListeners](#),
[getLabel](#), [getMargin](#), [getMnemonic](#), [getModel](#), [getMultiClickThreshold](#),
[getPressedIcon](#), [getRolloverIcon](#), [getRolloverSelectedIcon](#), [getSelectedIcon](#),
[getSelectedObjects](#), [getText](#), [getUI](#), [getVerticalAlignment](#),
[getVerticalTextPosition](#), [imageUpdate](#), [init](#), [isBorderPainted](#),
[isContentAreaFilled](#), [isFocusPainted](#), [isRolloverEnabled](#), [isSelected](#),
[paintBorder](#), [removeActionListener](#), [removeChangeListener](#),
[removeItemListener](#), [setAction](#), [setActionCommand](#), [setBorderPainted](#),
[setContentAreaFilled](#), [setDisabledIcon](#), [setDisabledSelectedIcon](#),
[setDisplayedMnemonicIndex](#), [setEnabled](#), [setFocusPainted](#),
[setHorizontalAlignment](#), [setHorizontalTextPosition](#), [setIcon](#), [setIconTextGap](#),
[setLabel](#), [setLayout](#), [setMargin](#), [setMnemonic](#), [setMnemonic](#), [setModel](#),
[setMultiClickThreshold](#), [setPressedIcon](#), [setRolloverEnabled](#),
[setRolloverIcon](#), [setRolloverSelectedIcon](#), [setSelected](#), [setSelectedIcon](#),

http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/AbstractButton.html#setBorderPainted(boolean)

Qt 4.3: QPushButton Class Reference - Mozilla Firefox

File Edit View History Bookmarks Yahoo! Tools Help

http://doc.trolltech.com/4.3/qpushbutton.html

Google

Container (Java 2 Platform SE 5.0) Qt 4.3: QPushButton Class Refere...

Home · All Classes · Main Classes · Grouped Classes · Modules · Functions

TROLLTECH

Trolltech Labs Blogs

QPushButton Class Reference

↑ class name, usually noun

The QPushButton widget provides a command button. [More...](#)

```
#include <QPushButton>
```

Inherits [QAbstractButton](#) ← inheritance

- List of all members, including inherited members
- Qt 3 support members

Properties

Done

Qt 4.3: QPushButton Class Reference - Mozilla Firefox

File Edit View History Bookmarks Yahoo! Tools Help

http://doc.trolltech.com/4.3/qpushbutton.html

Google

Container (Java 2 Platform SE 5.0) Qt 4.3: QPushButton Class Refere...

Inherits [QAbstractButton](#).

- List of all members, including inherited members
- Qt 3 support members

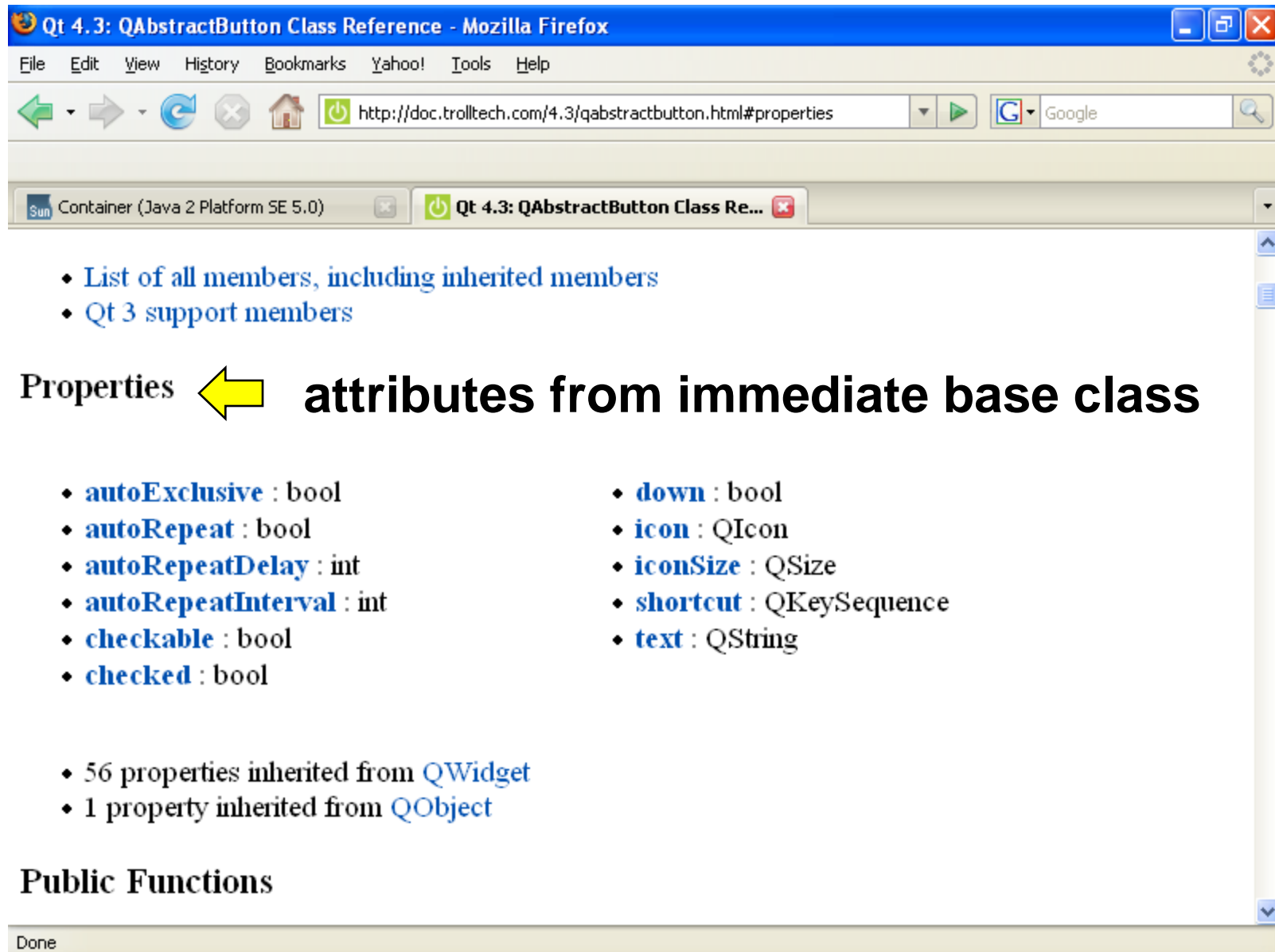
Properties ← attributes

- **autoDefault** : bool
- **default** : bool
- **flat** : bool
- 11 properties inherited from [QAbstractButton](#)
- 56 properties inherited from [QWidget](#)
- 1 property inherited from [QObject](#)

Public Functions

- **QPushButton** ([QWidget](#) * *parent* = 0)
- **QPushButton** (const [QString](#) & *text*, [QWidget](#) * *parent* = 0)
- **QPushButton** (const [QIcon](#) & *icon*, const [QString](#) & *text*, [QWidget](#) * *parent* = 0)

Done



Qt 4.3: QAbstractButton Class Reference - Mozilla Firefox

File Edit View History Bookmarks Yahoo! Tools Help

http://doc.trolltech.com/4.3/qabstractbutton.html#properties

Container (Java 2 Platform SE 5.0) Qt 4.3: QAbstractButton Class Re...

- List of all members, including inherited members
- Qt 3 support members

Properties ← attributes from immediate base class

- **autoExclusive** : bool
- **autoRepeat** : bool
- **autoRepeatDelay** : int
- **autoRepeatInterval** : int
- **checkable** : bool
- **checked** : bool
- **down** : bool
- **icon** : QIcon
- **iconSize** : QSize
- **shortcut** : QKeySequence
- **text** : QString

- 56 properties inherited from **QWidget**
- 1 property inherited from **QObject**

Public Functions

Done

Qt 4.3: QPushButton Class Reference - Mozilla Firefox

File Edit View History Bookmarks Yahoo! Tools Help

http://doc.trolltech.com/4.3/qpushbutton.html

Container (Java 2 Platform SE 5.0) Qt 4.3: QPushButton Class Refere...

Public Functions

- **QPushButton** (QWidget * *parent* = 0)
- **QPushButton** (const QString & *text*, QWidget * *parent* = 0)
- **QPushButton** (const QIcon & *icon*, const QString & *text*, QWidget * *parent* = 0)
- ~**QPushButton** ()
- bool **autoDefault** () const
- bool **isDefault** () const
- bool **isFlat** () const
- QMenu * **menu** () const
- void **setAutoDefault** (bool)
- void **setDefault** (bool)
- void **setFlat** (bool)
- void **setMenu** (QMenu * *menu*)
- 21 public functions inherited from **QAbstractButton**
- 12 public functions inherited from **QAbstractDevice**

methods usually verbs

Done

Qt 4.3: QWidget Class Reference - Mozilla Firefox

File Edit View History Bookmarks Yahoo! Tools Help

http://doc.trolltech.com/4.3/qwidget.html#properties

Container (Java 2 Platform SE 5.0) Qt 4.3: QWidget Class Reference

Public Functions

- **QWidget** (QWidget * *parent* = 0, Qt::WindowFlags *f* = 0)
- **~QWidget** ()
- bool **acceptDrops** () const
- QString **accessibleDescription** () const
- QString **accessibleName** () const
- QList<QAction *> **actions** () const
- void **activateWindow** ()
- void **addAction** (QAction * *action*)
- void **addActions** (QList<QAction *> *actions*)
- void **adjustSize** ()
- bool **autoFillBackground** () const
- QPalette::ColorRole **backgroundRole** () const
- Q:
- Q: **methods usually verbs**
- QWidget * **childAt** (const QPoint & *p*) const

Done

Qt 4.3: QWidget Class Reference - Mozilla Firefox

File Edit View History Bookmarks Yahoo! Tools Help

http://doc.trolltech.com/4.3/qwidget.html#properties

Container (Java 2 Platform SE 5.0) Qt 4.3: QWidget Class Reference

- QRect **frameGeometry** () const
- QSize **frameSize** () const
- const QRect & **geometry** () const
- void **getContentsMargins** (int * *left*, int * *top*, int * *right*, int * *bottom*) const
- virtual HDC **getDC** () const
- void **grabKeyboard** ()
- void **grabMouse** ()
- void **grabMouse** (const QCursor & *cursor*)
- int **grabShortcut** (const QKeySequence & *key*, Qt::ShortcutContext *context* = Qt::WindowShortcut)
- bool **hasEditFocus** () const
- bool **hasFocus** () const
- bool **hasMouseTracking** () const
- int **height** () const
- virtual int **heightForWidth** (int *w*) const
- QMethodQuery **methodQuery** (const QMethodSignature & *signature*) const
- void **insertAction** (QAction * *before*, QAction * *action*)

methods usually verbs

Done

Naming Convention

- class name: noun, capitalize first letter
- attribute name: noun, lower case
- method name: verb, lower case, followed by upper case

Self Test

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

Encapsulation and Polymorphism

Yung-Hsiang Lu
yunlu@purdue.edu

Implementation Independent

- If you ask a student for ID, do you **care** how the student finds the answer?
 - the student may remember the ID
 - the student may check the student's ID card
 - the student may call the department office and ask
 - the student may call a roommate
 - ...
- For you, these methods are the same. You obtain the ID number of the student.
- You do not need to know how the student **implements** the method to respond to your question.

Interface vs Implementation

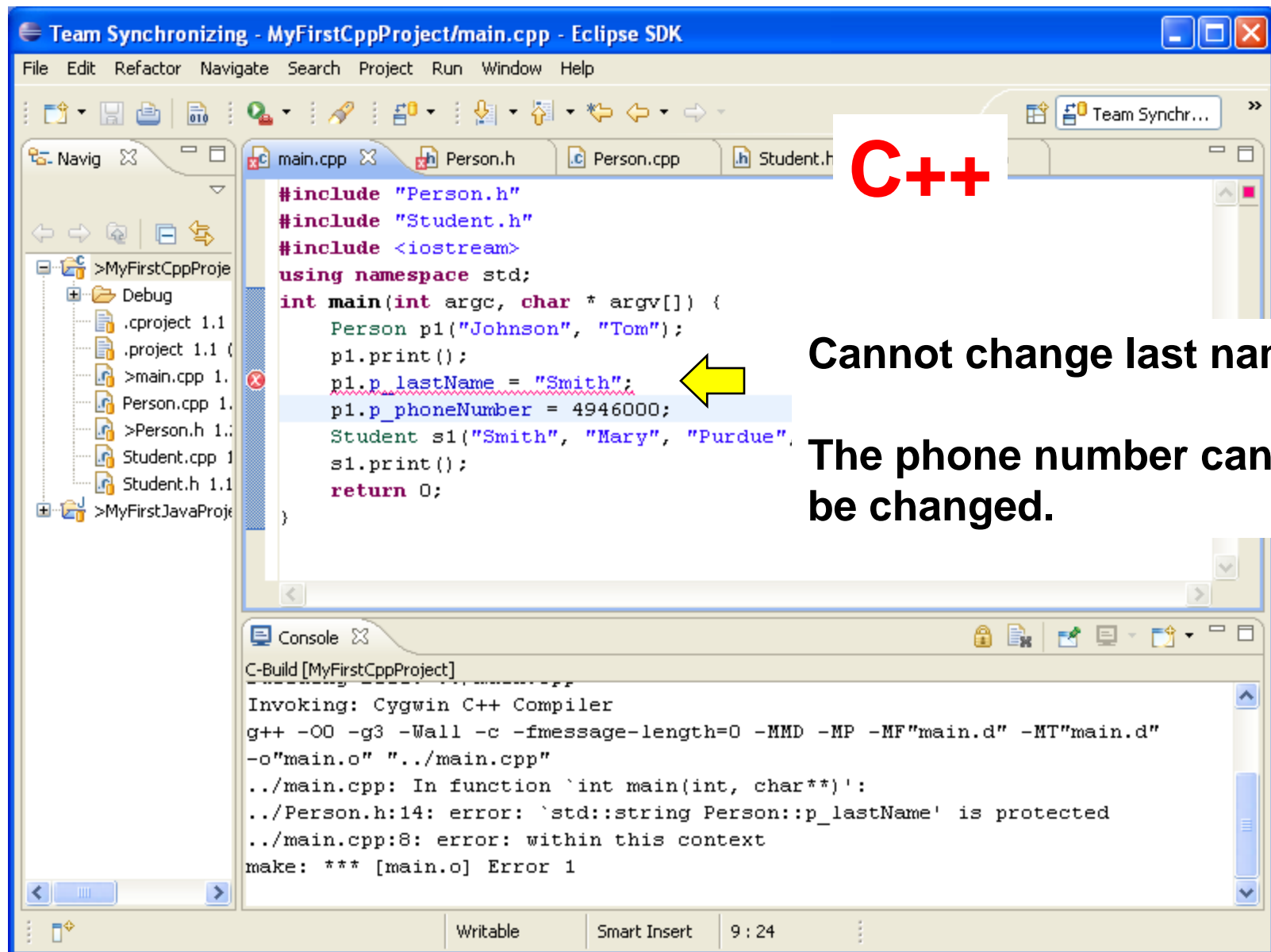
- Suppose you are creating a program to manage books in a library.
 - When the library has only hundreds of books, you can use a list to store each book (author, title, year ...)
 - As the number of books grows, you may need to use a more sophisticated indexing scheme or SQL
- The library provides the same **interface** to search books.
- Users should not feel any difference about the different **implementations**.
- Encapsulation is enforced by **compilers** (no run-time surprises).

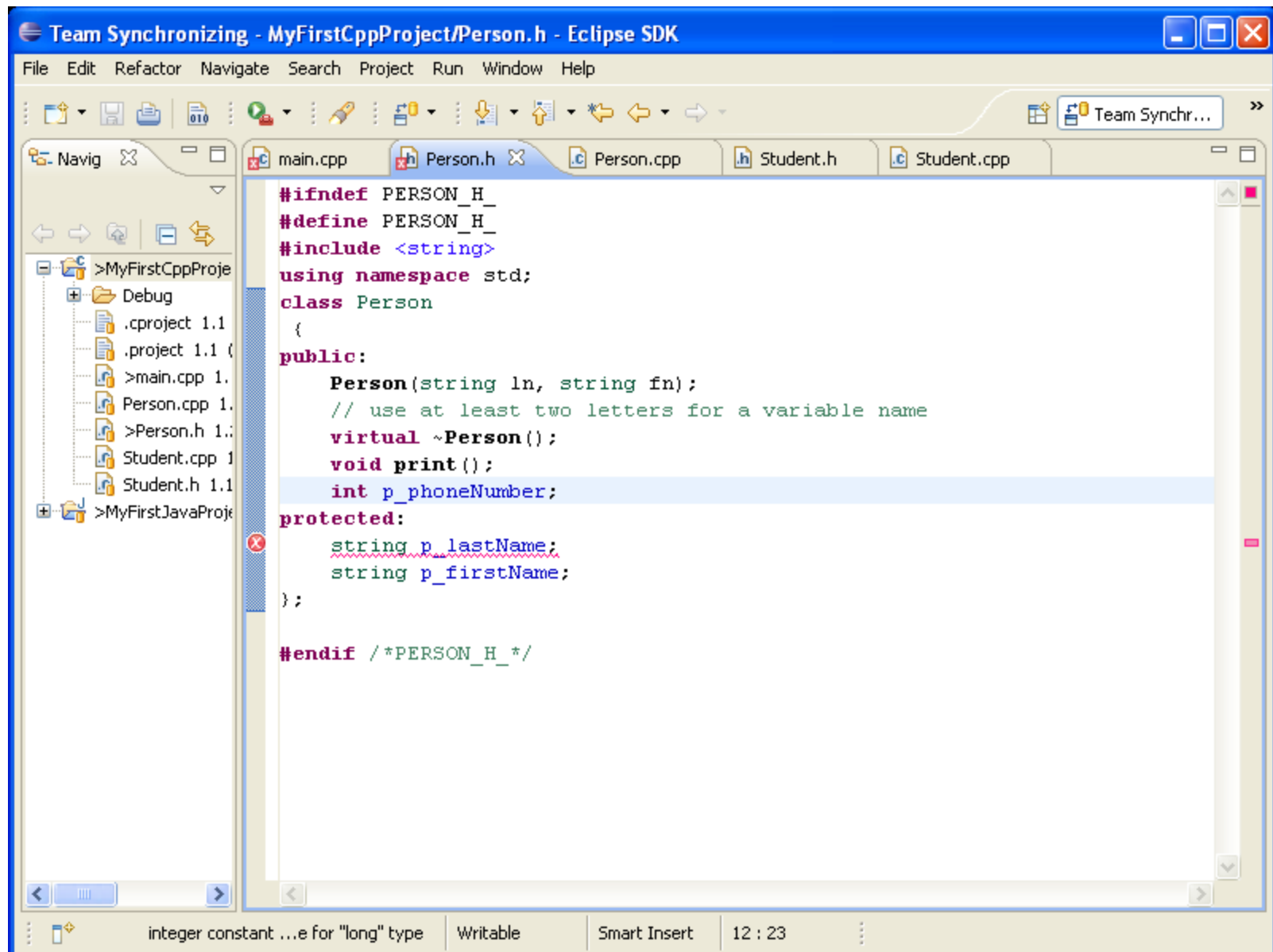
Interface

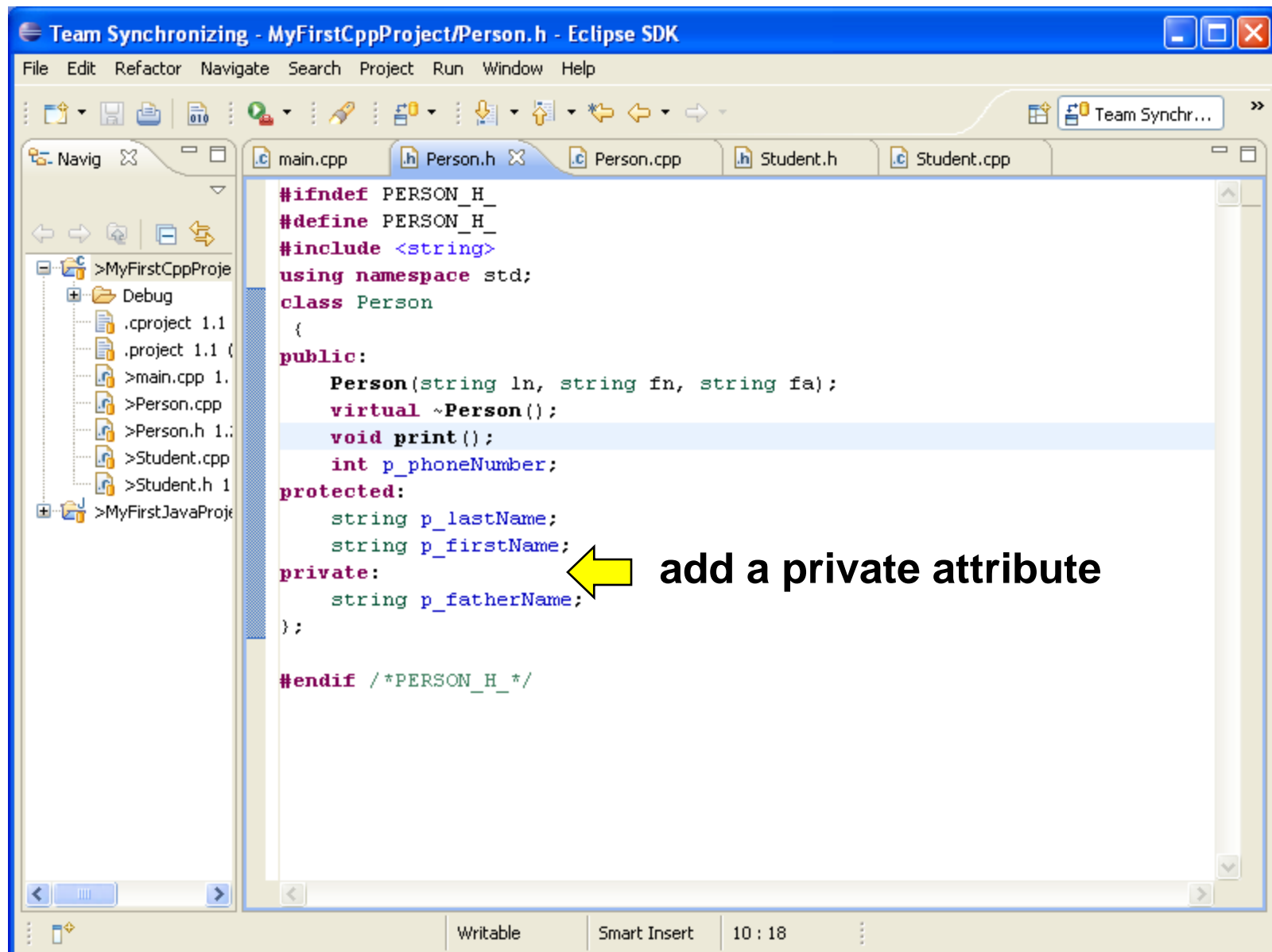
- A class' public attributes and methods form the interface for the objects of this class and all derived classes.
 - If `func` is a public method, the object must be able to respond to a call of `func`. **This is a "promise".**
 - Code reuse \Rightarrow A promise cannot be withdrawn.
 - Anything that is not in the interface (private attributes and methods) can be changed without affecting any code using the class.
- \Rightarrow The program is easier to maintain and reuse.
- Keep an **interface as small as possible**. Make only the essential functionalities visible. Hide all details.

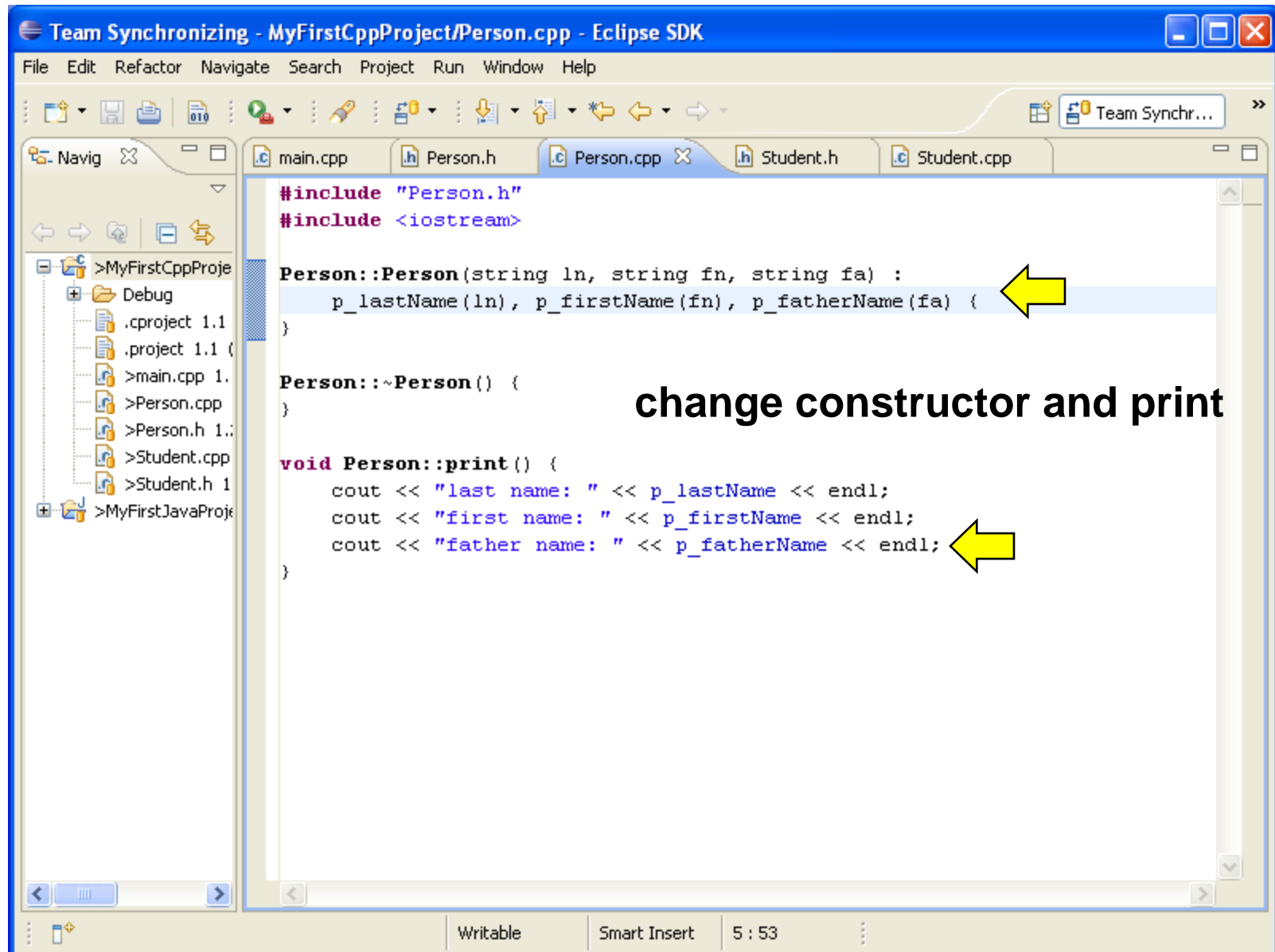
Why Encapsulation?

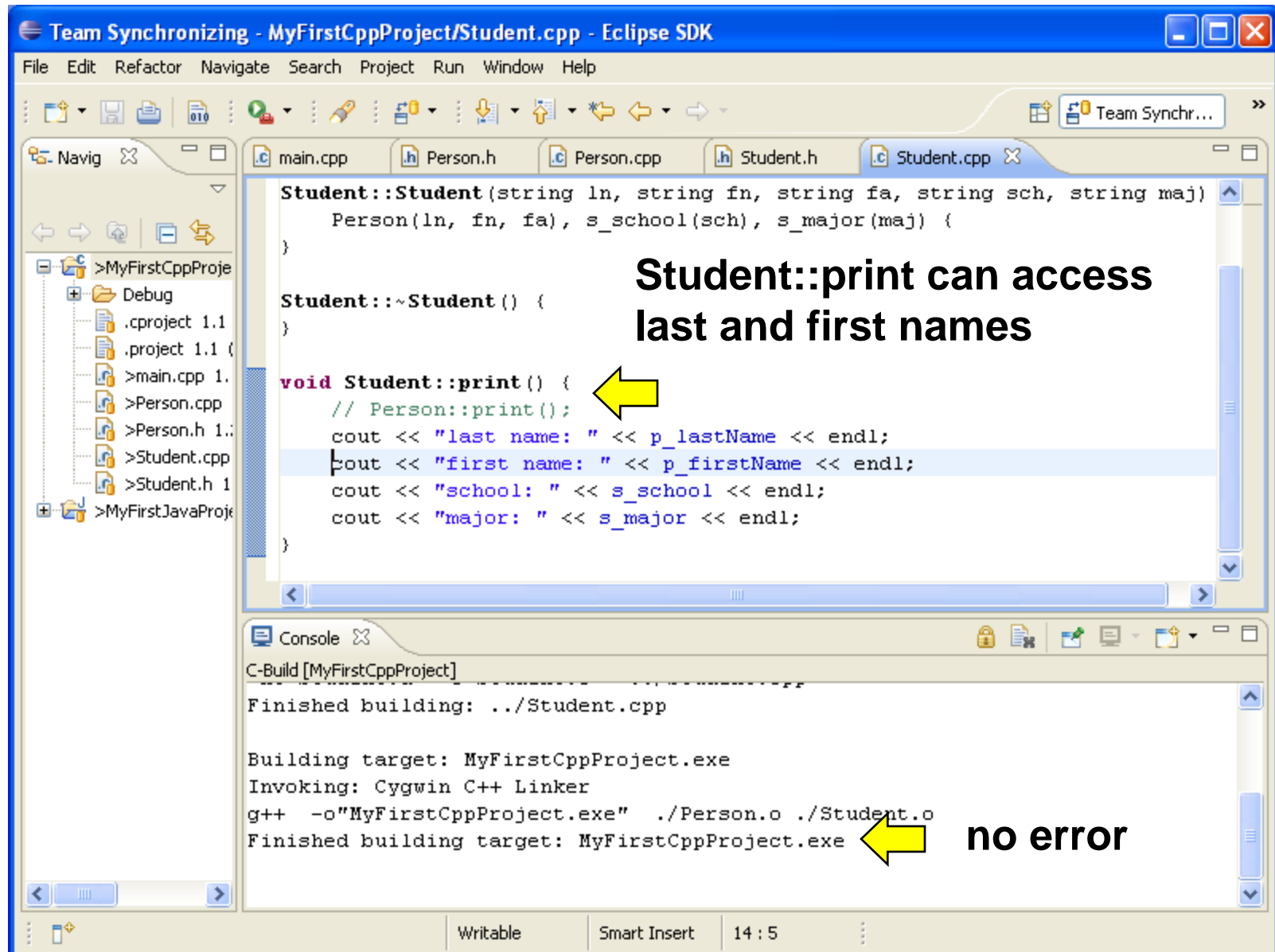
- prevent accidental modification of objects' attributes
- hide implementation details
- keep data consistency (some attributes must be changed simultaneously)
- reduce the amount of code to maintain (since the details are hidden, they will not affect the users of the code)
- success of OOP
 - large libraries for many functionalities
 - these libraries can be improved without breaking the users' code

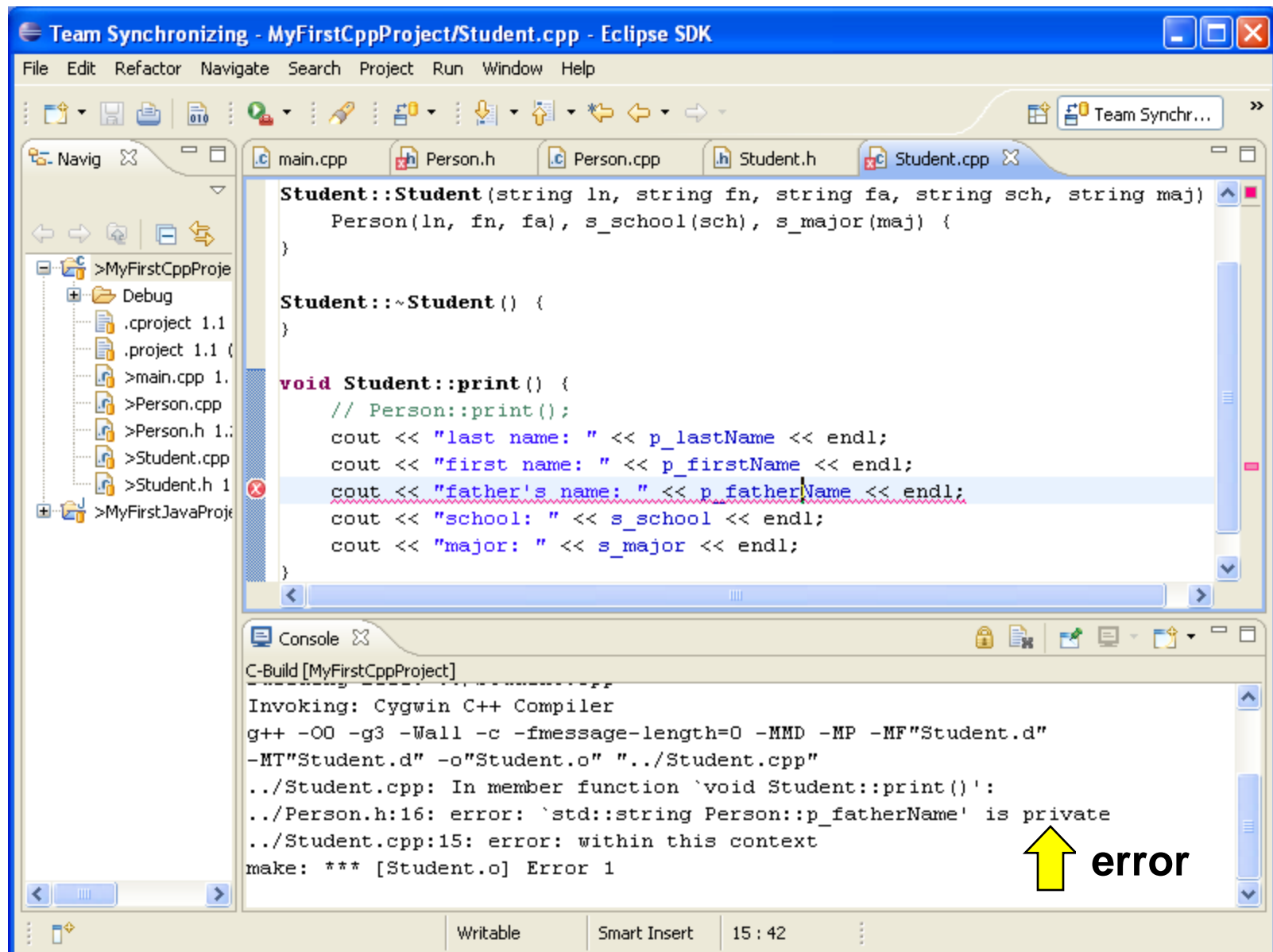




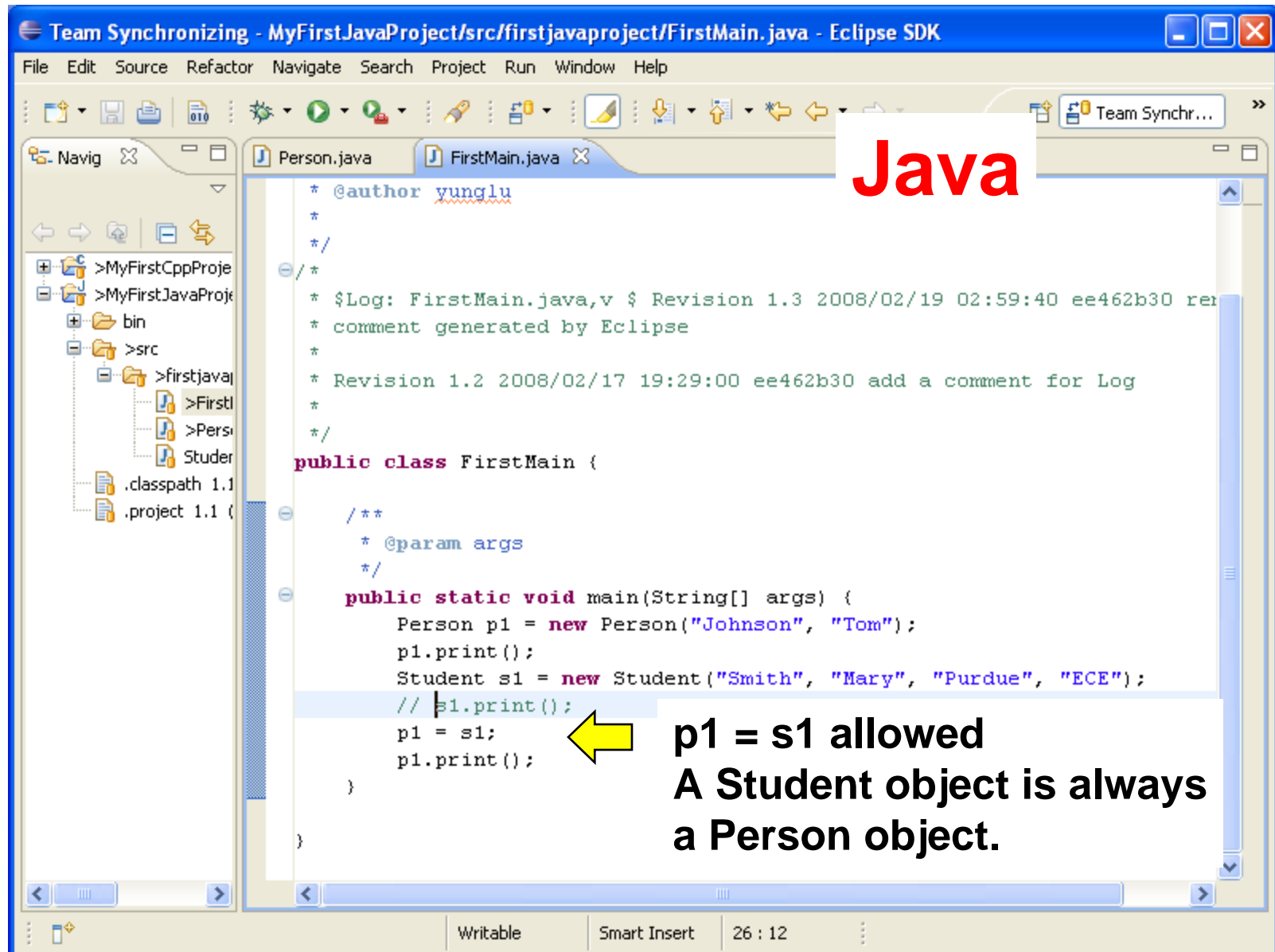


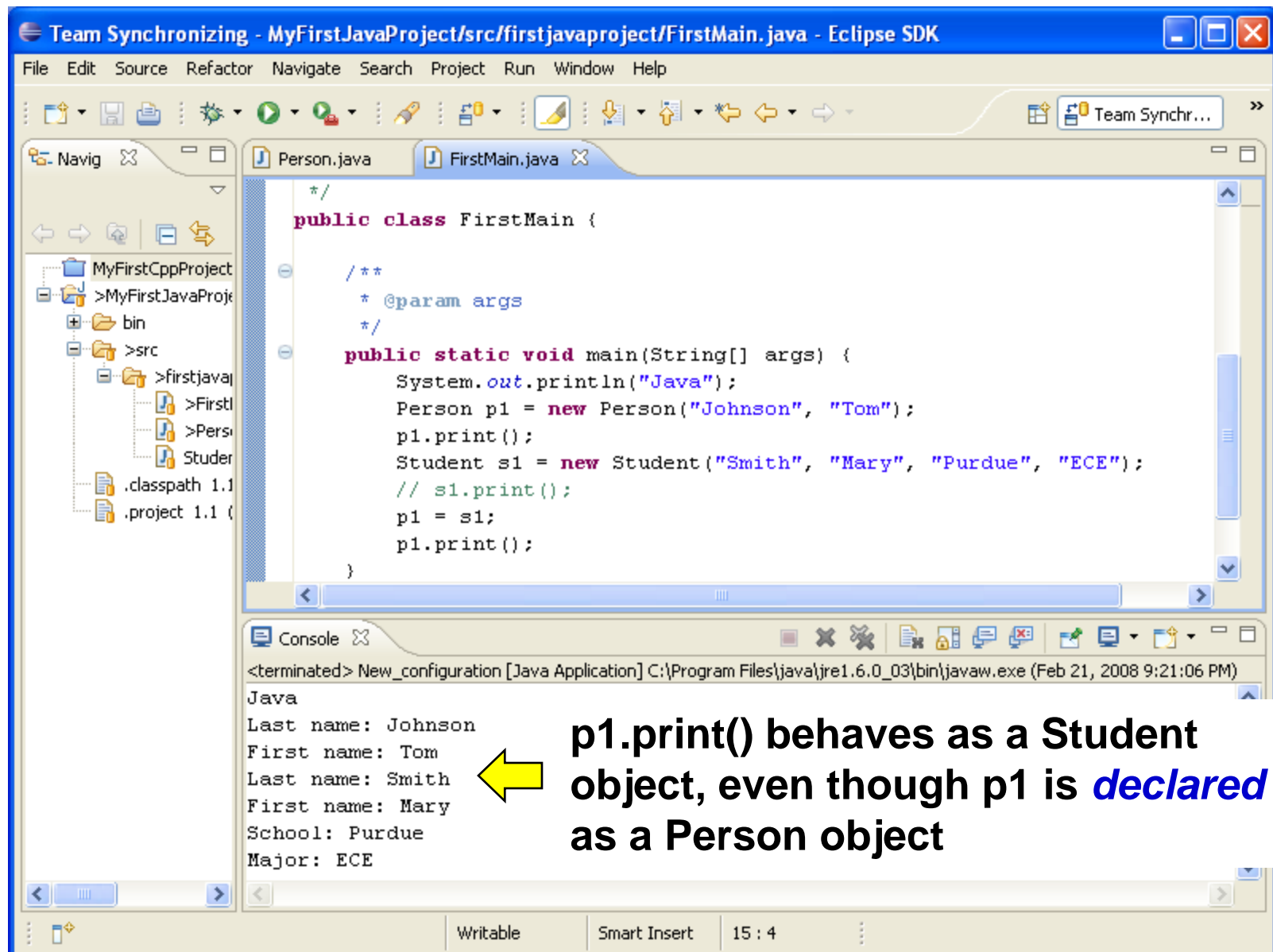


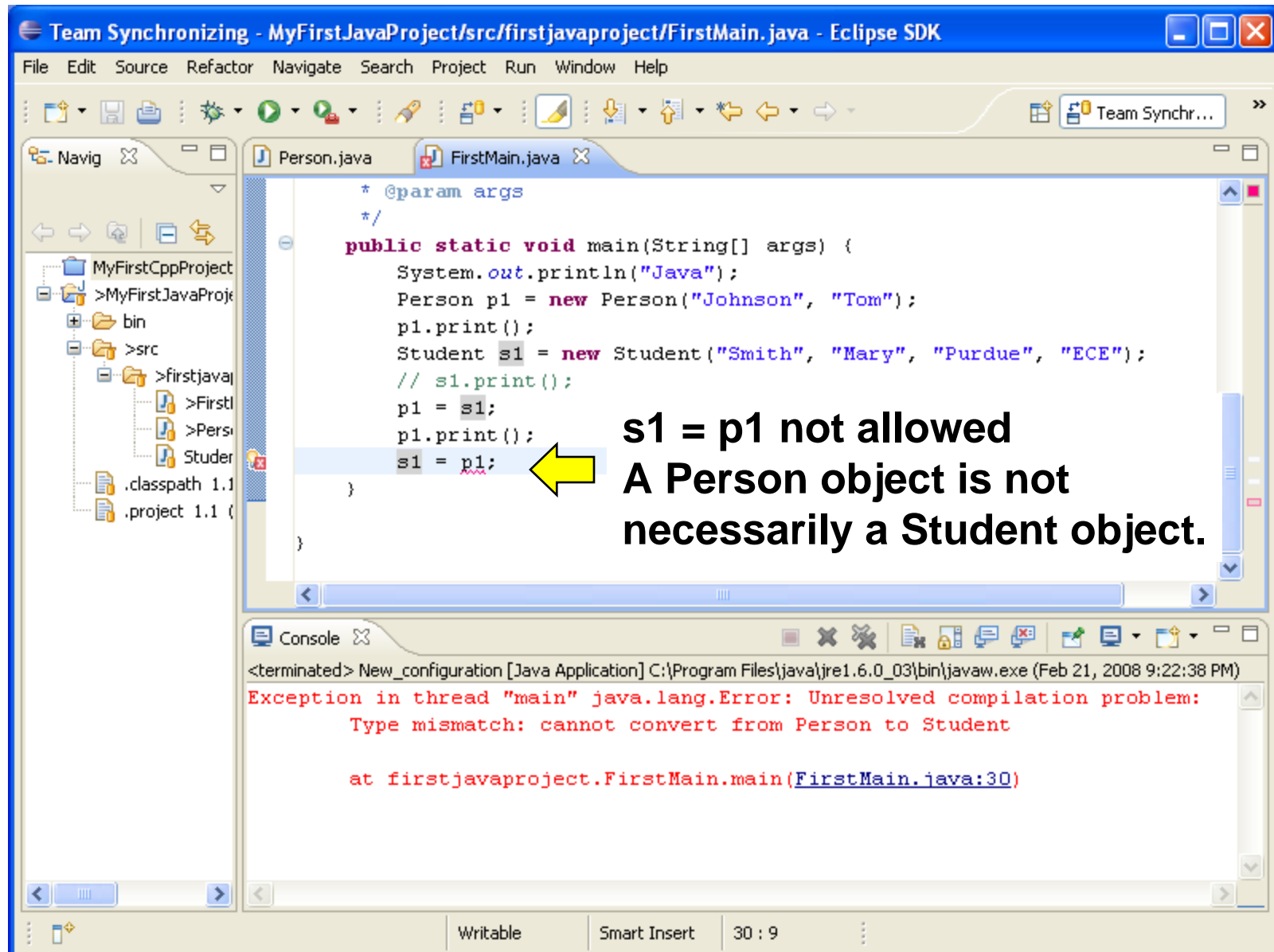




Polymorphism







Base	Derived	Object	Execute
Y	Y	Base	Base
Y	Y	Derived	Derived
Y	N	B	B
Y	N	D	B
N	Y	Base	Error
N	Y	Derived	D
N	N	B	Error
N	N	D	Error

Polymorphism

```
BaseClass obj1 = new BaseClass( ... );  
obj1.method();      // call base  
DerivedClass obj2 = new DerivedClass( ... );  
obj2.method();      // call derived (if available)  
obj1 = obj2;        // no problem  
obj1.method();      // call derived (if available)  
obj2 = obj1;        // error
```

Virtual Function in C++

- In Java, all functions are "virtual" and polymorphism is always supported.
- In C++, polymorphism is **not** enabled by default.
- A function is polymorphic only if it is declared **virtual** at the base class and the object is created by **new**.

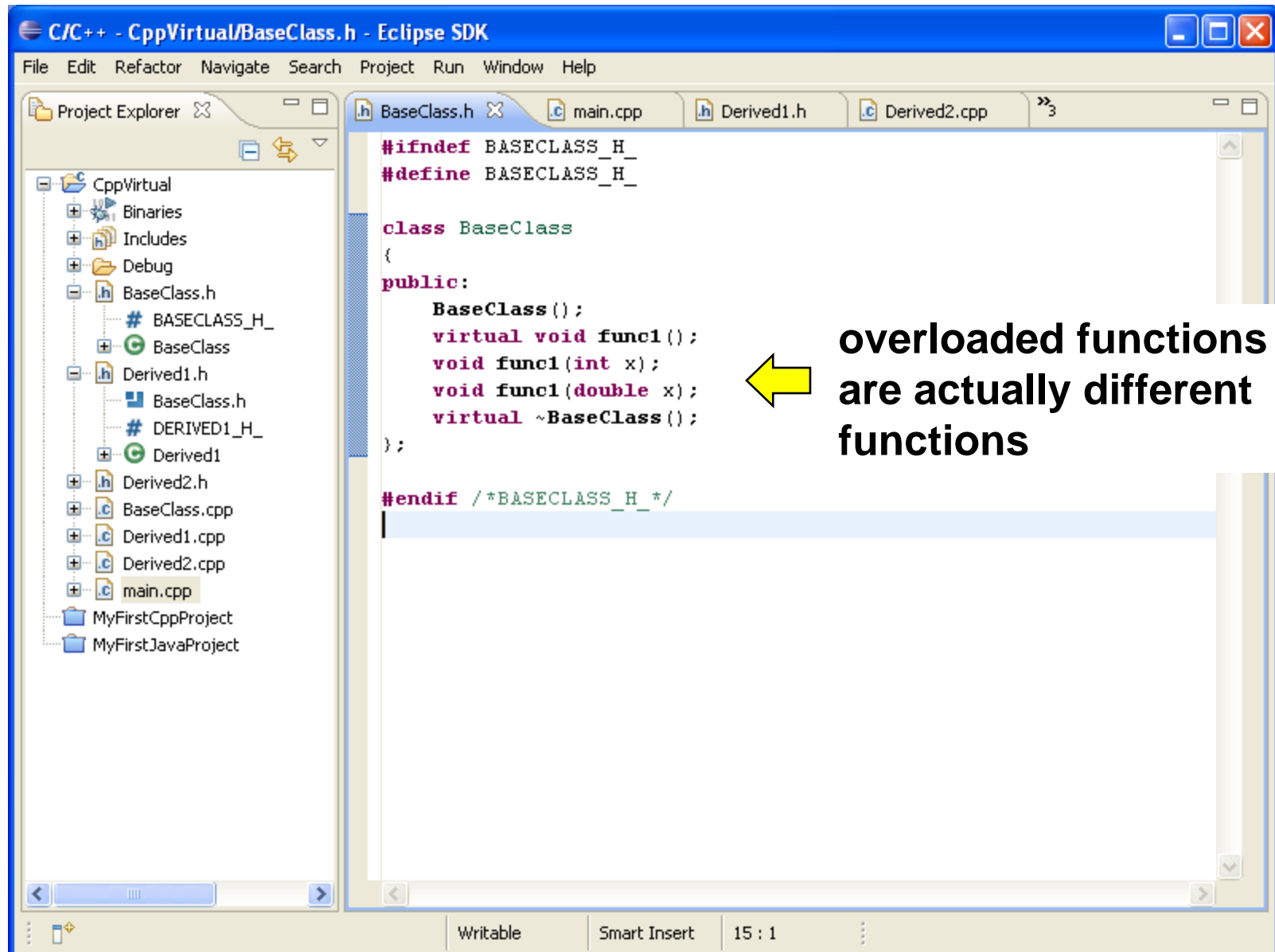
ClassName * obj = new ClassName(parameters);

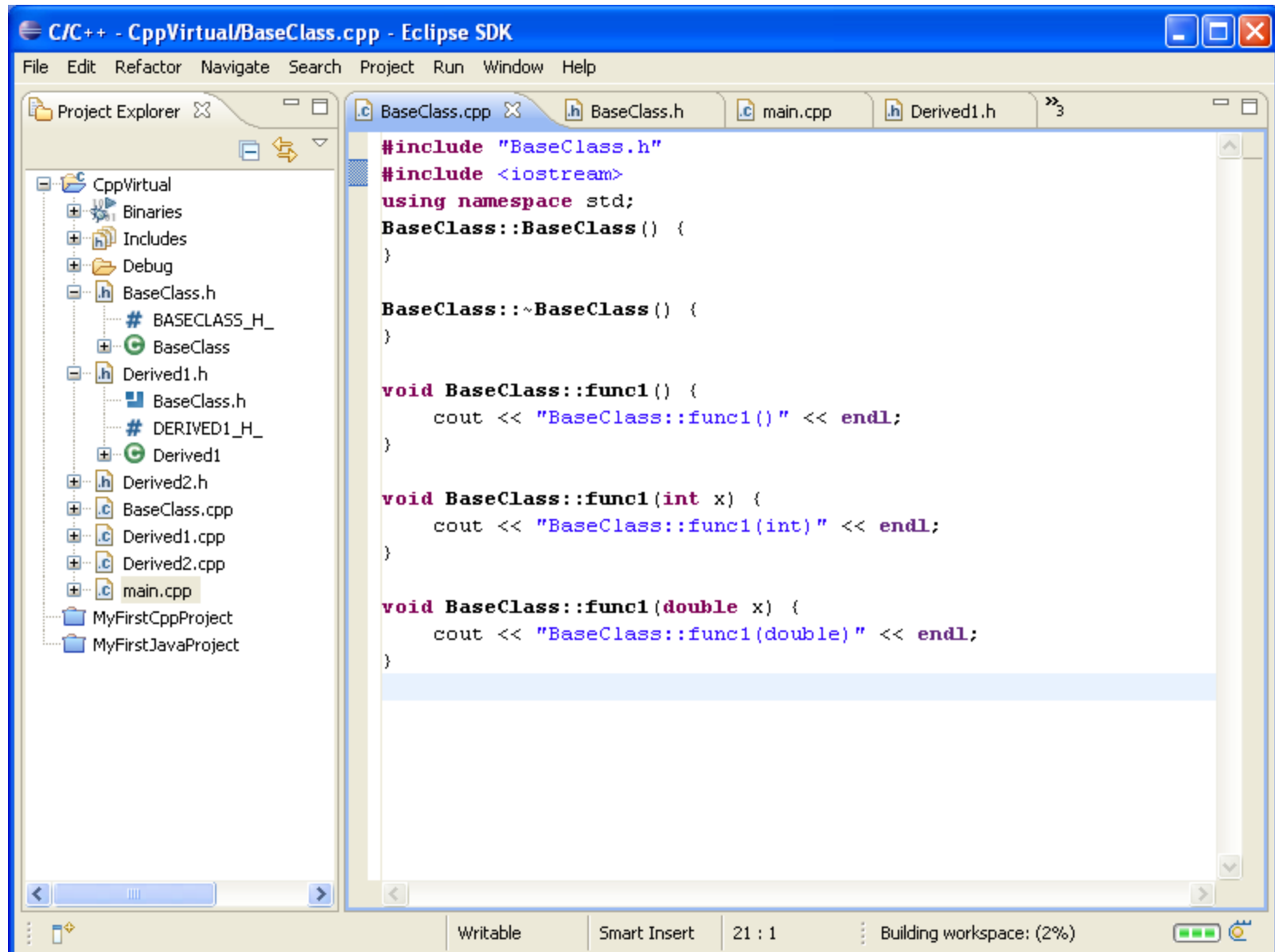
ClassName * obj = new ClassName; // no parameter

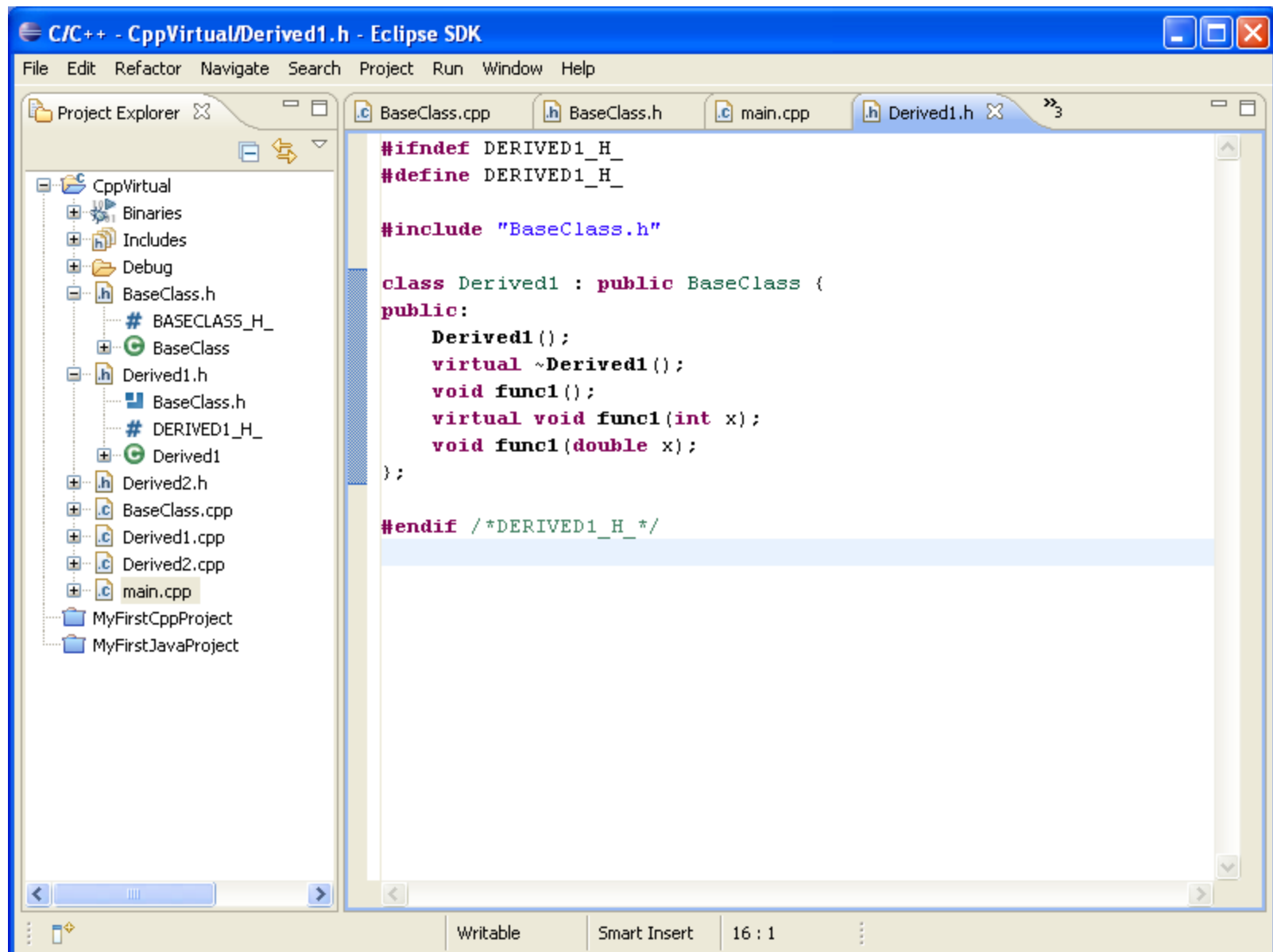
- Once a function is virtual, it is virtual for **all derived** classes.
- If a derived class can use the same method (implementation), do not override the method.

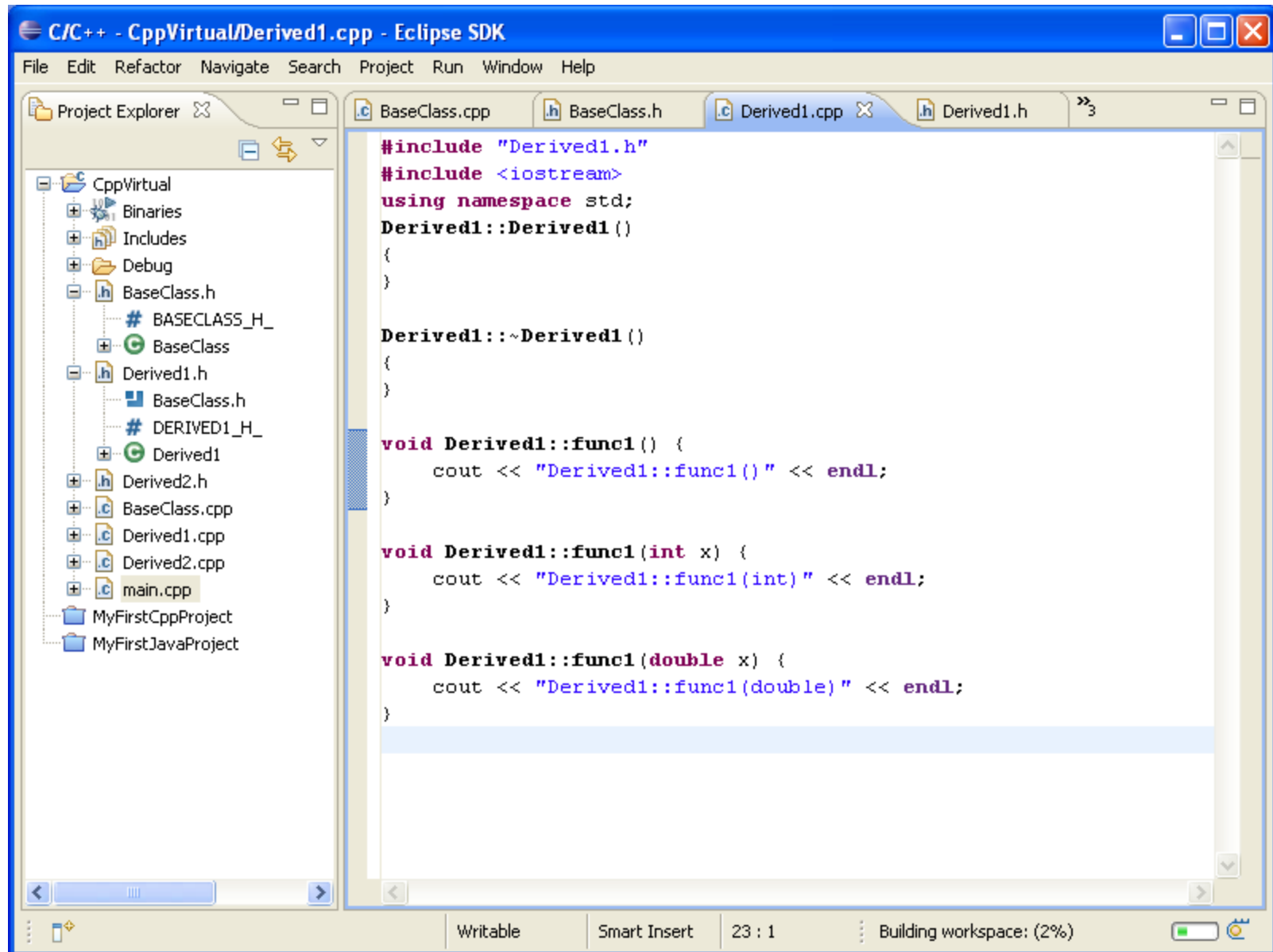
Virtual in C++

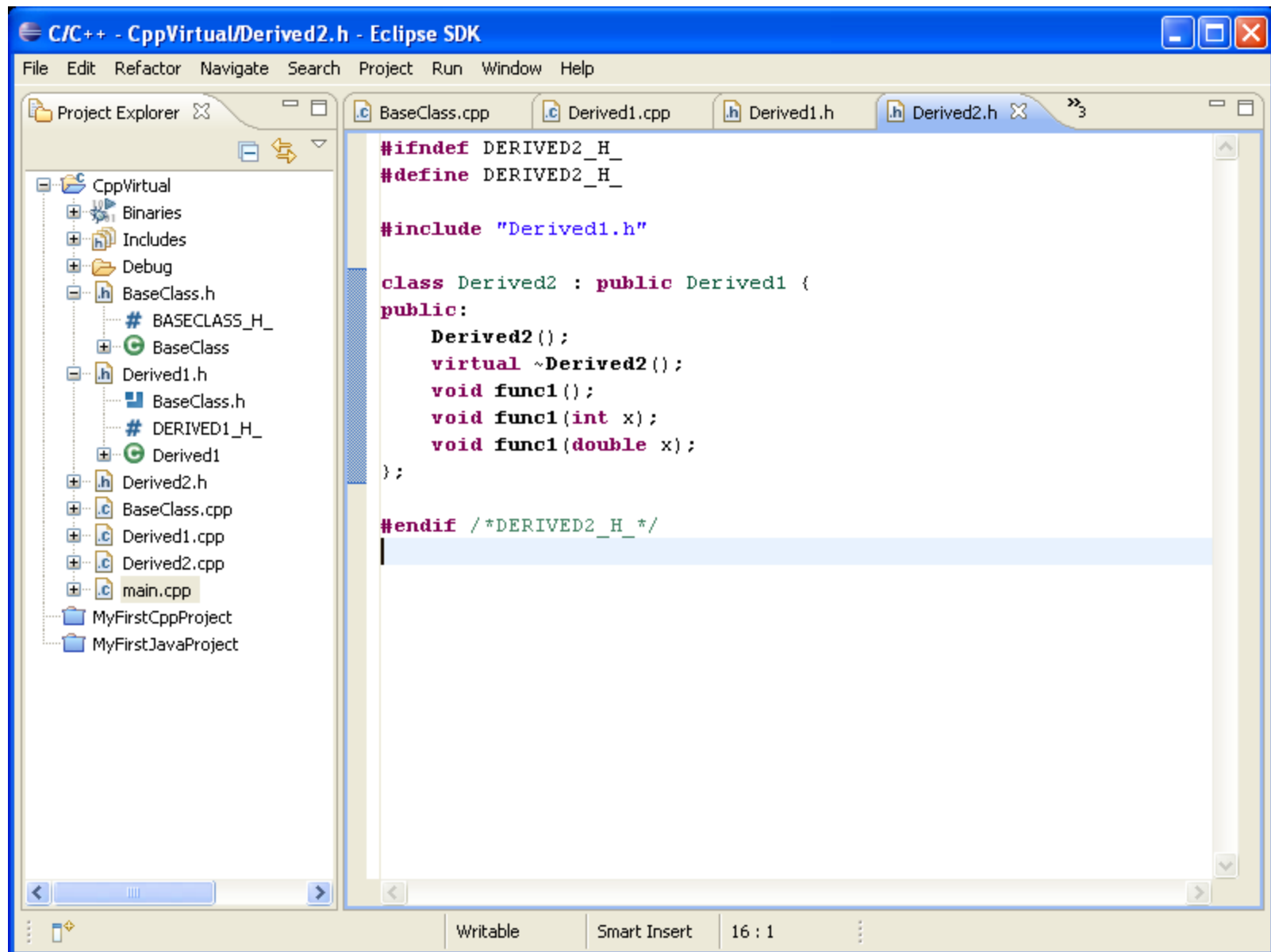
- All virtual methods must have **the same prototype** (i.e. return type and argument types).
- virtual \Rightarrow derived class may (not have to) override
- not virtual \Rightarrow should **not** override, compiler will allow, but don't ask for trouble
- why virtual in C++? slightly better performance for non-virtual ... but ... cause too much confusion
- In general, functions in C++ **should be virtual** unless you have strong reasons (and know what you are doing)
- Constructors are naturally virtual. **Destructors** should **always** be virtual (more about this later).

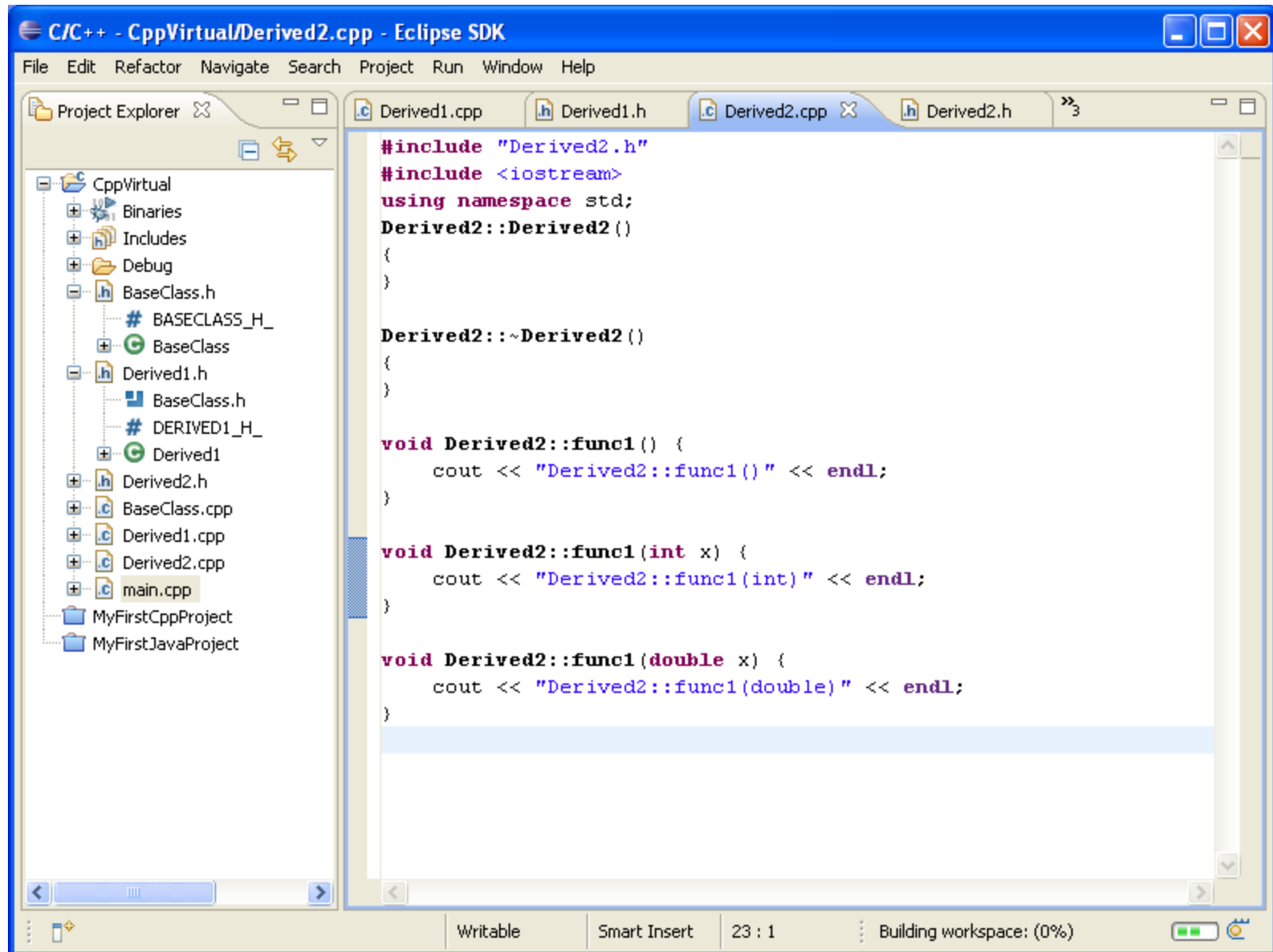


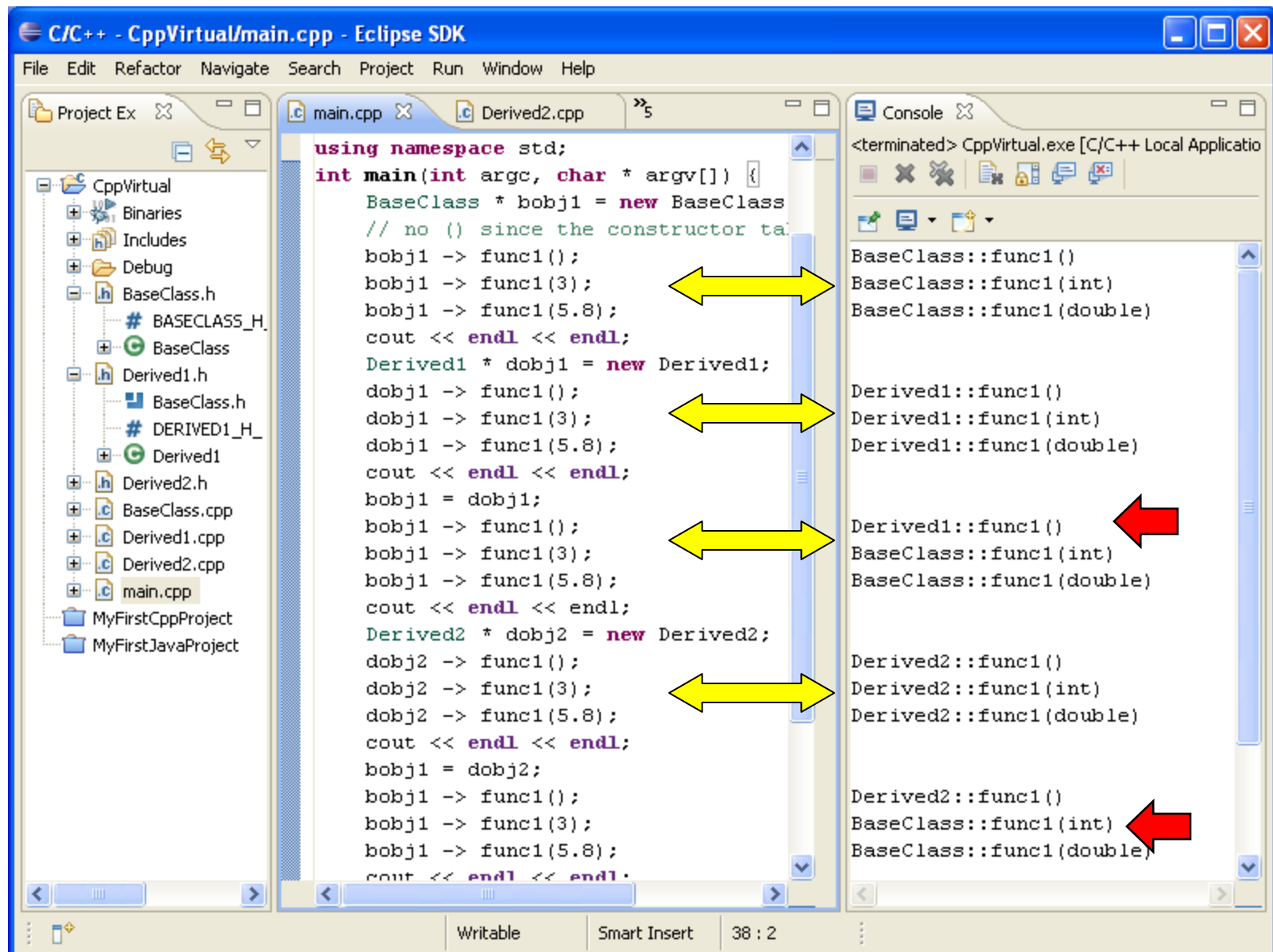


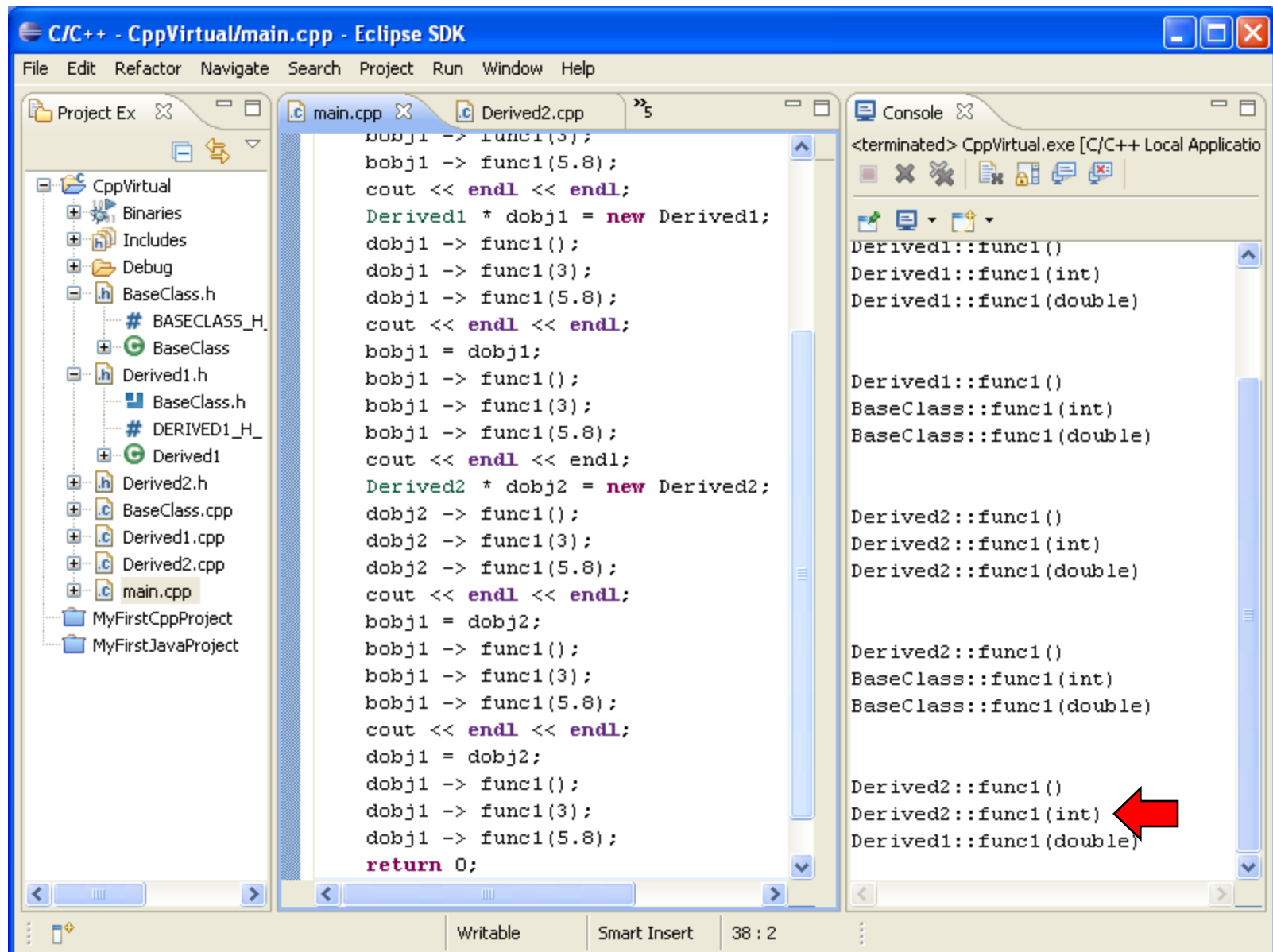




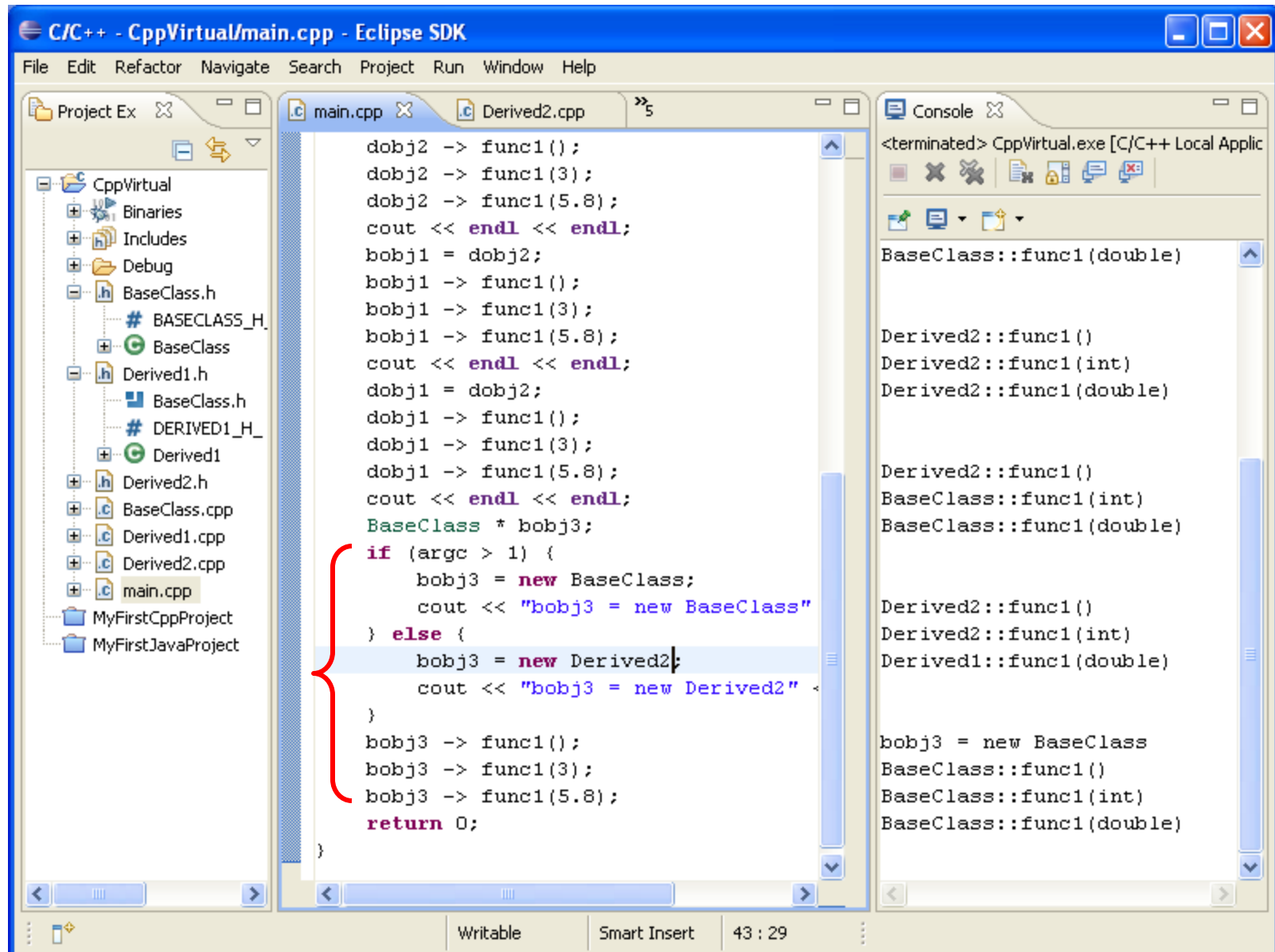


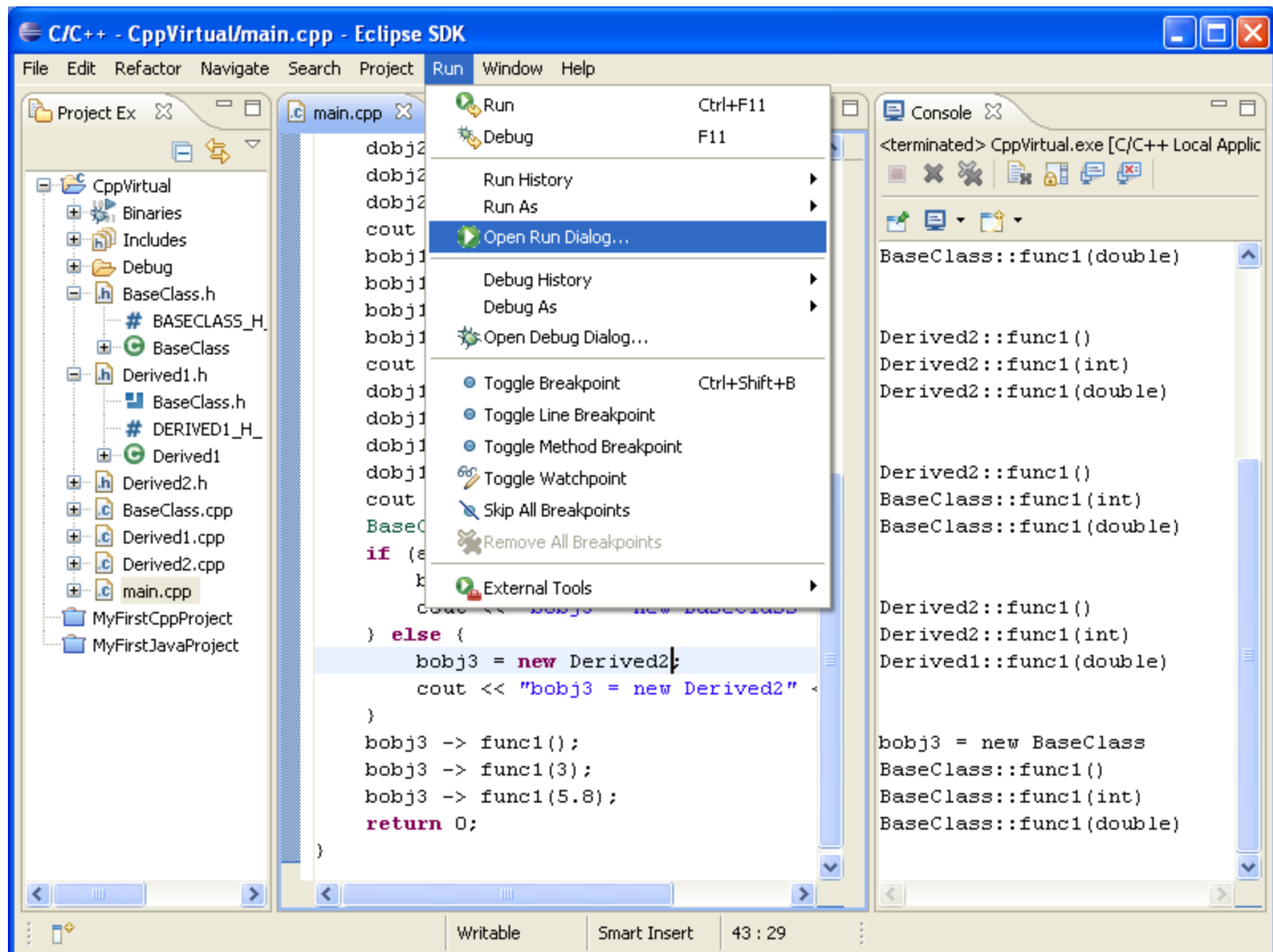


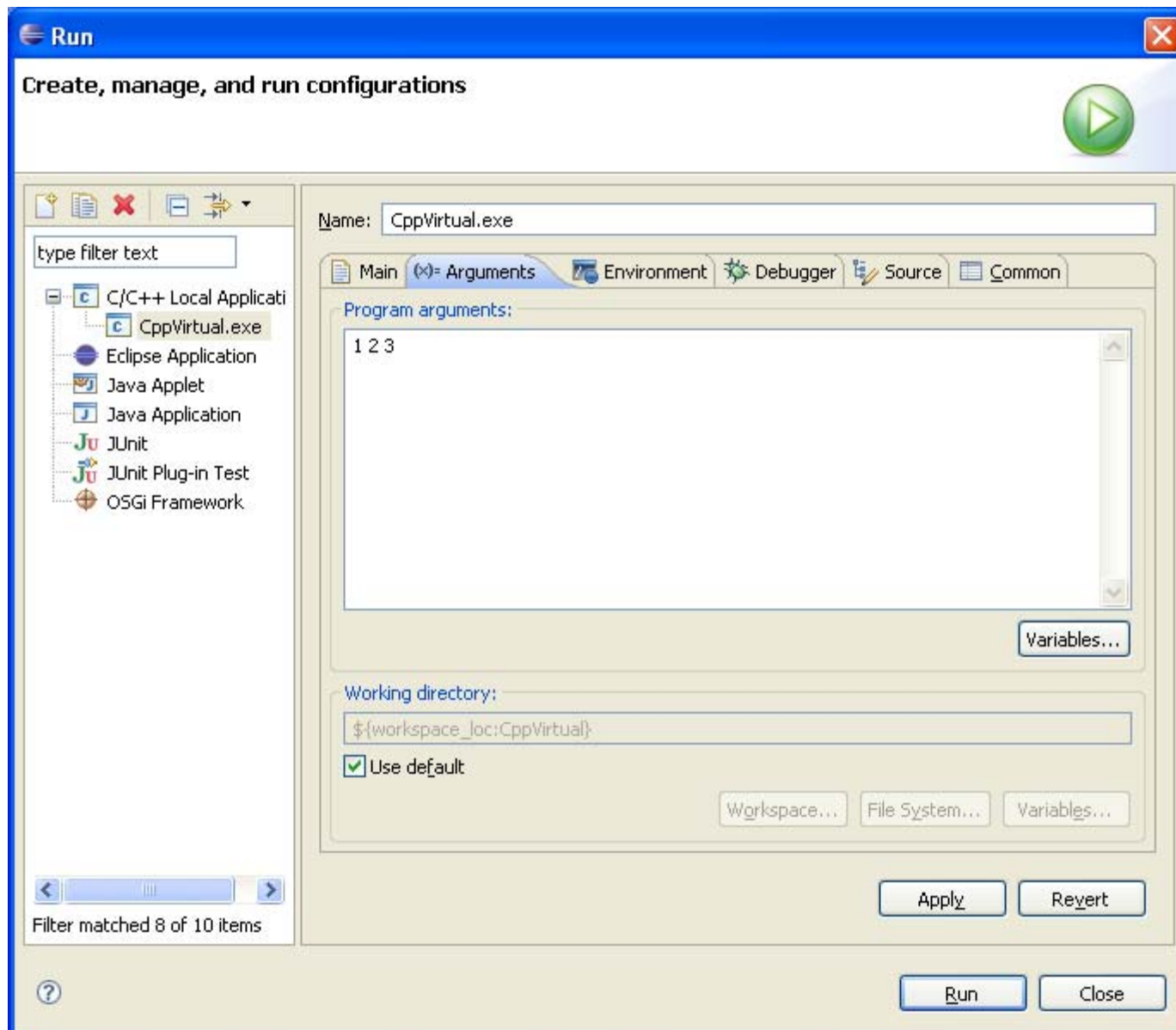


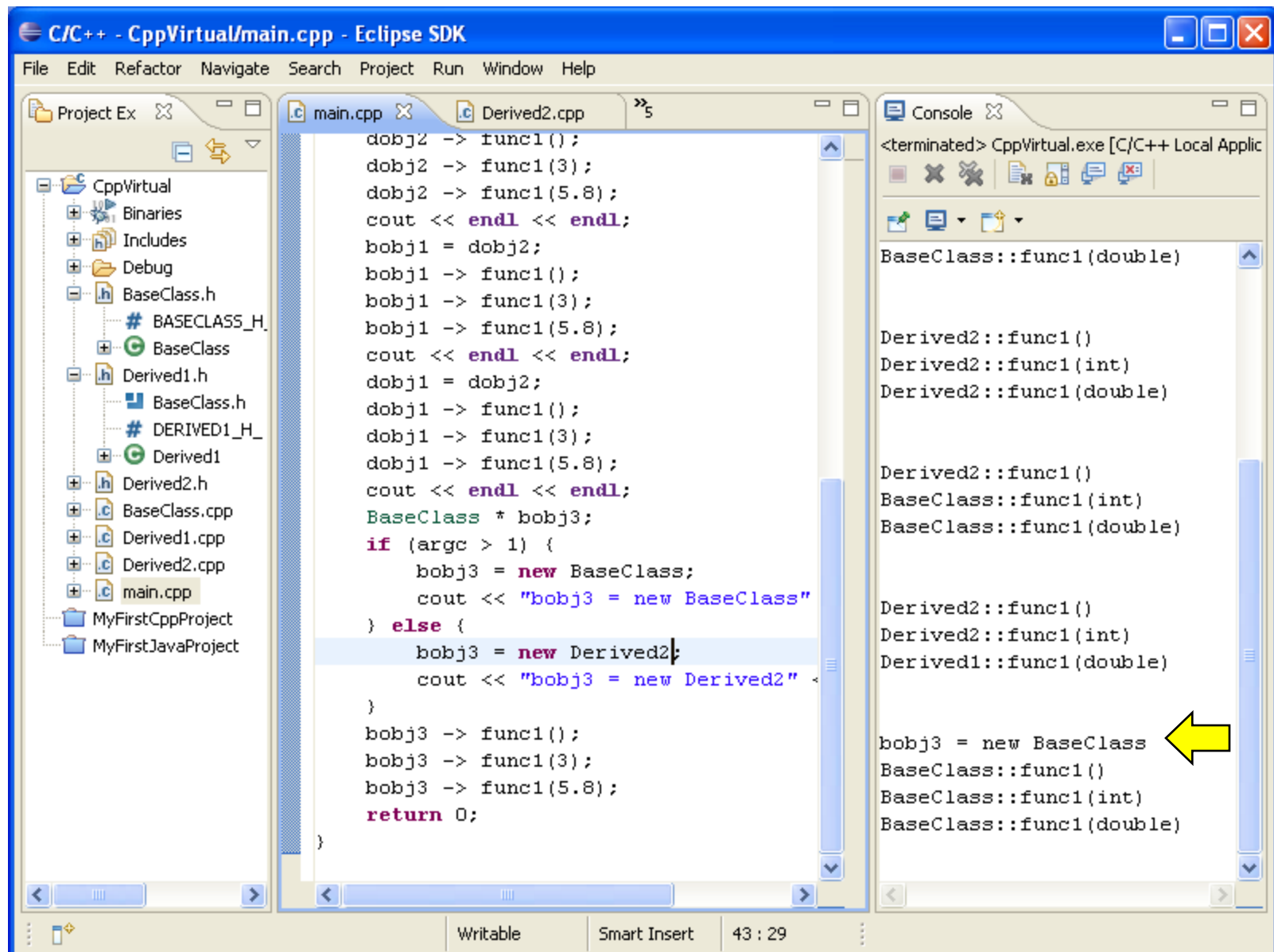


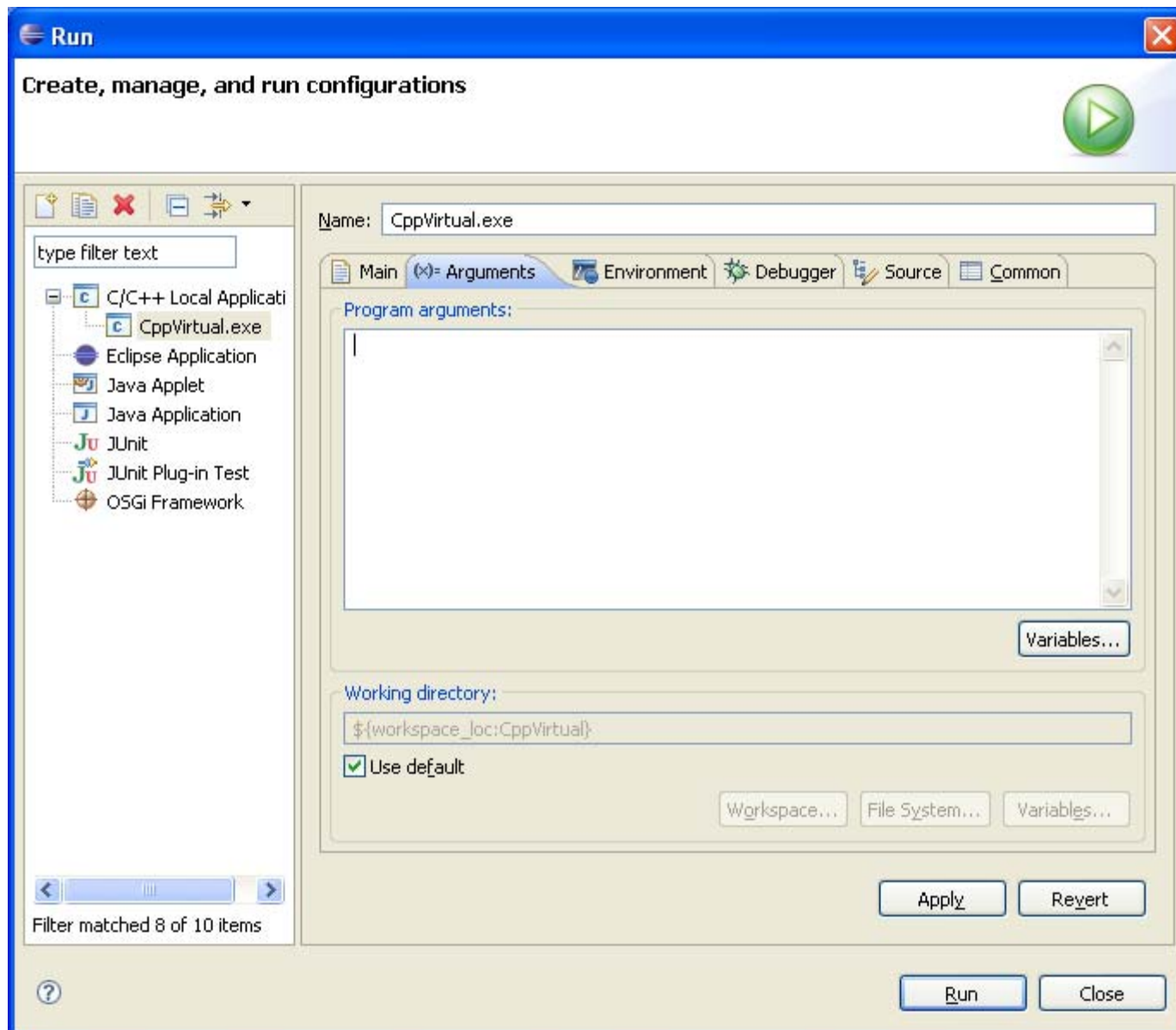
**Polymorphism is determined at
"run-time"**

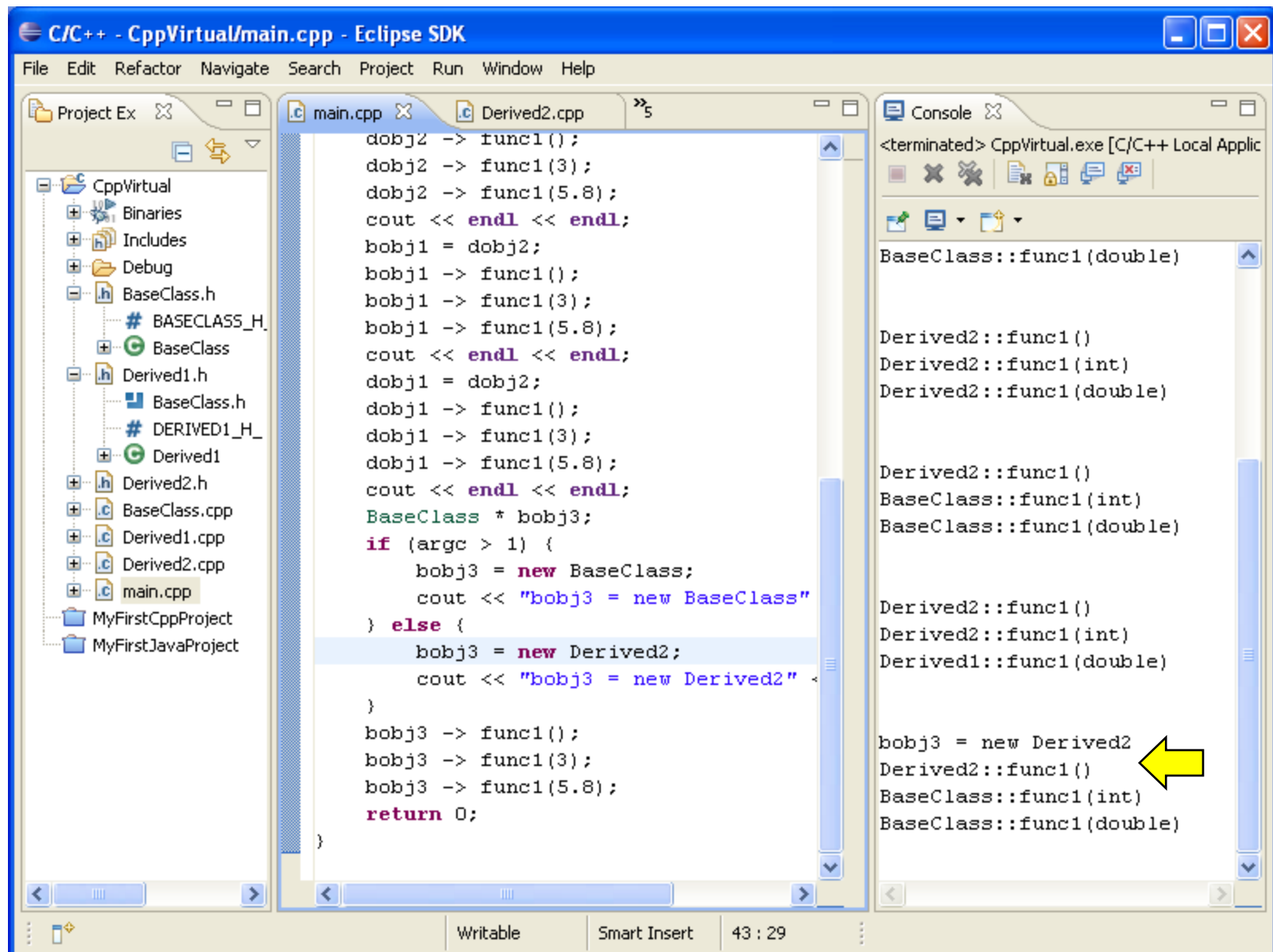


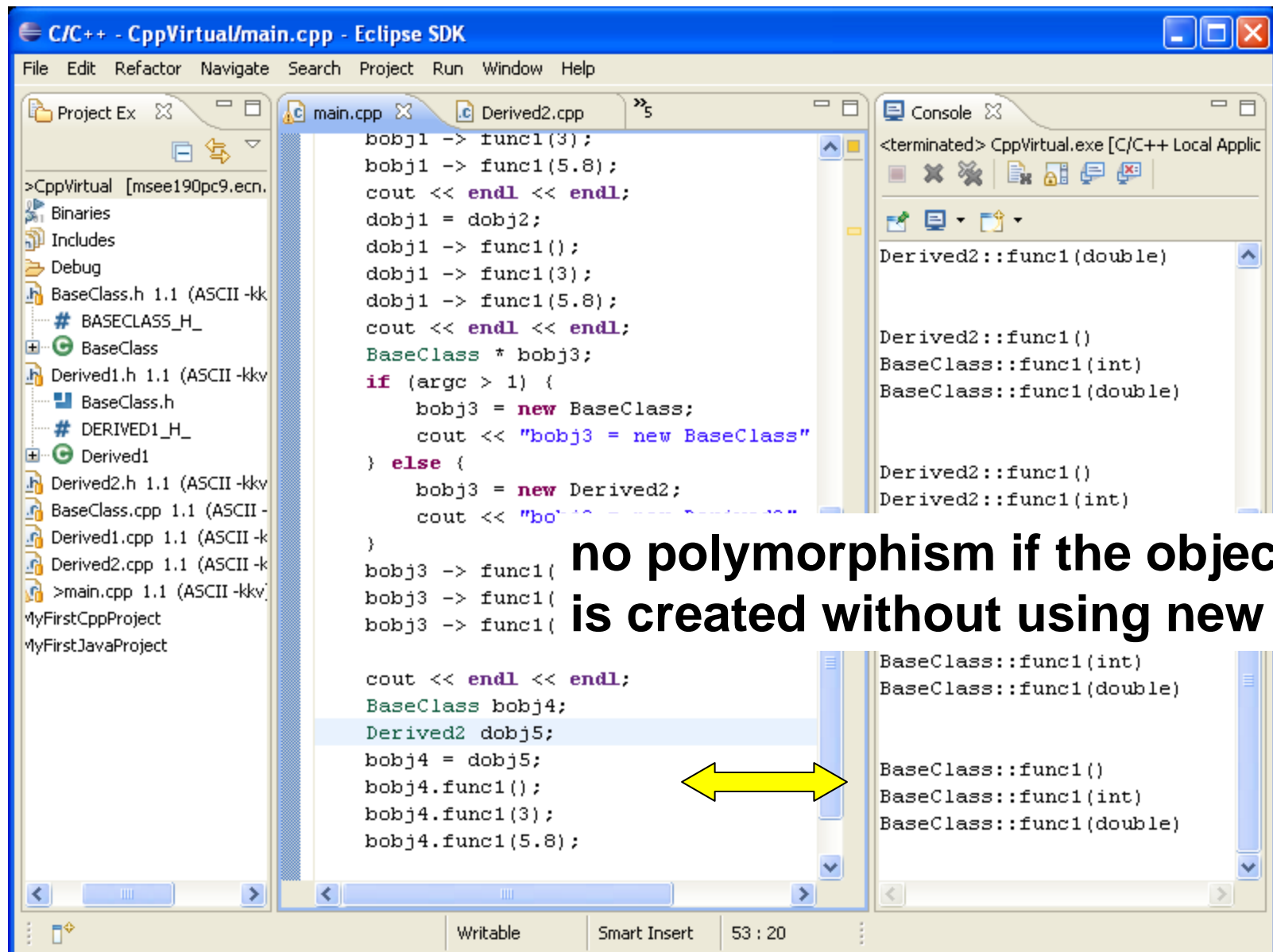












no polymorphism if the object is created without using new

Self Test

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

Constructor and Destructor (C++)

Yung-Hsiang Lu
yunlu@purdue.edu

Constructor (both C++ and Java)

- same name as the class' name (case sensitive)
- no return type
- usually public
- often overloaded
- usually create attribute objects by calling **new**
- "naturally" virtual (C++)

JLabel (Java 2 Platform SE 5.0) - Mozilla Firefox

File Edit View History Bookmarks Yahoo! Tools Help

http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/JLabel.html

Overview Package **Class** Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS

SUMMARY: [NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#)

DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

Java™ 2 Platform
Standard Ed. 5.0

javax.swing

Class JLabel

[java.lang.Object](#)

- [java.awt.Component](#)
 - [java.awt.Container](#)
 - [javax.swing.JComponent](#)
 - javax.swing.JLabel**

All Implemented Interfaces:

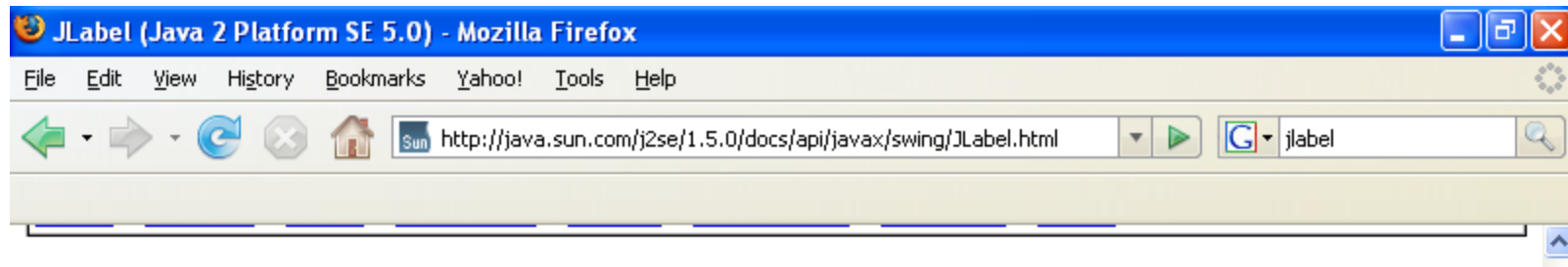
[ImageObserver](#), [MenuContainer](#), [Serializable](#), [Accessible](#), [SwingConstants](#)

Direct Known Subclasses:

[BasicComboBoxRenderer](#), [DefaultListCellRenderer](#), [DefaultTableCellRenderer](#), [DefaultTreeCellRenderer](#)

```
public class JLabel
extends JComponent
implements SwingConstants, Accessible
```

Done



Constructor Summary



JLabel constructor overloaded

[JLabel\(\)](#)

Creates a JLabel instance with no image and with an empty string for the title.

[JLabel\(Icon image\)](#)

Creates a JLabel instance with the specified image.

[JLabel\(Icon image, int horizontalAlignment\)](#)

Creates a JLabel instance with the specified image and horizontal alignment.

[JLabel\(String text\)](#)

Creates a JLabel instance with the specified text.

[JLabel\(String text, Icon icon, int horizontalAlignment\)](#)

Creates a JLabel instance with the specified text, image, and horizontal alignment.

[JLabel\(String text, int horizontalAlignment\)](#)

Creates a JLabel instance with the specified text and horizontal alignment.

Method Summary

protected int	checkHorizontalKey (int key, String message)
	Verify that key is a legal value for the horizontalAlignment properties.

Done

JLabel (Java 2 Platform SE 5.0) - Mozilla Firefox

File Edit View History Bookmarks Yahoo! Tools Help


http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/JLabel.html#JLat jlabel

Constructor Detail

JLabel

Constructors are usually public.

```
public JLabel(String text,  
             Icon icon,  
             int horizontalAlignment)
```




Creates a JLabel instance with the specified text, image, and horizontal alignment. The label is centered vertically in its display area. The text is on the trailing edge of the image.

Parameters:

- text - The text to be displayed by the label.
- icon - The image to be displayed by the label.
- horizontalAlignment - One of the following constants defined in SwingConstants: LEFT, CENTER, RIGHT, LEADING or TRAILING.

JLabel

```
public JLabel(String text,  
             int horizontalAlignment)
```



Creates a JLabel instance with the specified text and horizontal alignment. The label is centered vertically in its display

Done

Qt 4.3: QLabel Class Reference - Mozilla Firefox

File Edit View History Bookmarks Yahoo! Tools Help

http://doc.trolltech.com/4.3/qlabel.html

Google

Home · All Classes · Main Classes · Grouped Classes · Modules · Functions

TRÖLLTECH

QLabel Class Reference

[QtGui module]

The QLabel widget provides a text or image display. [More...](#)

```
#include <QLabel>
```

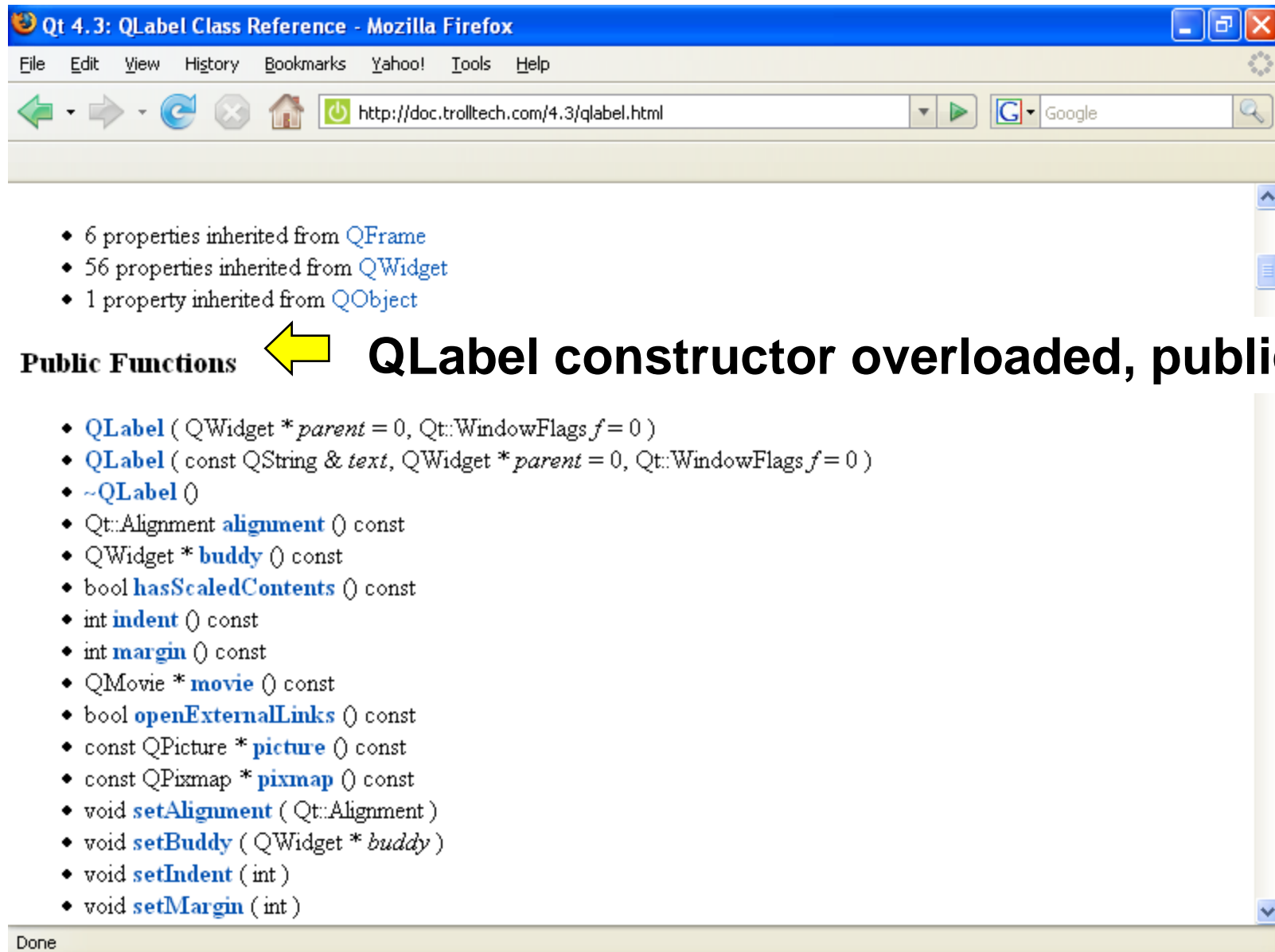
Inherits [QFrame](#).

- [List of all members, including inherited members](#)
- [Qt 3 support members](#)

Properties

• alignment : Qt::Alignment	• scaledContents : bool
• indent : int	• text : QString
• margin : int	• textFormat : Qt::TextFormat

Done




Qt 4.3: QLabel Class Reference - Mozilla Firefox

File Edit View History Bookmarks Yahoo! Tools Help

http://doc.trolltech.com/4.3/qlabel.html

Google

- ♦ 6 properties inherited from [QFrame](#)
- ♦ 56 properties inherited from [QWidget](#)
- ♦ 1 property inherited from [QObject](#)

Public Functions  **QLabel constructor overloaded, public**

- ♦ [QLabel](#) ([QWidget](#) *parent = 0, Qt::WindowFlags f = 0)
- ♦ [QLabel](#) (const [QString](#) &text, [QWidget](#) *parent = 0, Qt::WindowFlags f = 0)
- ♦ ~[QLabel](#) ()
- ♦ Qt::Alignment [alignment](#) () const
- ♦ [QWidget](#) * [buddy](#) () const
- ♦ bool [hasScaledContents](#) () const
- ♦ int [indent](#) () const
- ♦ int [margin](#) () const
- ♦ [QMovie](#) * [movie](#) () const
- ♦ bool [openExternalLinks](#) () const
- ♦ const [QPicture](#) * [picture](#) () const
- ♦ const [QPixmap](#) * [pixmap](#) () const
- ♦ void [setAlignment](#) (Qt::Alignment)
- ♦ void [setBuddy](#) ([QWidget](#) *buddy)
- ♦ void [setIndent](#) (int)
- ♦ void [setMargin](#) (int)

Done

Object Creation

- Java:

ClassName obj = new ClassName(parameters);

- C++

ClassName obj(parameters);

or

ClassName * obj = new ClassName(parameters);



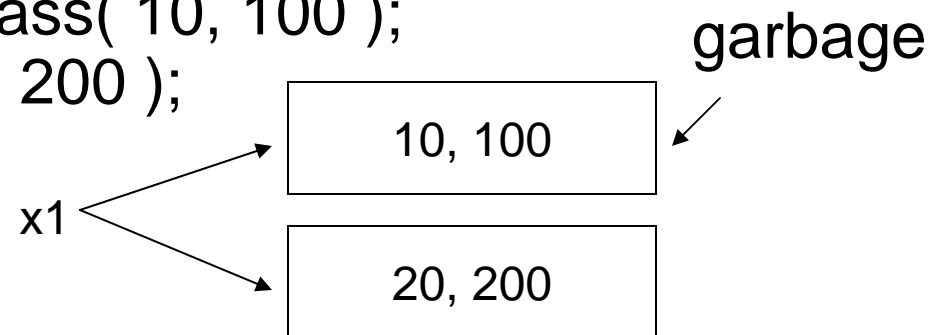
notice *

- In C++ and Java, a class is also a type (similar to int, char, or float).

"new" = Memory Allocation

- In Java, unused memory will be automatically reclaimed (called **garbage collection**).

```
AClass x1 = new AClass( 10, 100 );  
x1 = new AClass( 20, 200 );
```

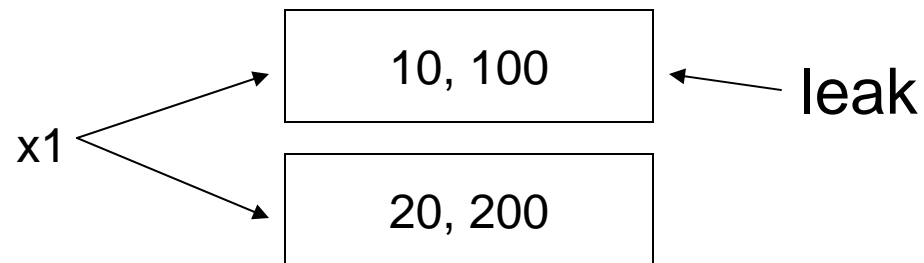


- It is unnecessary to worry about the objects that **cannot be reached**. Java reclaims the memory.
- You have to **remove all references** to the object.
- Too much garbage, however, **degrades performance**.

C++ Memory Leak

- In C++, unreachable memory is lost, "memory leak".

```
AClass * x1 = new AClass( 10, 100 ); // x1 is a pointer  
x1 = new AClass( 20, 200 );
```



- To prevent memory leak
delete x1;
- **Do not use malloc / free** in C++. They do not call the constructor / destructor.

new - delete

- If an object is created by calling "new", it should be removed by calling "delete".

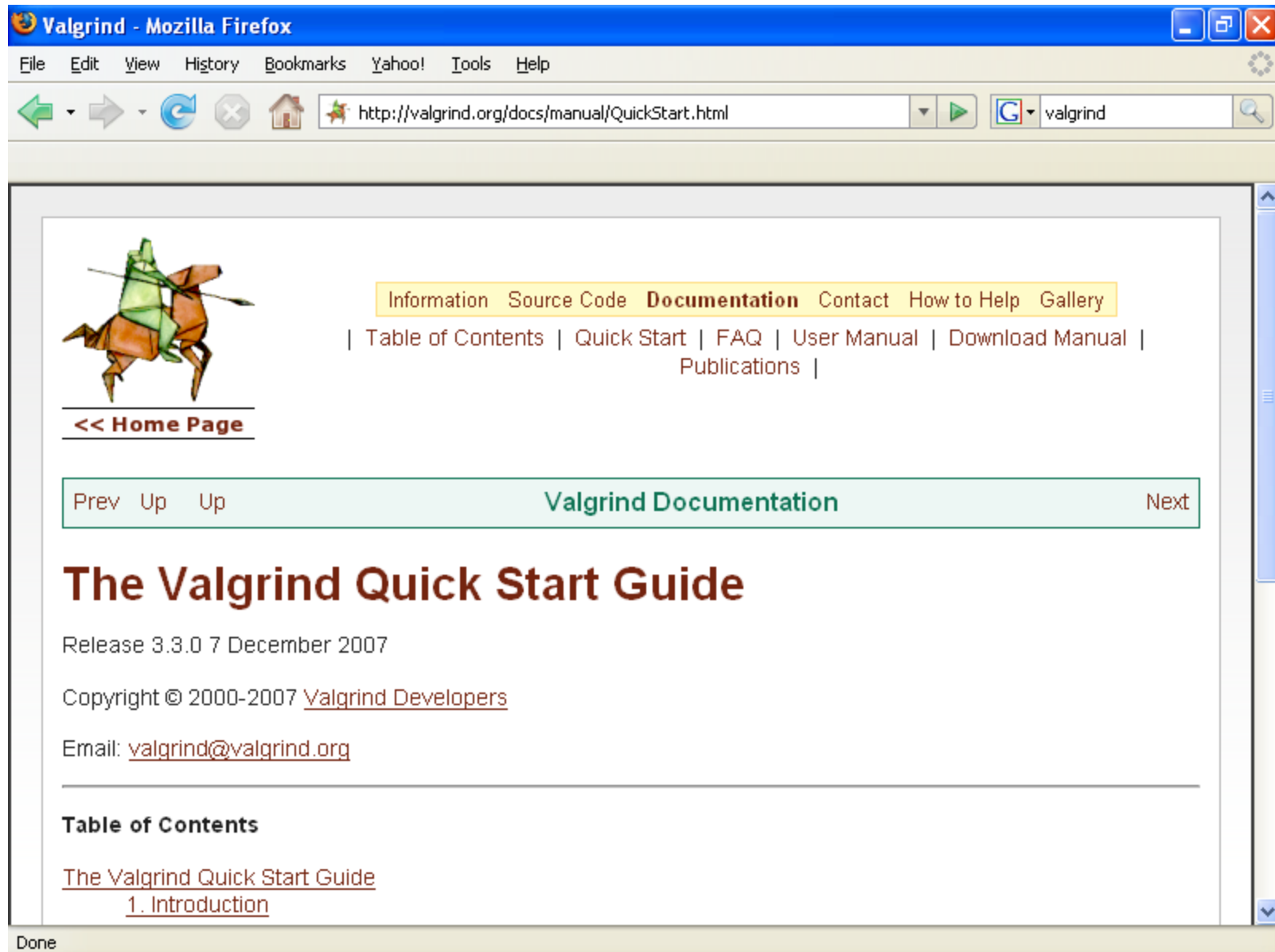
```
AClass * x1 = new AClass( 10, 100 );  
delete x1;
```

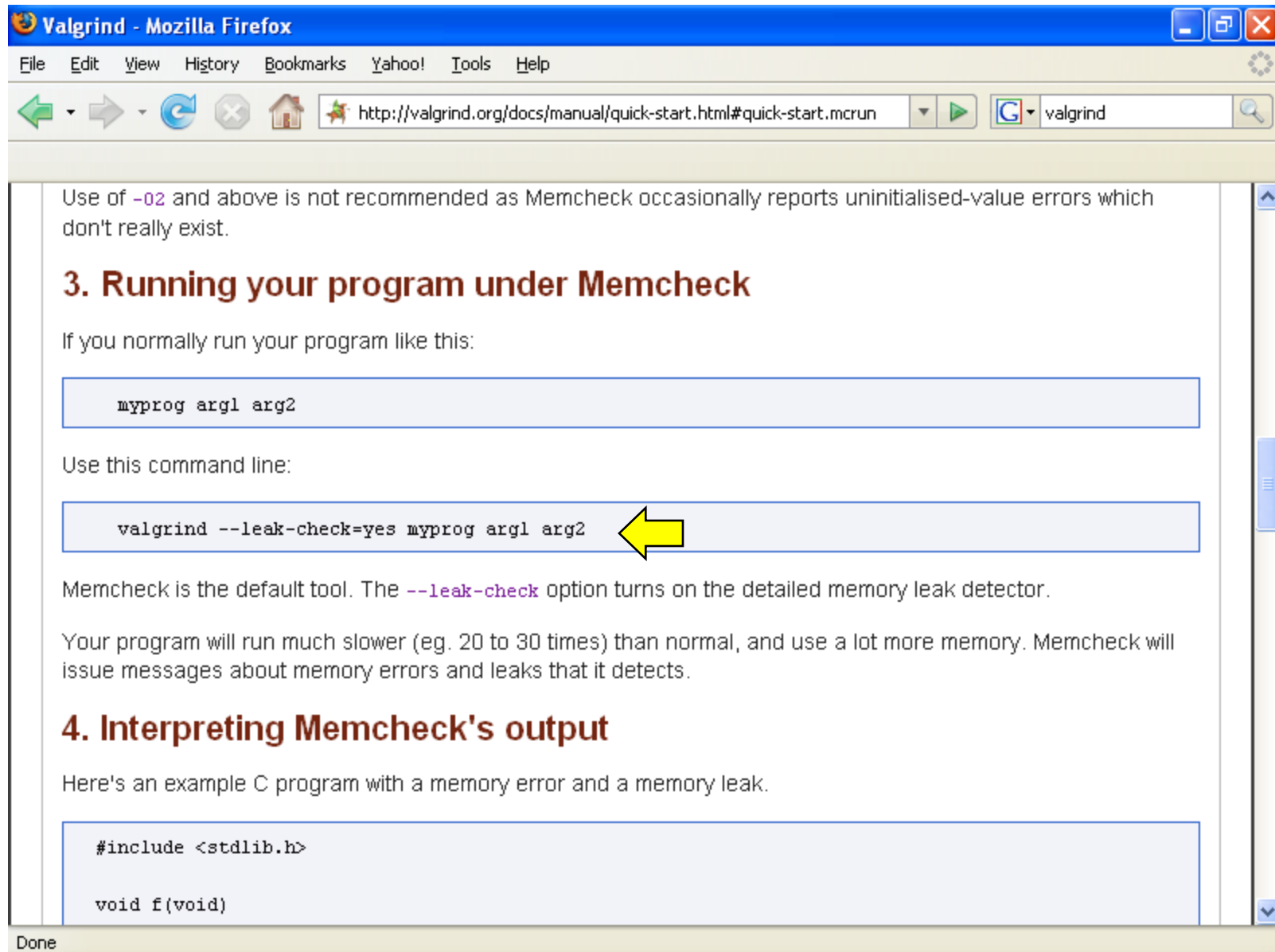
- new – delete pair
 - call delete only if new is called earlier
 - in the same level (not necessarily the same function)
 - an object can be deleted only once
- use “**valgrind --tool=memcheck --leak-check=yes executable**” to check memory leak

Memory Leak

- If allocated memory is not reclaimed, the program will gradually run out of memory and eventually crash.
- Memory leak is hard to detect by running the program
 - usually leak a small amount each time
 - can take hours, days, or weeks to run out of memory
 - sometime mistaken as performance problems
- In Java, if an object is no longer needed, remove all references to the object. It will be garbage collected.
- In C++, if an object is no longer needed and the object is created by calling **new**, **delete** the object.







Destructor (C++ only)

- add ~ in front of the class' name
- no return type
- usually public
- **cannot** take parameters \Rightarrow **cannot** be overloaded
- usually symmetric to constructor
- remove attribute objects by calling **delete** (if they are created by calling **new**)
- should always be **virtual**

Calling Order in Class Hierarchy

- In both C++ and Java
 - the constructor of the base class is called first
 - the constructor of the derived class is called later
 - ⇒ Constructors are naturally virtual because the constructor in the derived class is always called.
 - ⇒ C++ cannot add virtual in front of a constructor.
- In C++
 - the destructor of the derived class is called first
 - the destructor of the base class is called later
 - ⇒ need to add virtual to ensure the derived class' destructor is called

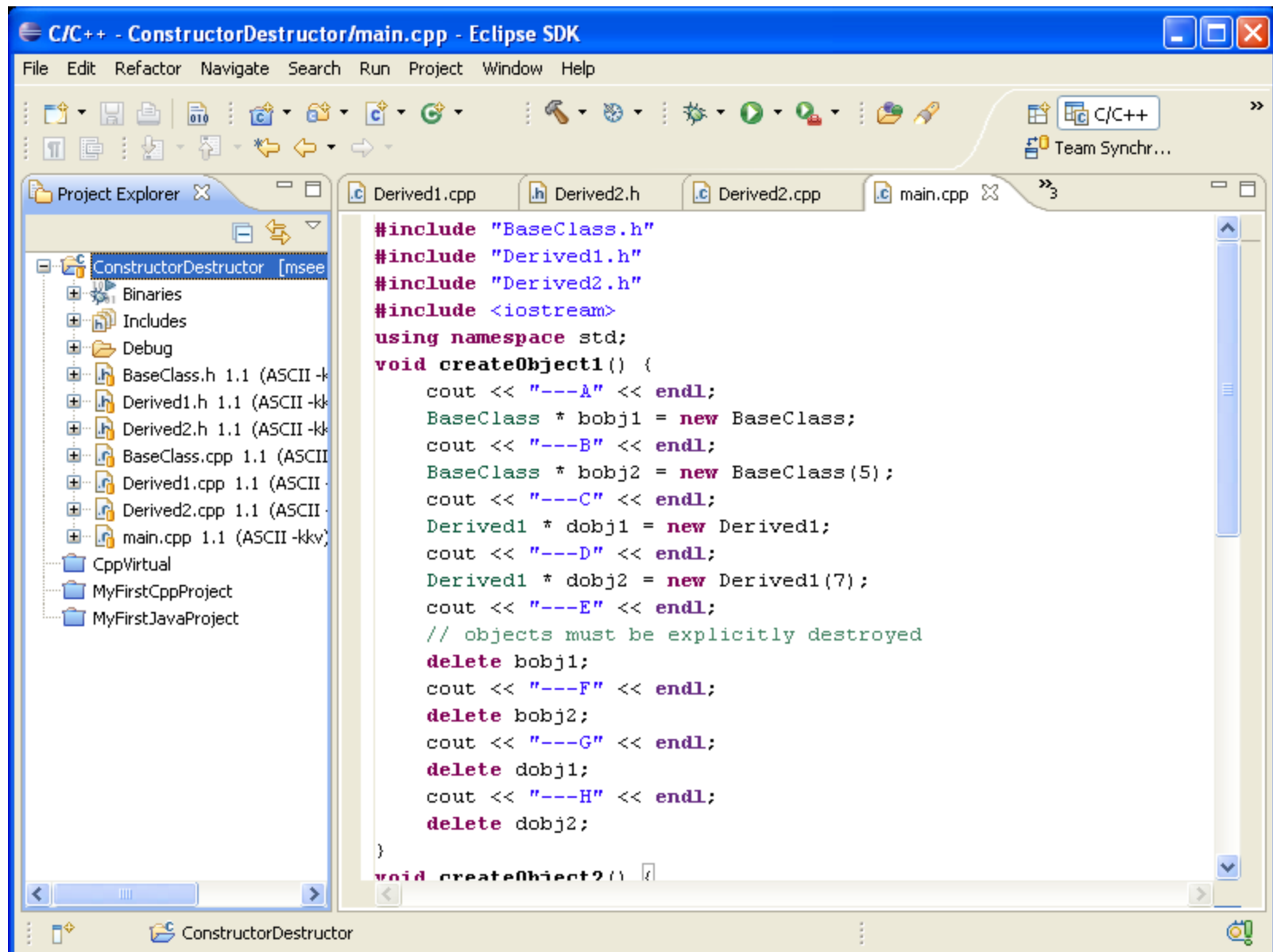
Constant Attributes

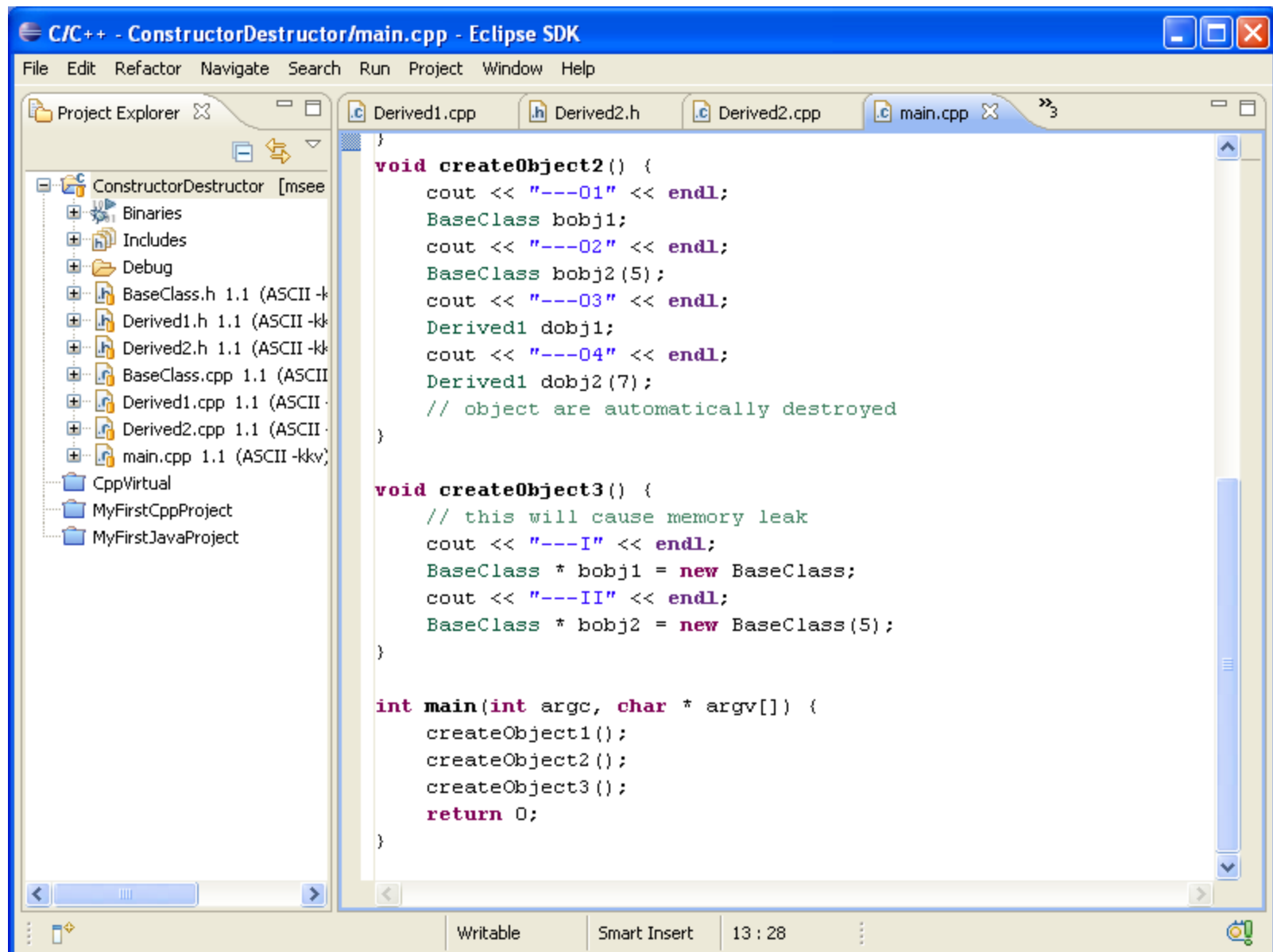
- Constant attributes must be initialized in constructors.
- C++

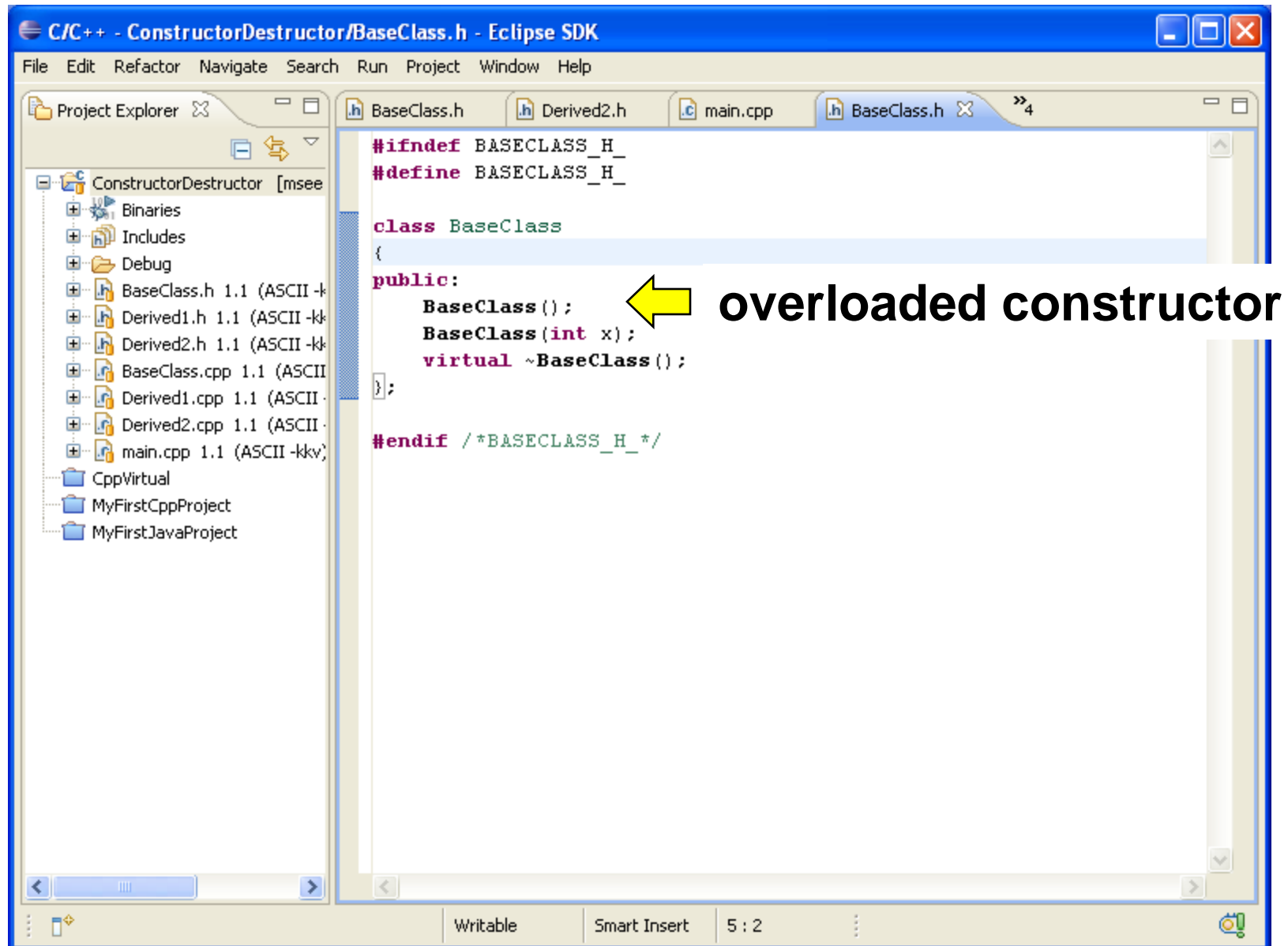
```
const type attributeName;  
constructor(...): attributeName(value) {  
    ...  
}
```

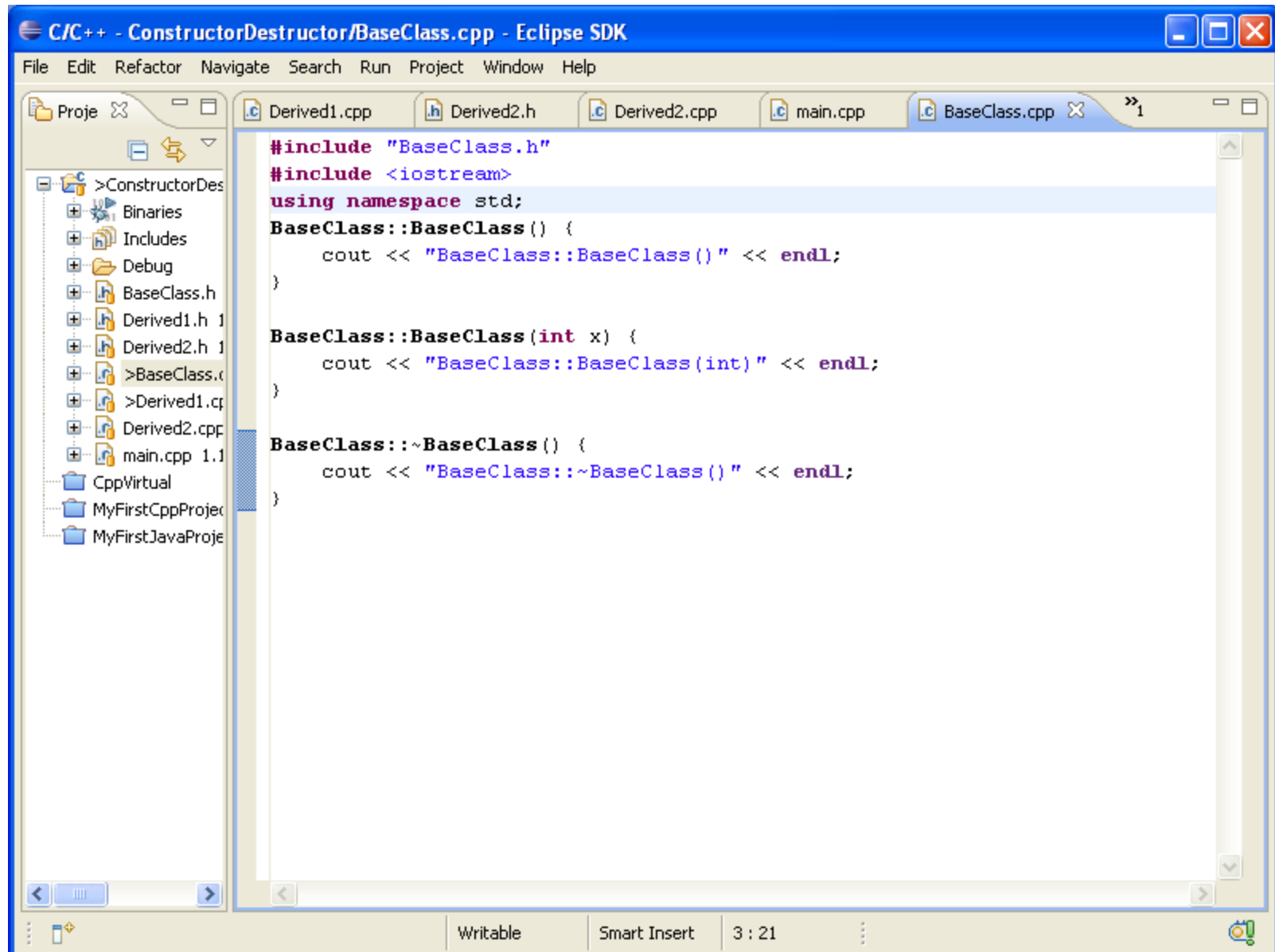
- Java

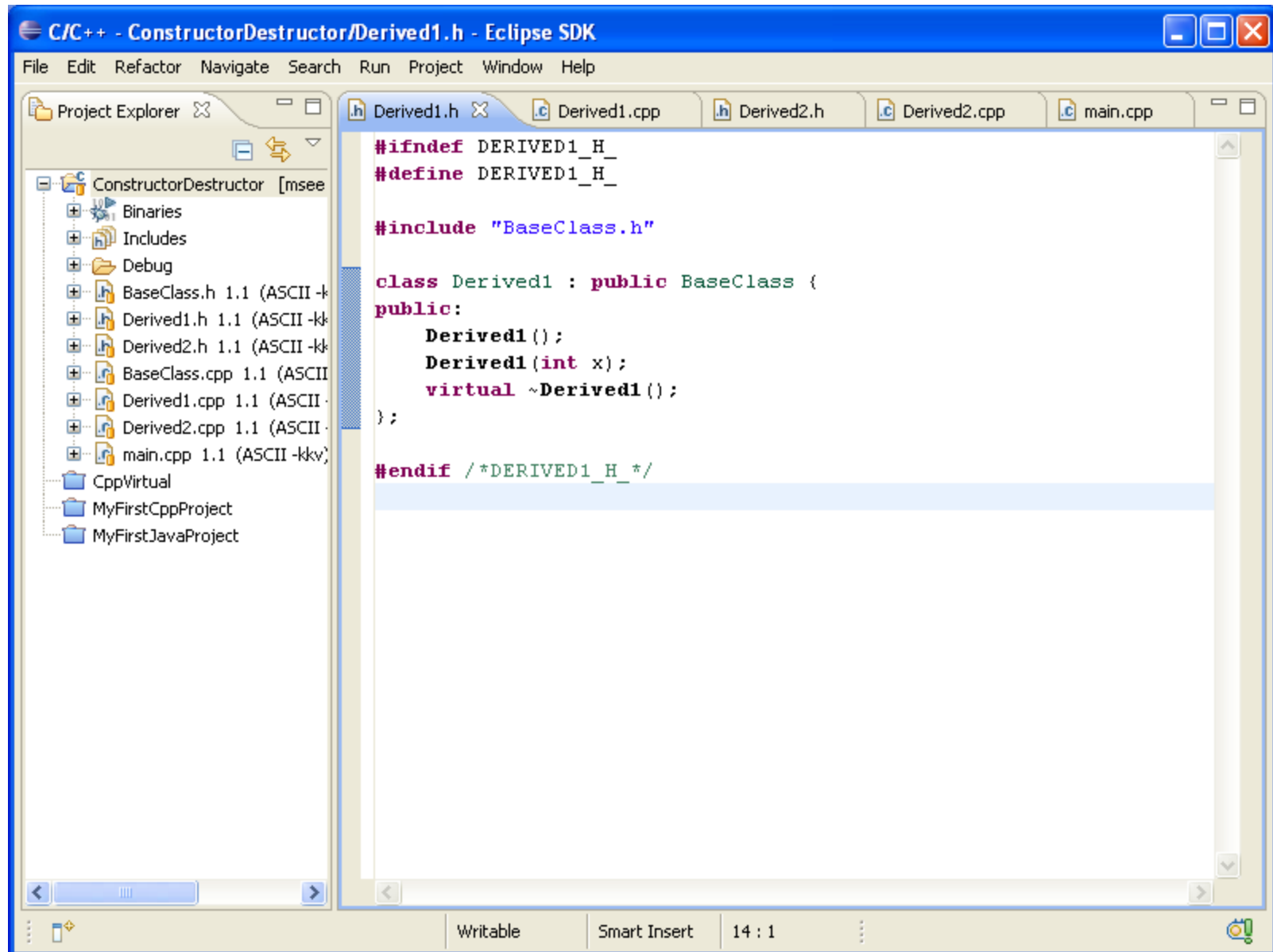
```
final type attributeName;  
⇒ assign value in constructor
```

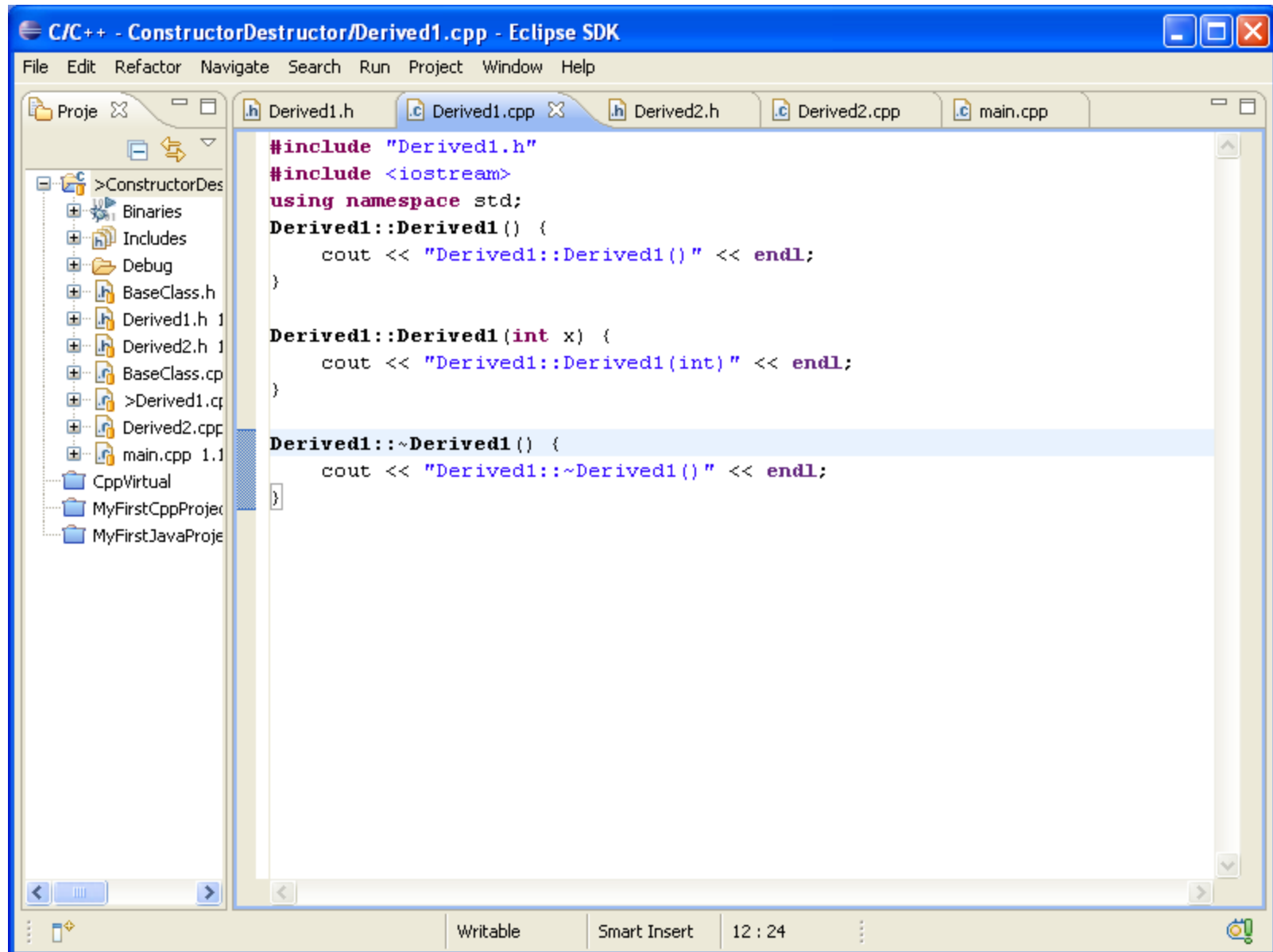


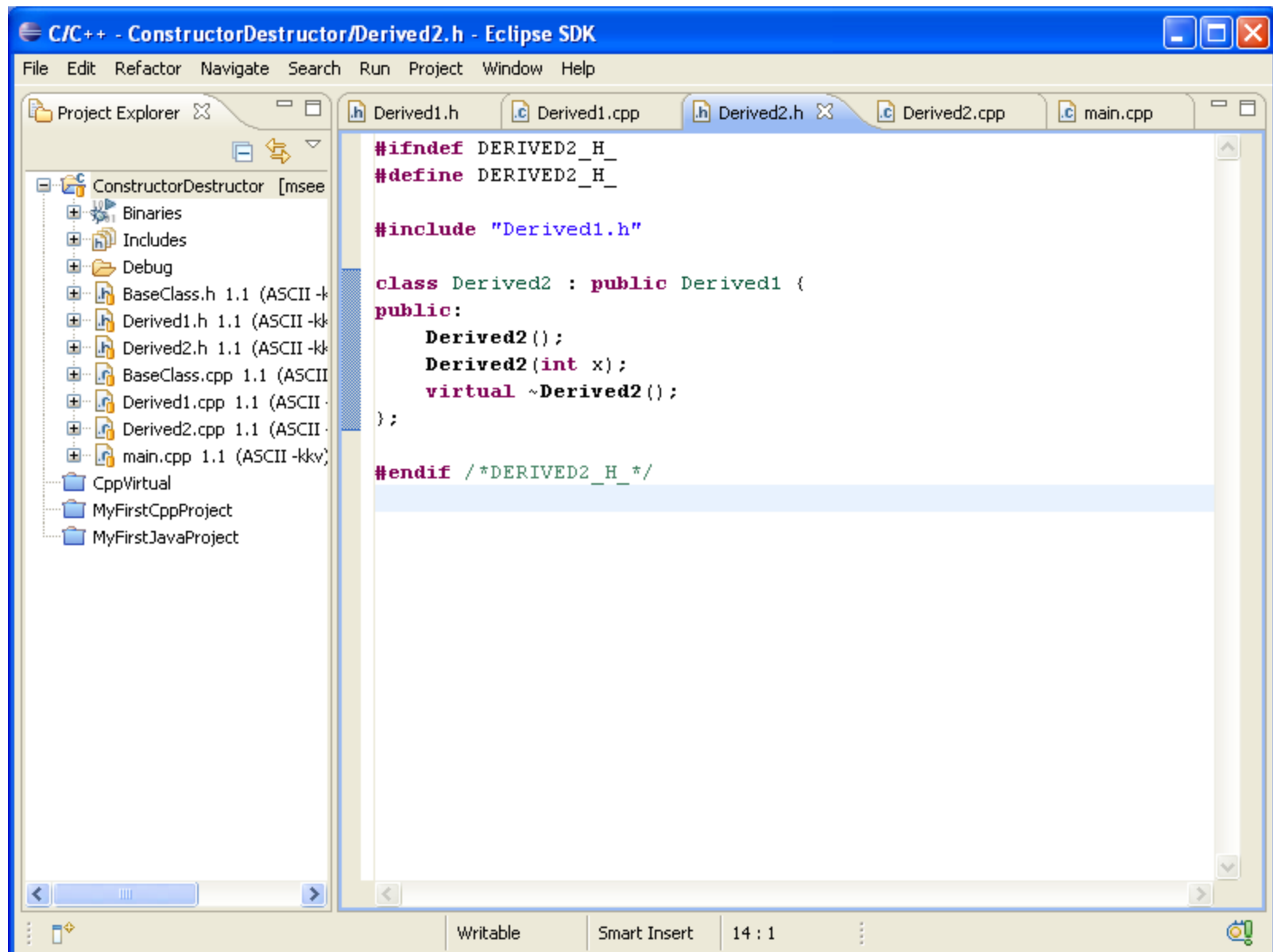


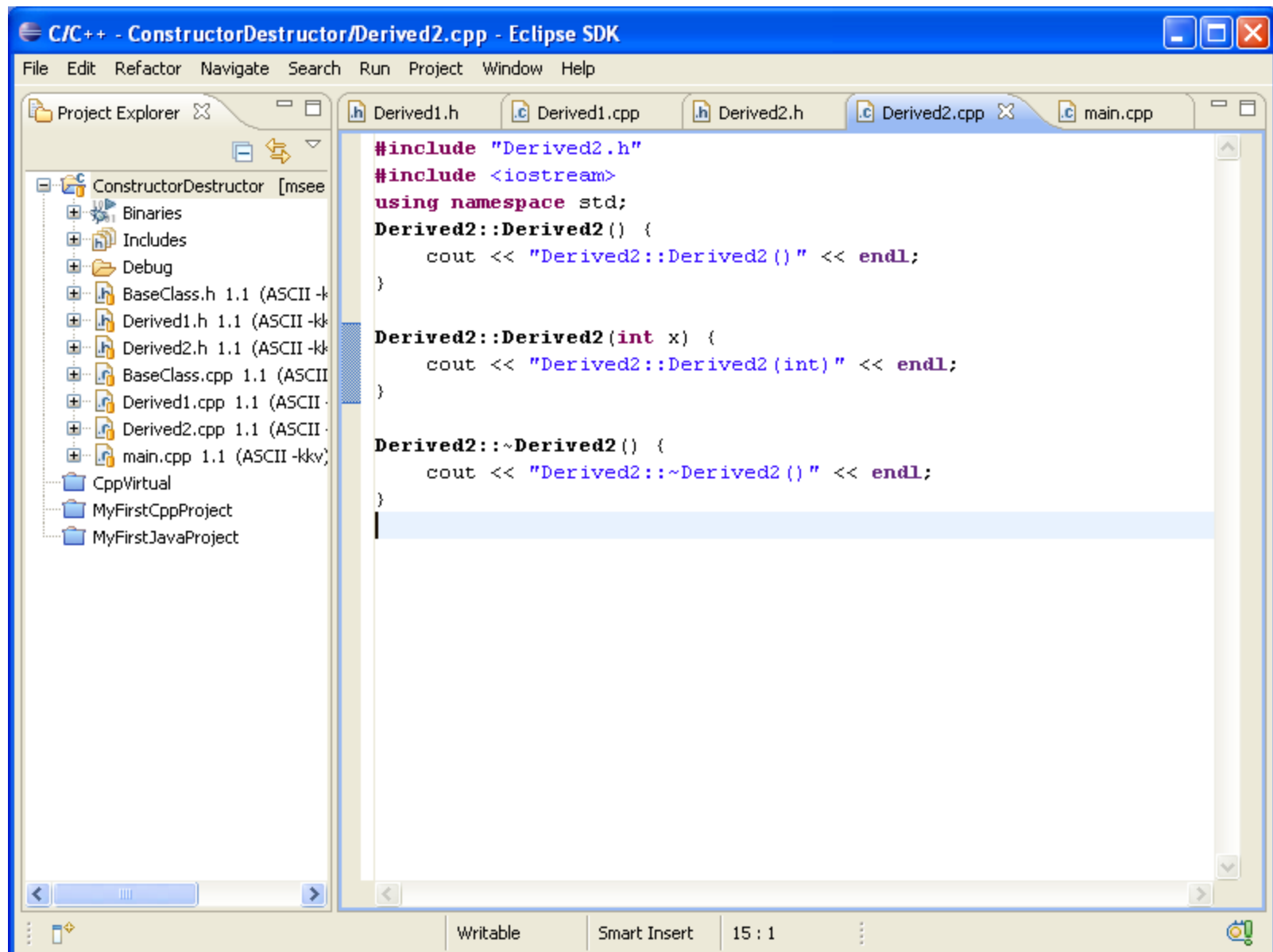












The screenshot shows the Eclipse IDE with the following components:

- Project Explorer:** Shows a project named 'ConstructorDestructor' with subfolders 'Binaries', 'Includes', 'Debug', and 'Base'. It also lists files 'BaseClass.h', 'Derived1.h', and 'Derived2.h'.
- Editor:** Displays the file 'main.cpp'. The code includes headers for 'BaseClass.h', 'Derived1.h', and 'Derived2.h', and uses the 'std' namespace. It defines two functions: 'createObject1()' and 'createObject2()'. 'createObject1()' creates objects 'bobj1' (BaseClass) and 'dobj2' (Derived1), prints their addresses, and deletes 'bobj1'. 'createObject2()' prints addresses for 'bobj1' and 'dobj2'.
- Console:** Shows the output of the program, which is a sequence of memory addresses for various constructor and destructor calls, labeled A through O2.

Code in main.cpp:

```
#include "BaseClass.h"
#include "Derived1.h"
#include "Derived2.h"
#include <iostream>
using namespace std;

void createObject1() {
    cout << "----A" << endl;
    BaseClass * bobj1 = new BaseClass;
    cout << "----B" << endl;

    Derived1 * dobj1 = new Derived1;
    cout << "----D" << endl;
    Derived1 * dobj2 = new Derived1(7);
    cout << "----E" << endl;
    // objects must be explicitly destroyed
    delete bobj1;
    cout << "----F" << endl;

    cout << "----H" << endl;
    delete dobj2;
}

void createObject2() {
    cout << "----O1" << endl;
    BaseClass bobj1;
    cout << "----O2" << endl;
}
```

Console Output:

```
<terminated> ConstructorDestructor.exe [C/C++ Local]
---A
BaseClass::BaseClass()
---B
BaseClass::BaseClass(int)
---C
BaseClass::BaseClass()
Derived1::Derived1()
---D
BaseClass::BaseClass()
Derived1::Derived1(int)
---E
BaseClass::~~BaseClass()
---F
BaseClass::~~BaseClass()
---G
Derived1::~~Derived1()
BaseClass::~~BaseClass()
---H
Derived1::~~Derived1()
BaseClass::~~BaseClass()
---O1
BaseClass::BaseClass()
---O2
BaseClass::BaseClass(int)
```

base constructor first

base destructor second

C/C++ - ConstructorDestructor/main.cpp - Eclipse SDK

File Edit Refactor Navigate Search Run Project Window Help

Project Explorer: >ConstructorDes, Binaries, Includes, Debug, BaseClass.h, Derived1.h, Derived2.h, >BaseClass.c, >Derived1.cp, Derived2.cpp, main.cpp 1.1, CppVirtual, MyFirstC++Project

Editor: main.cpp

```
delete bobj2;
cout << "----G" << endl;
delete dobj1;
cout << "----H" << endl;
delete dobj2;
}

void createObject2() {
    cout << "----01" << endl;
    BaseClass bobj1;
    cout << "----02" << endl;
    BaseClass bobj2(5);
    cout << "----03" << endl;
    Derived1 dobj1;
    cout << "----04" << endl;
}
```

objects automatically destroyed

Editor: main.cpp

```
void createObject3() {
    // this will cause memory leak
    cout << "----I" << endl;
    BaseClass * bobj1 = new BaseClass;
    cout << "----J" << endl;
}
```

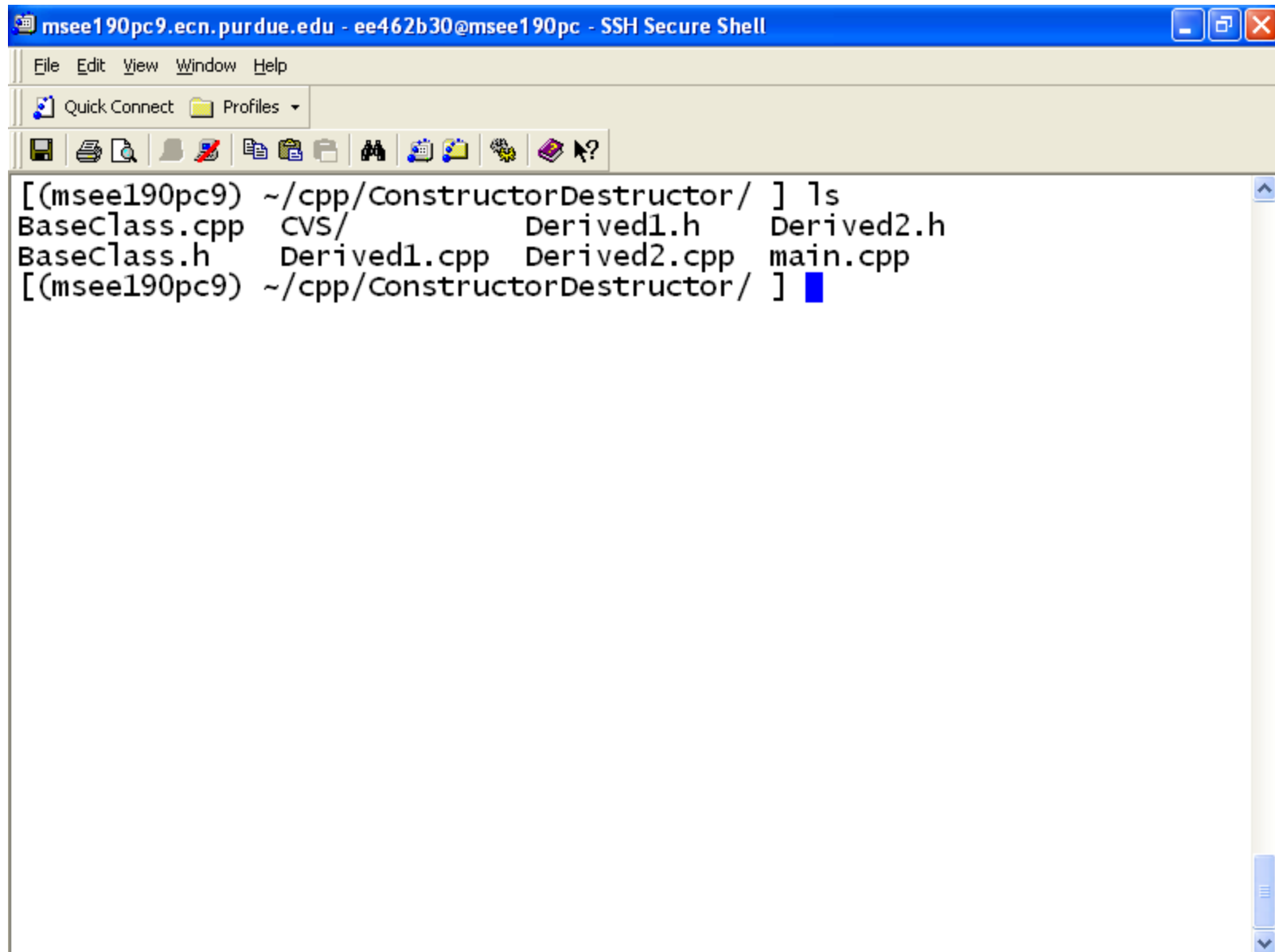
destructor not called, memory leak

Editor: main.cpp

```
int main(int argc, char * argv[]) {
    createObject1();
}
```

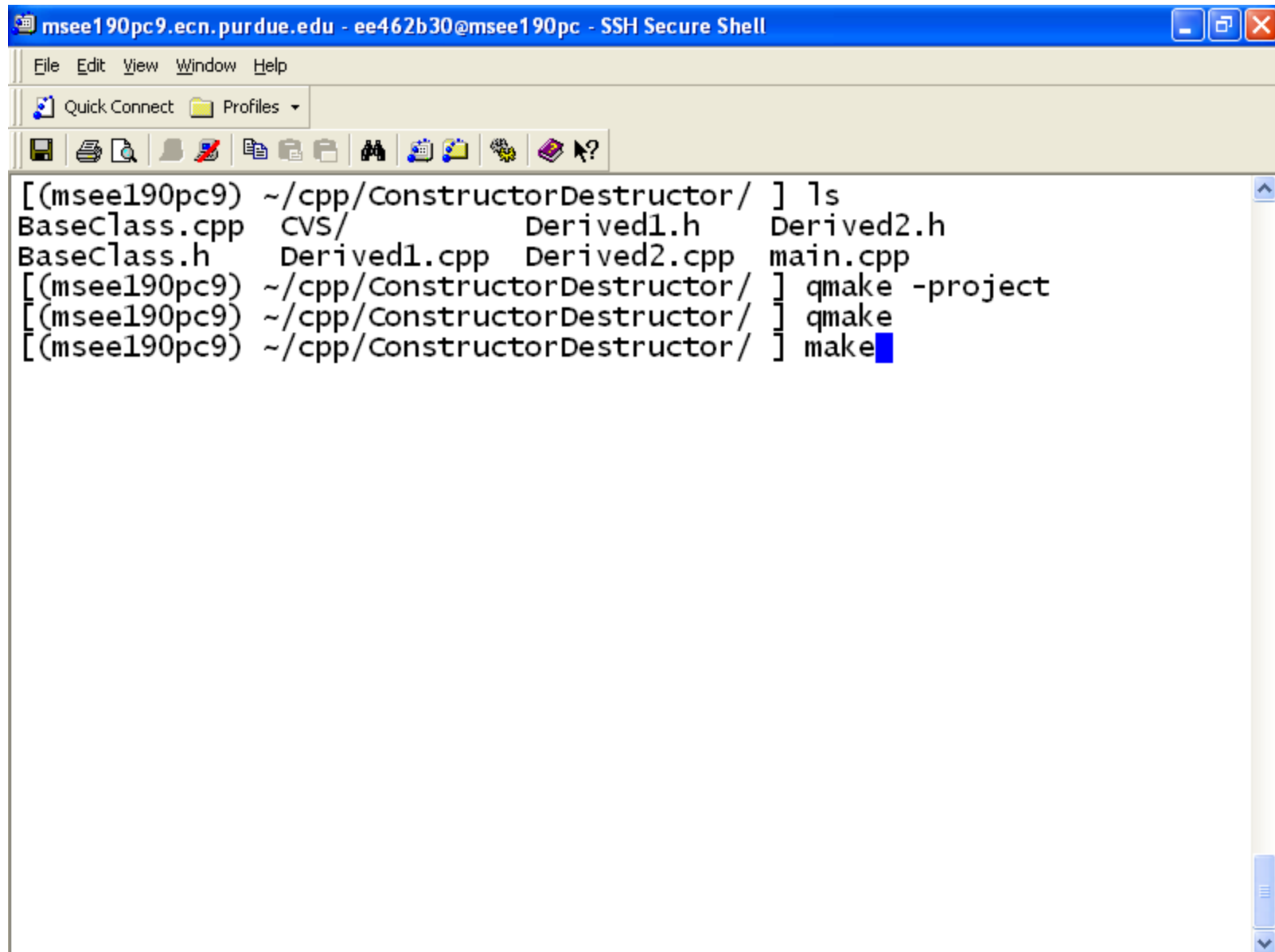
Console: <terminated> ConstructorDestructor.exe [C/C++ Local]

```
----H
Derived1::~Derived1()
BaseClass::~BaseClass()
----01
BaseClass::BaseClass()
----02
BaseClass::BaseClass(int)
----03
BaseClass::BaseClass()
Derived1::Derived1()
----04
BaseClass::BaseClass()
Derived1::Derived1(int)
Derived1::~Derived1()
BaseClass::~BaseClass()
Derived1::~Derived1()
BaseClass::~BaseClass()
BaseClass::~BaseClass()
BaseClass::~BaseClass()
----I
BaseClass::BaseClass()
----II
BaseClass::BaseClass(int)
```



The image shows a screenshot of an SSH Secure Shell window. The title bar at the top reads "msee190pc9.ecn.purdue.edu - ee462b30@msee190pc - SSH Secure Shell". Below the title bar is a menu bar with "File", "Edit", "View", "Window", and "Help". Under the "View" menu, there are options for "Quick Connect" and "Profiles". A toolbar with various icons is located below the menu bar. The main area of the window is a terminal displaying the output of a "ls" command. The prompt is "[msee190pc9) ~/cpp/ConstructorDestructor/]". The output lists the following files and directories: "BaseClass.cpp", "CVS/", "Derived1.h", "Derived2.h", "BaseClass.h", "Derived1.cpp", "Derived2.cpp", and "main.cpp". The prompt is repeated on the next line, followed by a blue cursor.

```
[msee190pc9) ~/cpp/ConstructorDestructor/ ] ls
BaseClass.cpp  CVS/           Derived1.h     Derived2.h
BaseClass.h    Derived1.cpp   Derived2.cpp   main.cpp
[msee190pc9) ~/cpp/ConstructorDestructor/ ]
```



```
msee190pc9.ecn.purdue.edu - ee462b30@msee190pc - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
[Icons: Save, Print, Find, Copy, Paste, Undo, Redo, Run, Stop, Help]

[(msee190pc9) ~/cpp/ConstructorDestructor/ ] ls
BaseClass.cpp  CVS/          Derived1.h    Derived2.h
BaseClass.h    Derived1.cpp  Derived2.cpp  main.cpp
[(msee190pc9) ~/cpp/ConstructorDestructor/ ] qmake -project
[(msee190pc9) ~/cpp/ConstructorDestructor/ ] qmake
[(msee190pc9) ~/cpp/ConstructorDestructor/ ] make
```

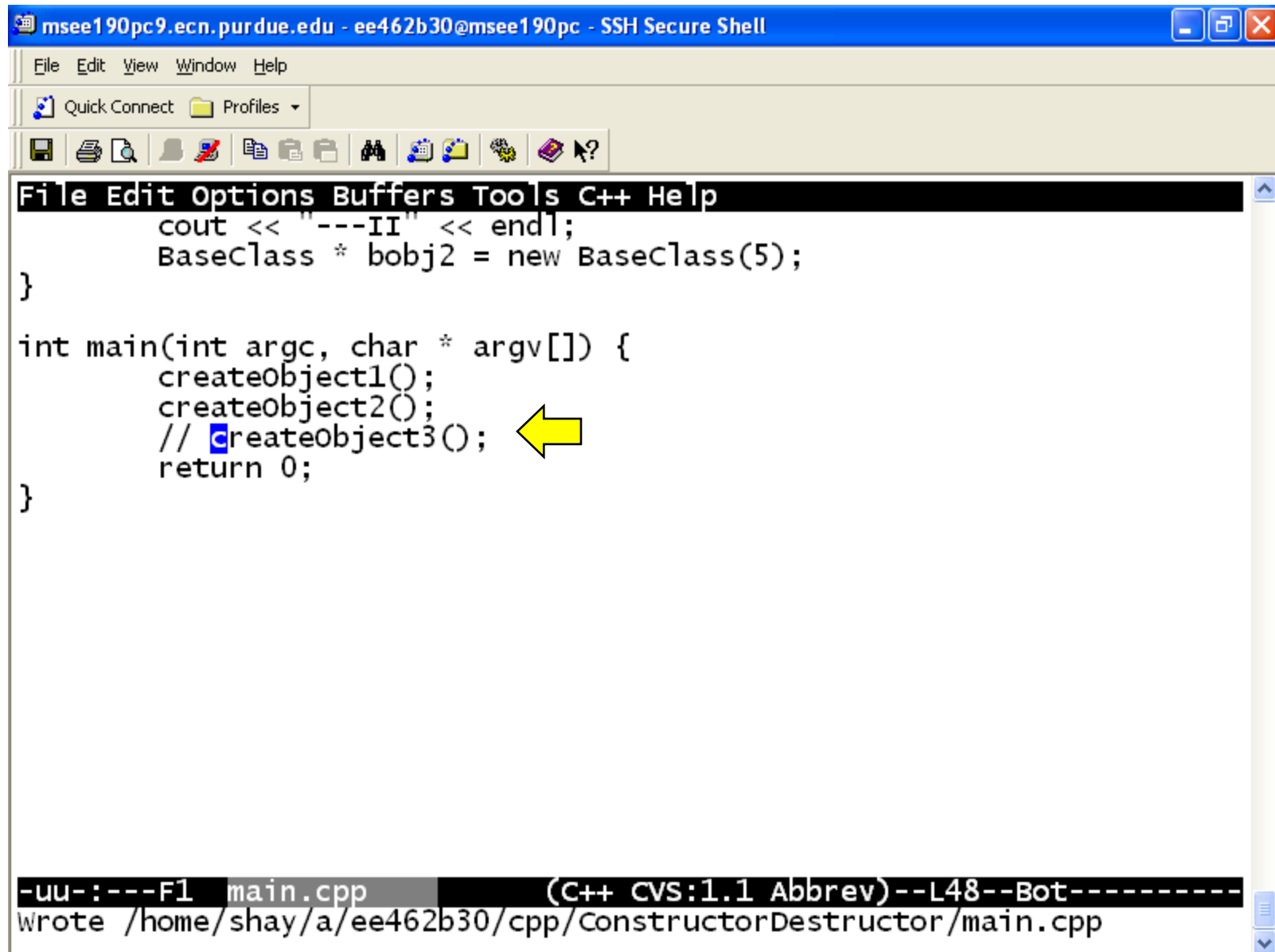


```
msee190pc9.ecn.purdue.edu - ee462b30@msee190pc - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
[(msee190pc9) ~/cpp/ConstructorDestructor/ ] valgrind --leak-check=yes
./ConstructorDestructor
```

```
msee190pc9.ecn.purdue.edu - ee462b30@msee190pc - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
loc.c:168)
==15112== by 0x4016C2: createObject3() (in /home/shay/a/ee462b30/cpp/ConstructorDestructor/ConstructorDestructor)
==15112== by 0x401712: main (in /home/shay/a/ee462b30/cpp/ConstructorDestructor/ConstructorDestructor)
==15112==
==15112==
==15112== 8 bytes in 1 blocks are definitely lost in loss record 2 of 2
==15112== at 0x4904DB5: operator new(unsigned long) (vg_replace_malloc.c:168)
==15112== by 0x401696: createObject3() (in /home/shay/a/ee462b30/cpp/ConstructorDestructor/ConstructorDestructor)
==15112== by 0x401712: main (in /home/shay/a/ee462b30/cpp/ConstructorDestructor/ConstructorDestructor)
==15112==
==15112== LEAK SUMMARY:
==15112== definitely lost: 16 bytes in 2 blocks.
==15112== possibly lost: 0 bytes in 0 blocks.
==15112== still reachable: 0 bytes in 0 blocks.
==15112== suppressed: 0 bytes in 0 blocks.
==15112== Reachable blocks (those to which a pointer was found) are not shown.
==15112== To see them, rerun with: --show-reachable=yes
[(msee190pc9) ~/cpp/ConstructorDestructor/ ]
```

leak detected





The screenshot shows a C++ IDE window titled "msee190pc9.ecn.purdue.edu - ee462b30@msee190pc - SSH Secure Shell". The menu bar includes "File", "Edit", "View", "Window", and "Help". Below the menu bar is a toolbar with icons for "Quick Connect" and "Profiles". The main editor area has a menu bar with "File", "Edit", "Options", "Buffers", "Tools", "C++", and "Help". The code in the editor is as follows:

```
cout << "---II" << endl;
BaseClass * bobj2 = new BaseClass(5);
}

int main(int argc, char * argv[]) {
    createObject1();
    createObject2();
    // createObject3();
    return 0;
}
```

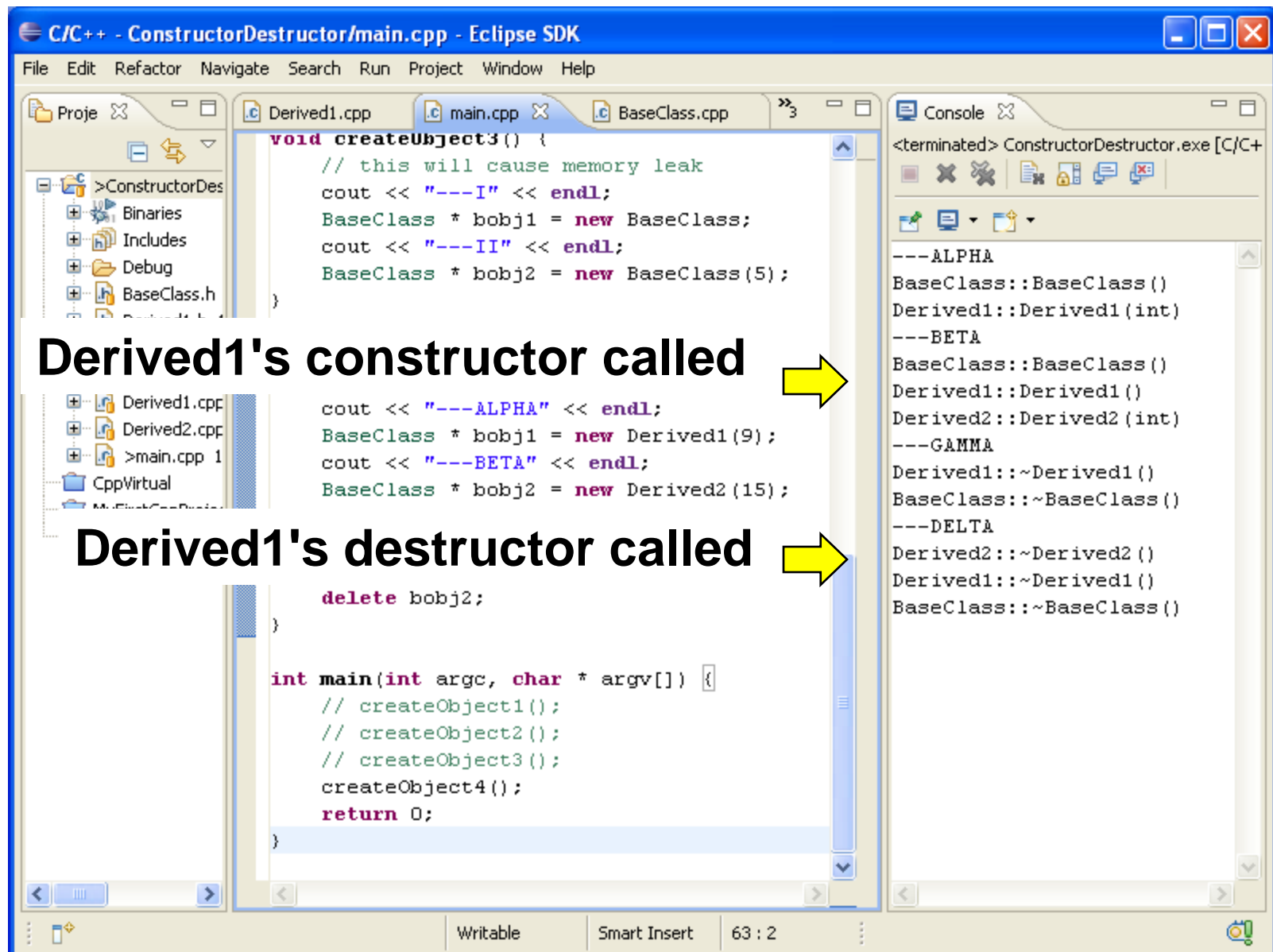
A yellow arrow points to the line `// createObject3();`. The status bar at the bottom displays the following information:

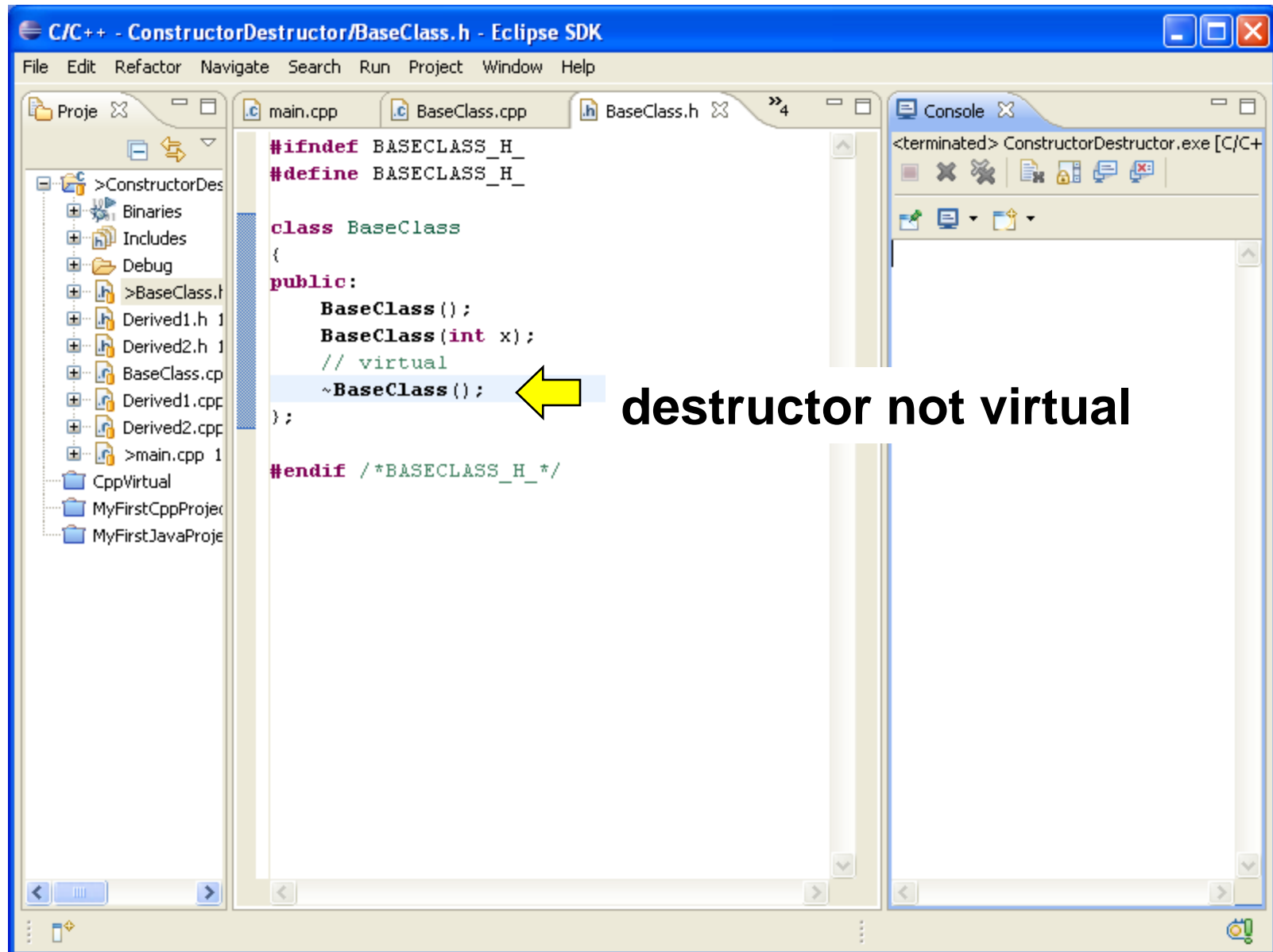
-uu:---F1 main.cpp (C++ CVS:1.1 Abbrev)--L48--Bot-----
wrote /home/shay/a/ee462b30/cpp/ConstructorDestructor/main.cpp

```
msee190pc9.ecn.purdue.edu - ee462b30@msee190pc - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
BaseClass::~~BaseClass()
---01
BaseClass::BaseClass()
---02
BaseClass::BaseClass(int)
---03
BaseClass::BaseClass()
Derived1::Derived1()
---04
BaseClass::BaseClass()
Derived1::Derived1(int)
Derived1::~~Derived1()
BaseClass::~~BaseClass()
Derived1::~~Derived1()
BaseClass::~~BaseClass()
BaseClass::~~BaseClass()
BaseClass::~~BaseClass()
==15200==
==15200== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 5 from
2)
==15200== malloc/free: in use at exit: 0 bytes in 0 blocks.
==15200== malloc/free: 16 allocs, 16 frees, 464 bytes allocated.
==15200== For counts of detected errors, rerun with: -v
==15200== All heap blocks were freed -- no leaks are possible.
[(msee190pc9) ~/cpp/ConstructorDestructor/ ]
```

no leak







Derived1's constructor called →

Derived1's destructor **not called** →

⇒ possible memory leak

```
void createObject3() {
    // this will cause memory leak
    cout << "----I" << endl;
    BaseClass * bobj1 = new BaseClass;
    cout << "----II" << endl;
    BaseClass * bobj2 = new BaseClass(5);
}

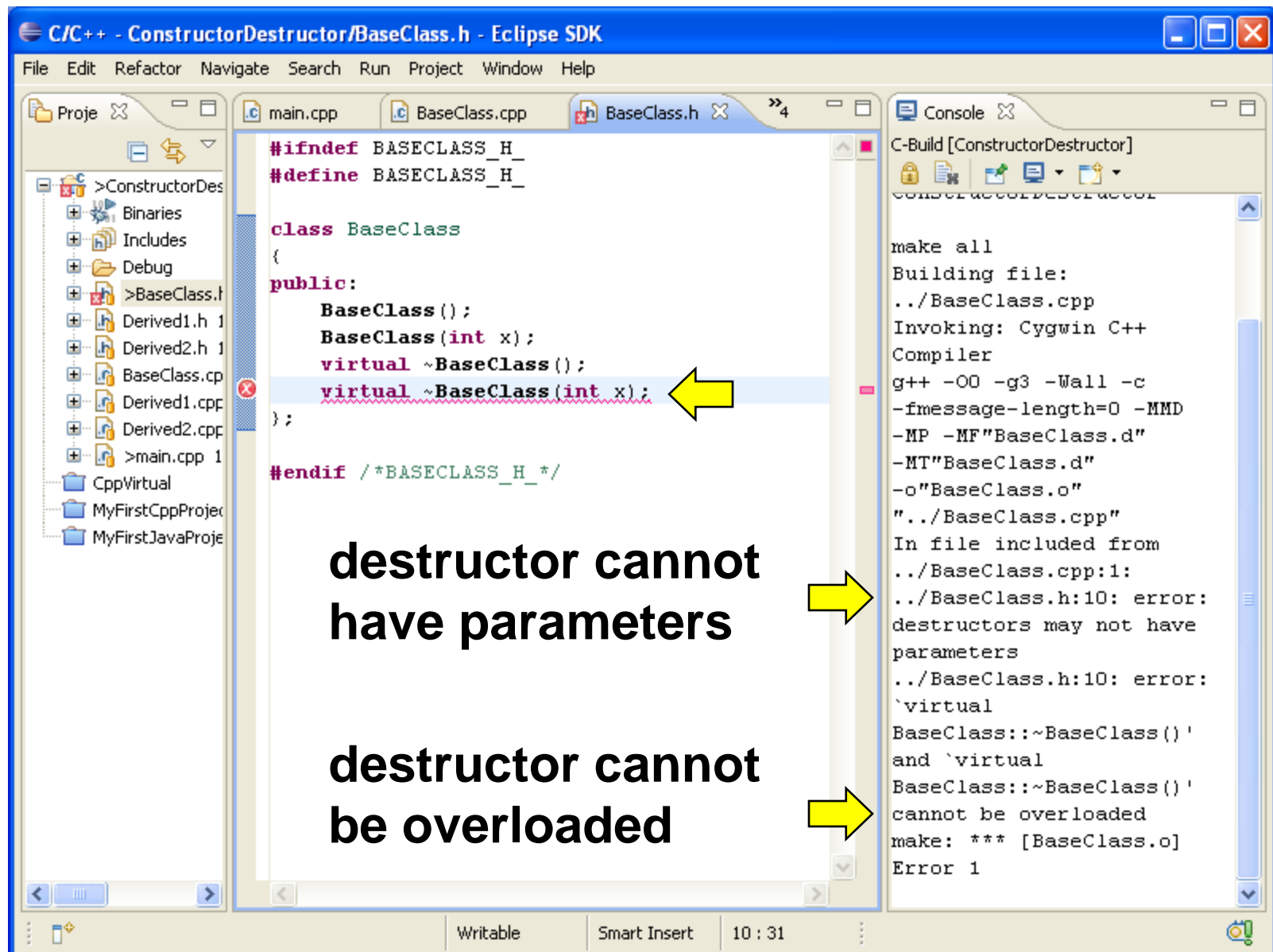
cout << "----ALPHA" << endl;
BaseClass * bobj1 = new Derived1(9);

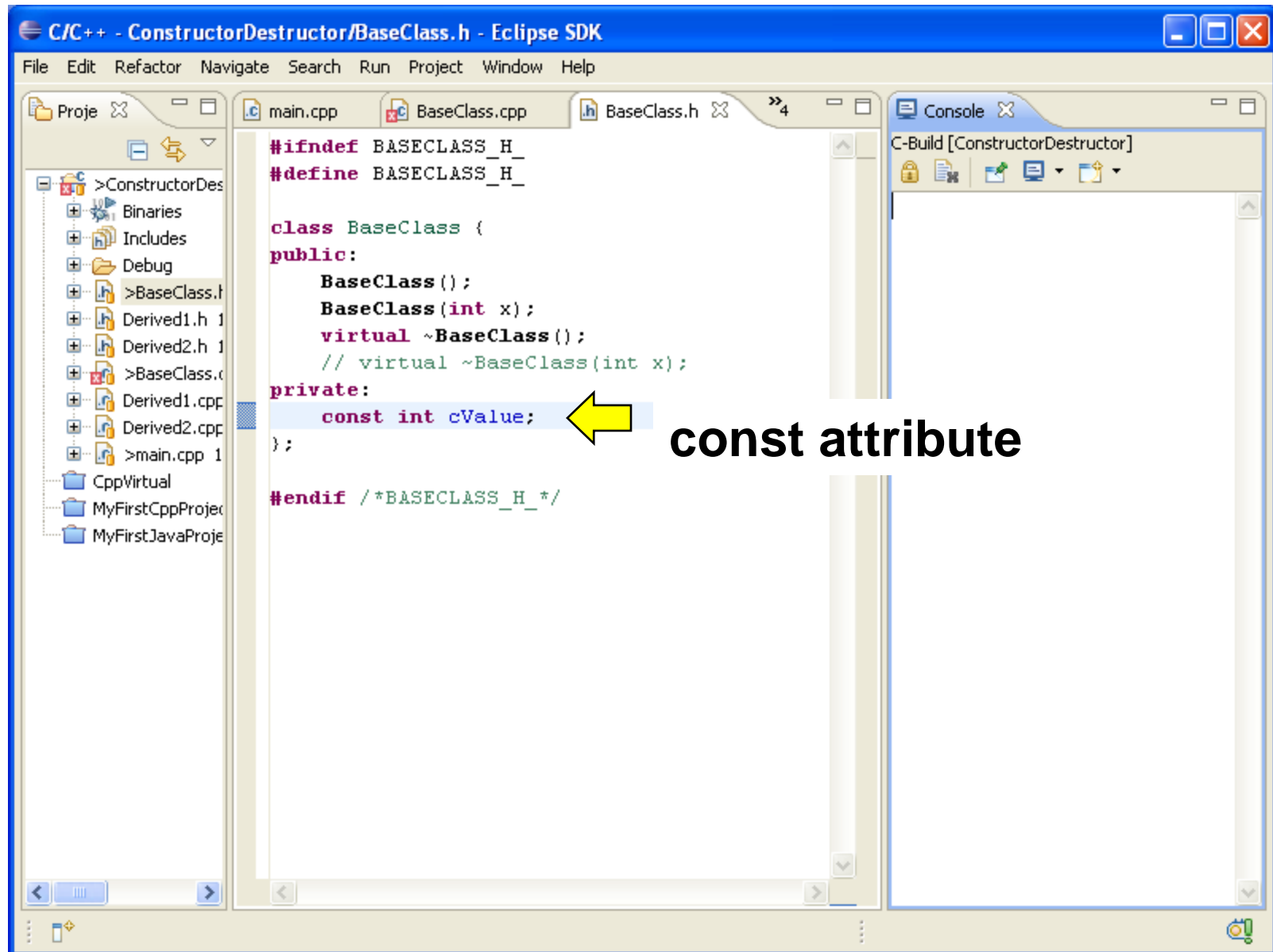
delete bobj4;

int main(int argc, char * argv[]) {
    // createObject1();
    // createObject2();
    // createObject3();
    createObject4();
    return 0;
}
```

Console Output:

```
<terminated> ConstructorDestructor.exe [C/C++
---ALPHA
BaseClass::BaseClass()
Derived1::Derived1(int)
---BETA
BaseClass::BaseClass()
Derived1::Derived1()
Derived2::Derived2(int)
---GAMMA
BaseClass::~~BaseClass()
---DELTA
BaseClass::~~BaseClass()
```





The screenshot shows the Eclipse IDE with the following components:

- Project Explorer:** Shows a project named 'ConstructorDestructor' containing files like 'main.cpp', 'BaseClass.cpp', 'BaseClass.h', 'Derived1.h', 'Derived2.h', 'Derived1.cpp', 'Derived2.cpp', and 'main.cpp'.
- Editor:** Displays the code for 'BaseClass.cpp'. The code includes 'BaseClass.h' and 'iostream', uses the 'std' namespace, and defines three functions: a default constructor, a constructor taking an 'int' parameter, and a destructor. Red squiggly lines indicate errors in the constructor definitions.
- Console:** Shows the output of the C-Build process. It lists the compiler options and the compilation command. The output shows two errors: 'uninitialized member' for 'BaseClass::cValue' in the default constructor (line 4) and the 'int' constructor (line 8).

Code in BaseClass.cpp:

```
#include "BaseClass.h"
#include <iostream>
using namespace std;

BaseClass::BaseClass() {
    cout << "BaseClass::BaseClass() " << endl
}

BaseClass::BaseClass(int x) {
    cout << "BaseClass::BaseClass(int) " << endl
}

BaseClass::~~BaseClass() {
    cout << "BaseClass::~~BaseClass() " << endl
}
```

Console Output:

```
C-Build [ConstructorDestructor]
Building file:
../BaseClass.cpp
Invoking: Cygwin C++
Compiler
g++ -O0 -g3 -Wall -c
-fmessage-length=0 -MMD
-MP -MF"BaseClass.d"
-MT"BaseClass.d"
-o"BaseClass.o"
"../BaseClass.cpp"
../BaseClass.cpp: In
constructor
`BaseClass::BaseClass()':
../BaseClass.cpp:4: error:
uninitialized member
`BaseClass::cValue' with
`const' type `const int'
../BaseClass.cpp: In
constructor
`BaseClass::BaseClass(int)':
../BaseClass.cpp:8: error:
uninitialized member
`BaseClass::cValue' with
`const' type `const int'
make: *** [BaseClass.o]
Error 1
```

Text Overlay: A yellow arrow points from the text "const attribute must be initialized in constructor" to the error messages in the console.

The screenshot shows the Eclipse IDE with the following components:

- Project Explorer:** Shows a project named 'ConstructorDestructor' containing files like 'main.cpp', 'BaseClass.cpp', 'BaseClass.h', 'Derived1.h', 'Derived2.h', 'Derived1.cpp', 'Derived2.cpp', and 'main.cpp'.
- Editor:** Displays the code for 'BaseClass.cpp'. The code includes 'BaseClass.h' and 'iostream', uses the 'std' namespace, and defines a 'BaseClass' with a static member variable 'cValue'. The code is as follows:

```
#include "BaseClass.h"
#include <iostream>
using namespace std;

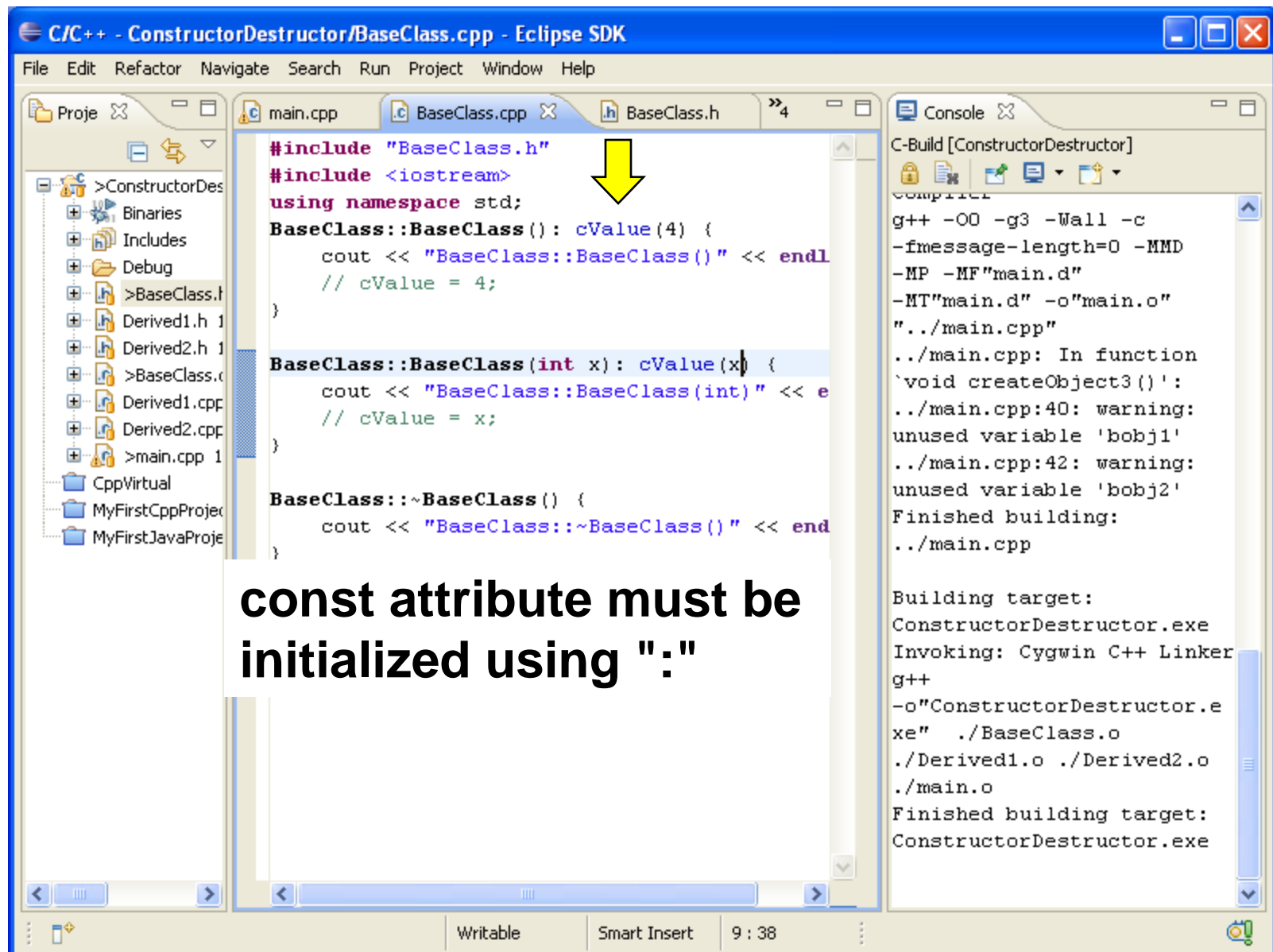
BaseClass::BaseClass() {
    cout << "BaseClass::BaseClass() " << endl;
    cValue = 4;
}

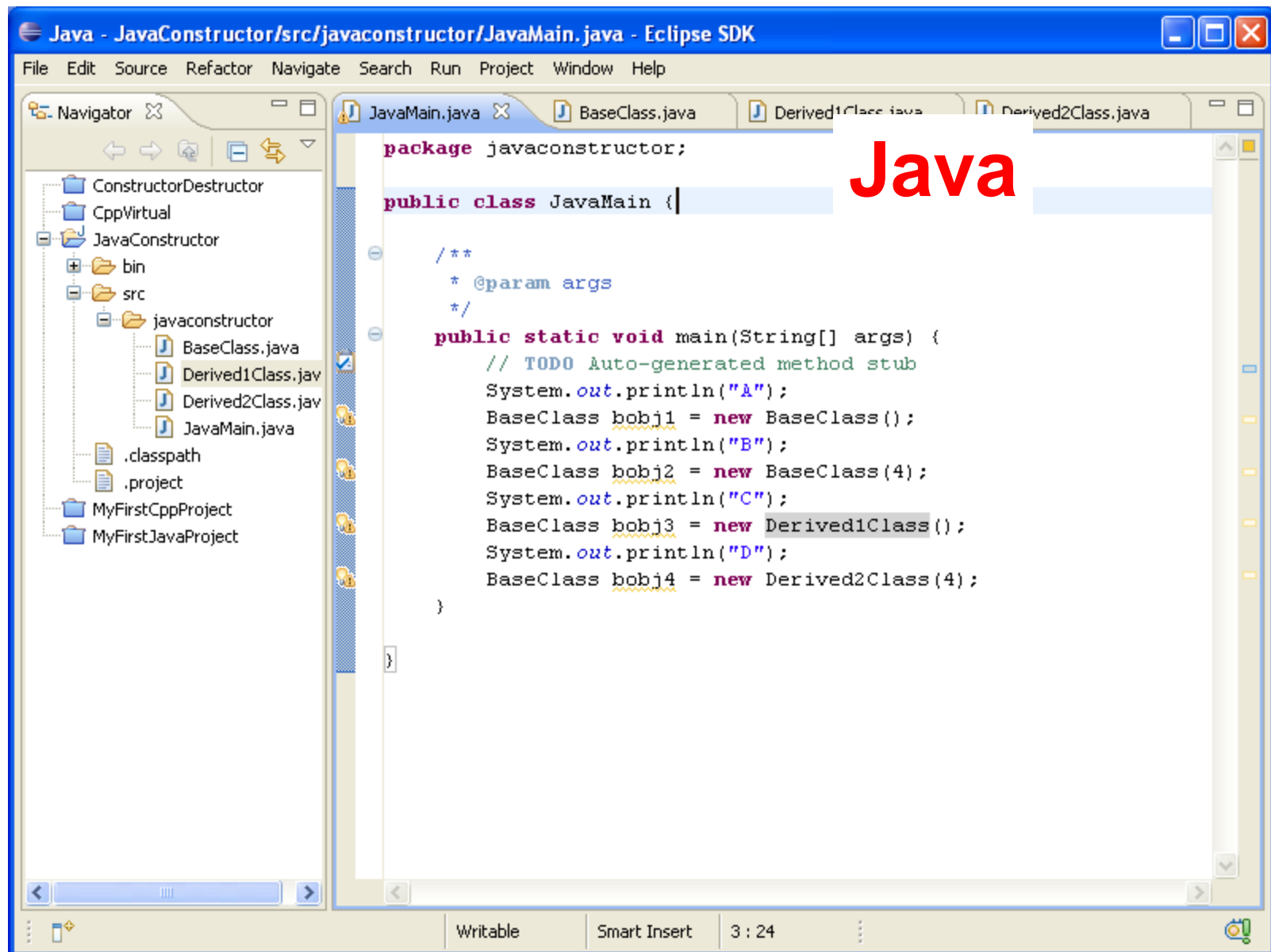
BaseClass::BaseClass(int x) {
    cout << "BaseClass::BaseClass(int) " << endl;
    cValue = x;
}

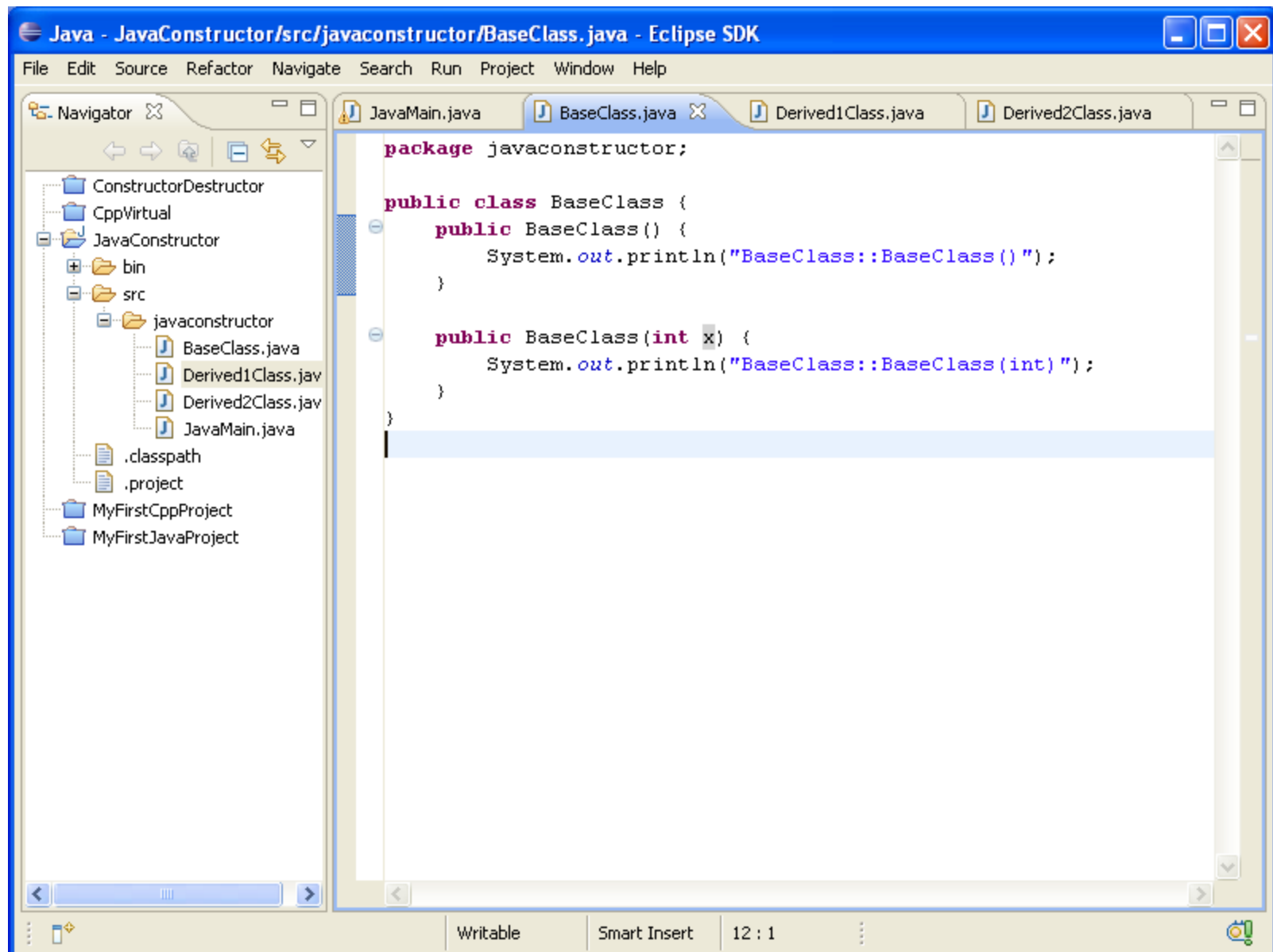
BaseClass::~BaseClass() {
    cout << "BaseClass::~BaseClass() " << endl;
}
```
- Console:** Shows the output of the C-Build process. It reports several errors related to the static member variable 'cValue':
 - Uninitialized member 'BaseClass::cValue' in the default constructor.
 - Assignment of read-only data-member 'BaseClass::cValue' in the parameterized constructor.The console output ends with 'make: *** [BaseClass.o] Error 1'.

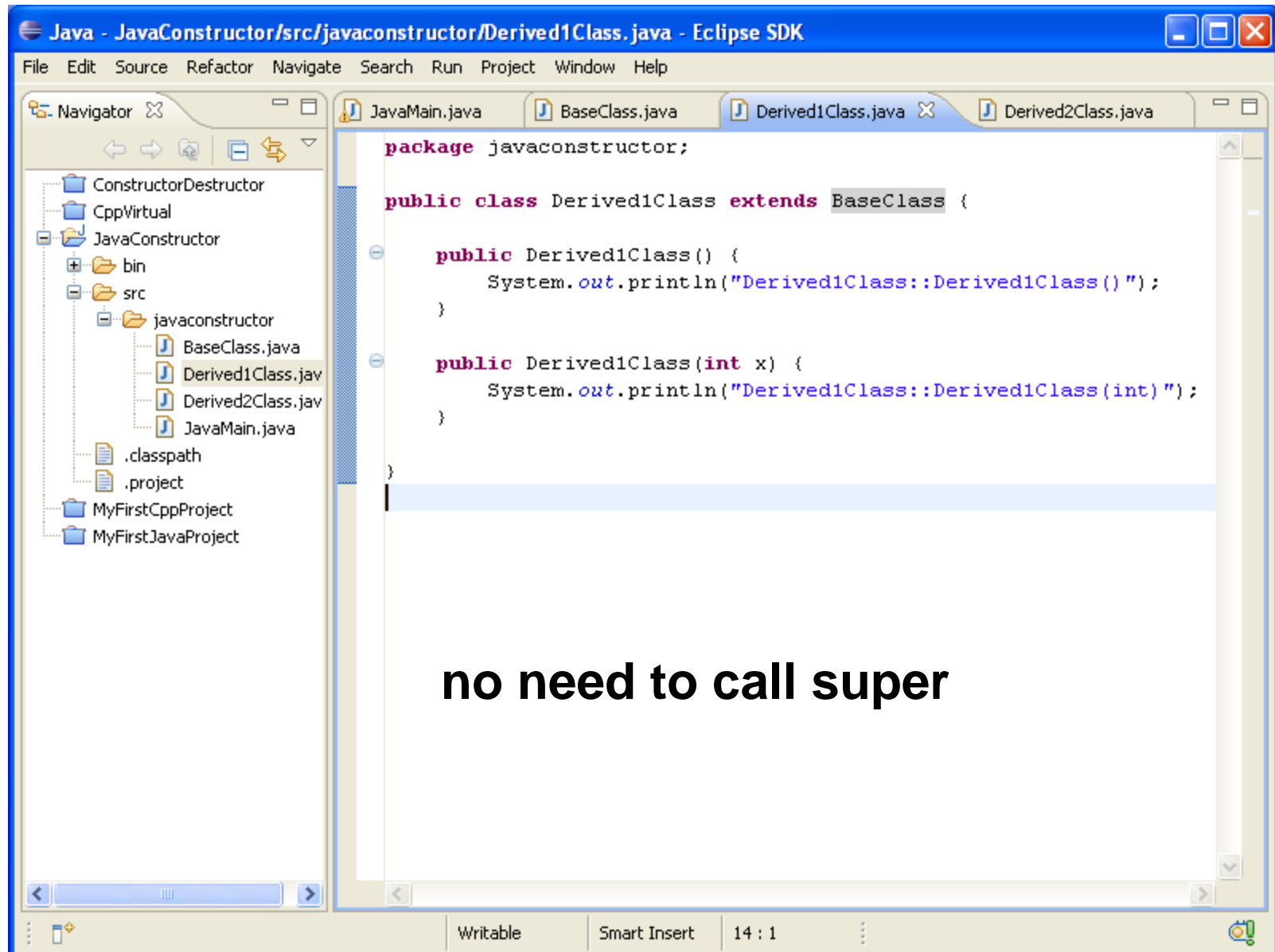
A yellow arrow points from the text 'const attribute cannot be initialized using assignments' to the line 'cValue = x;' in the code editor.

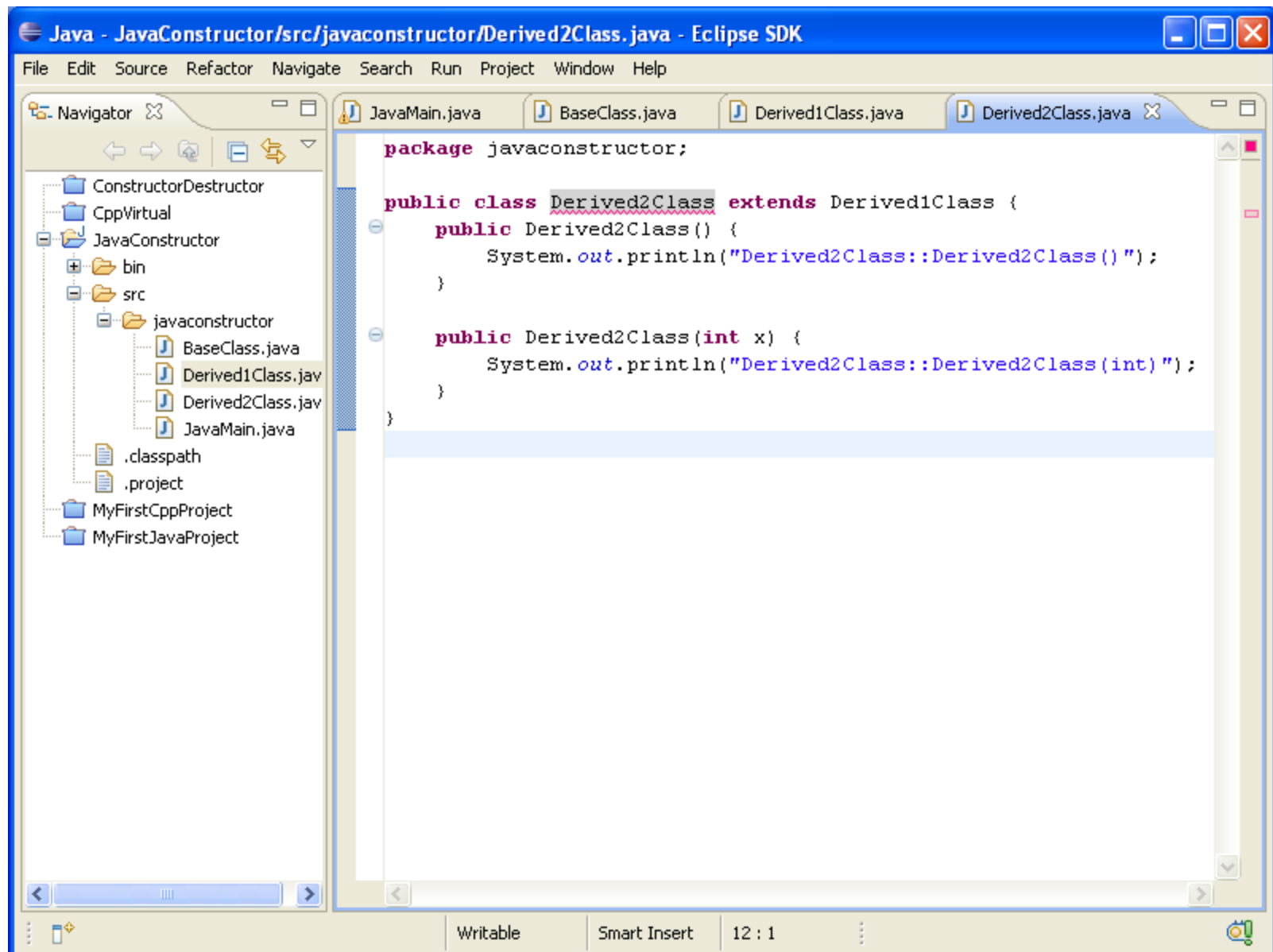
**const attribute cannot be
initialized using assignments**

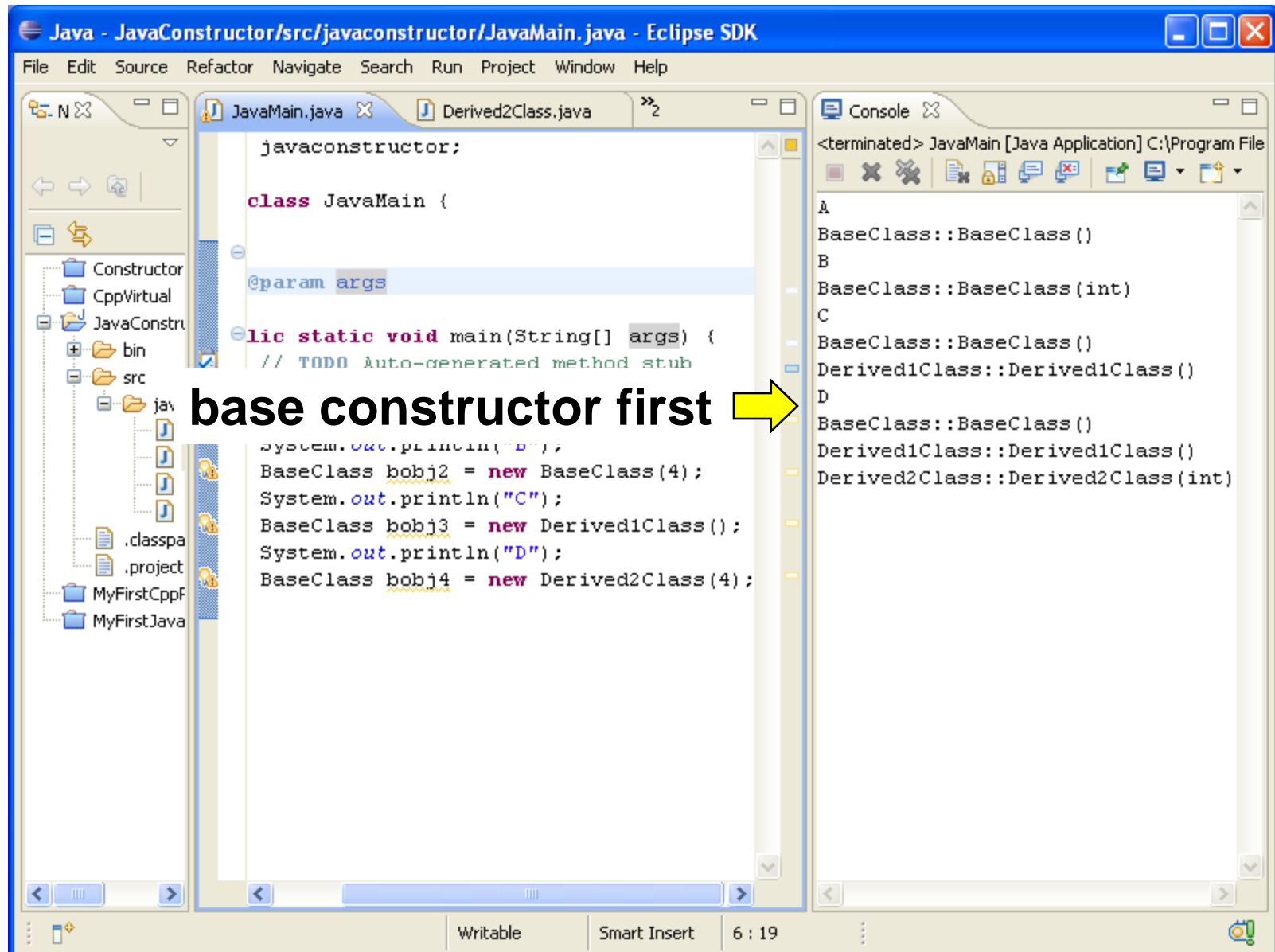


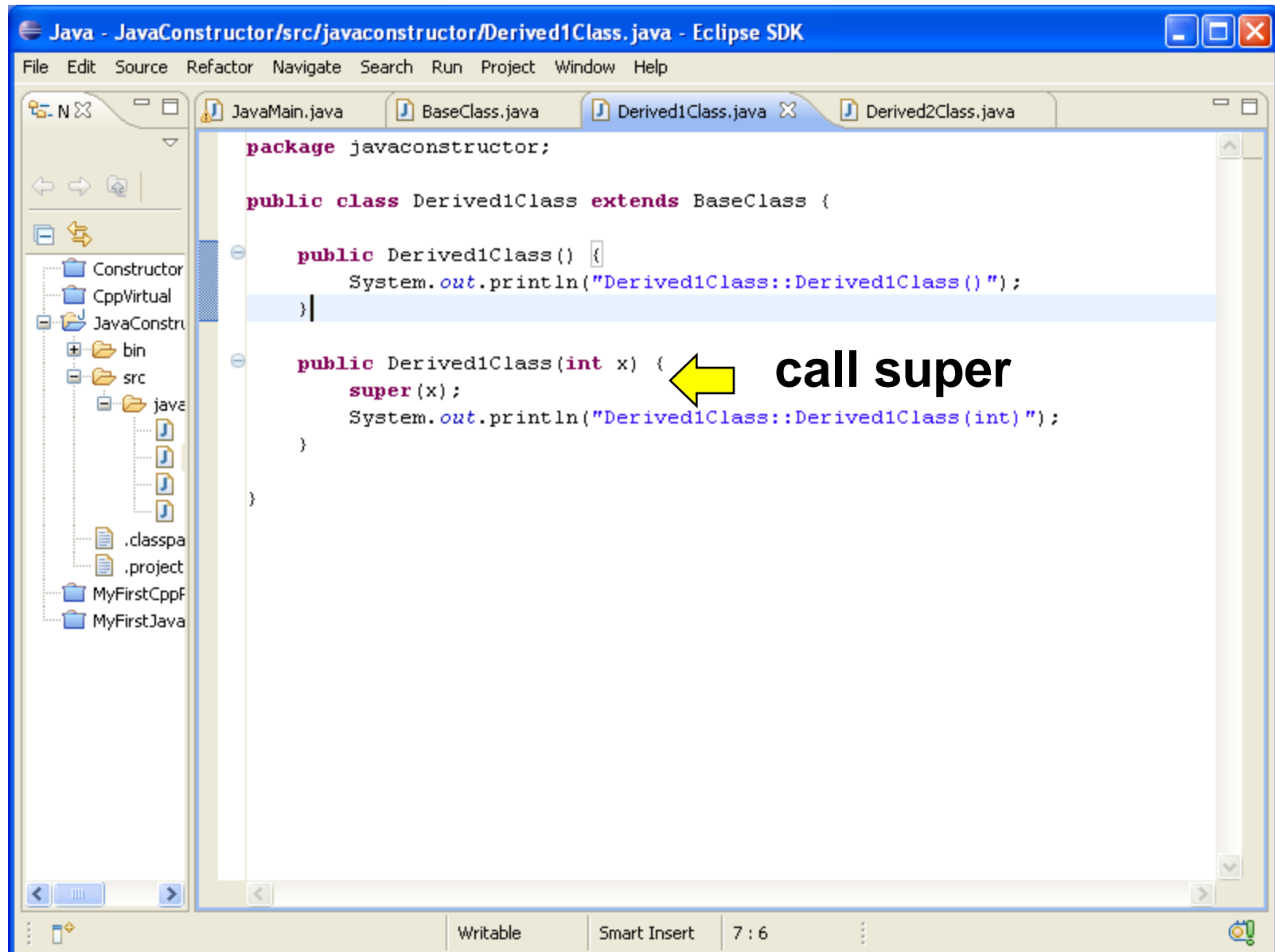


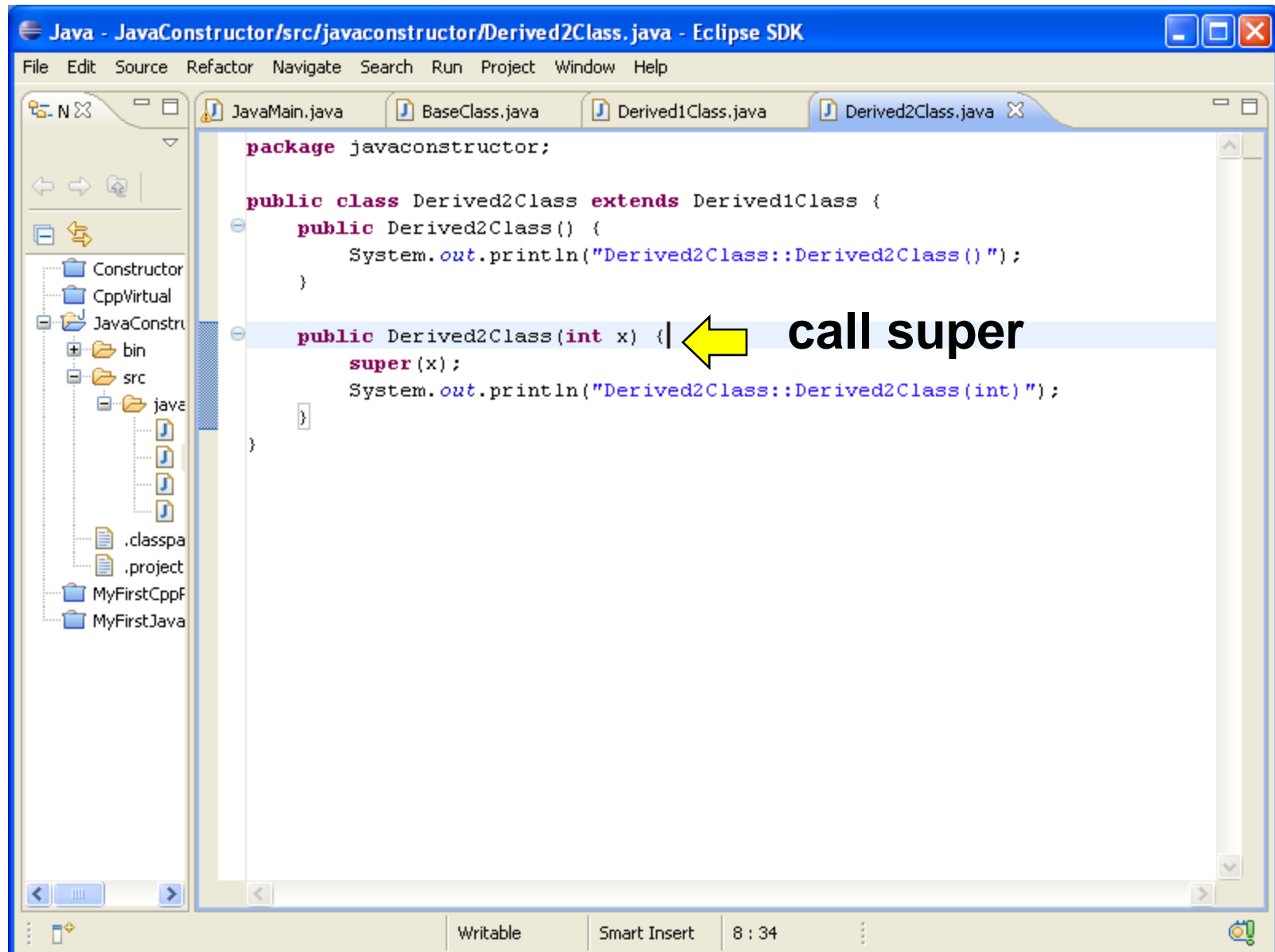


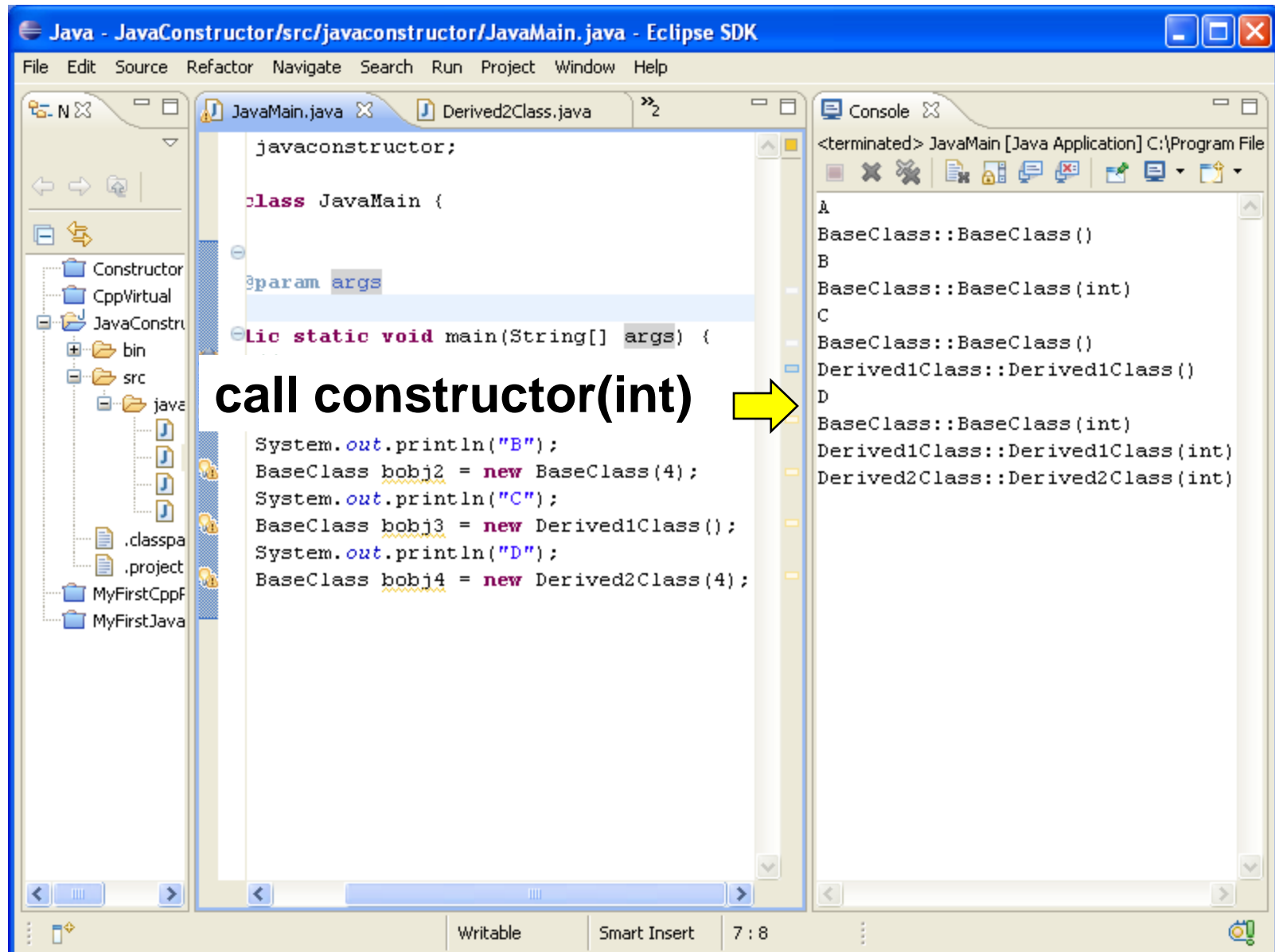


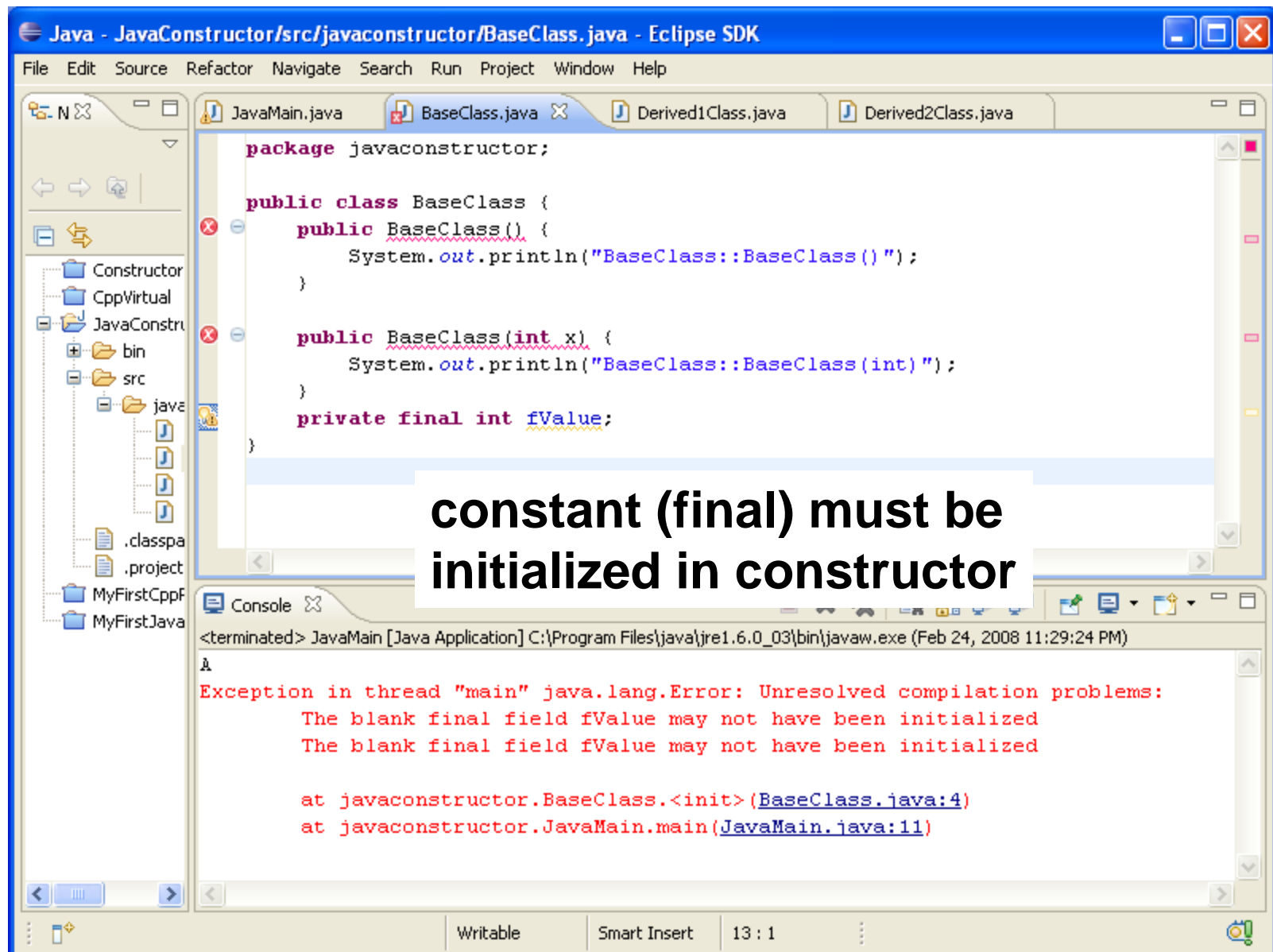


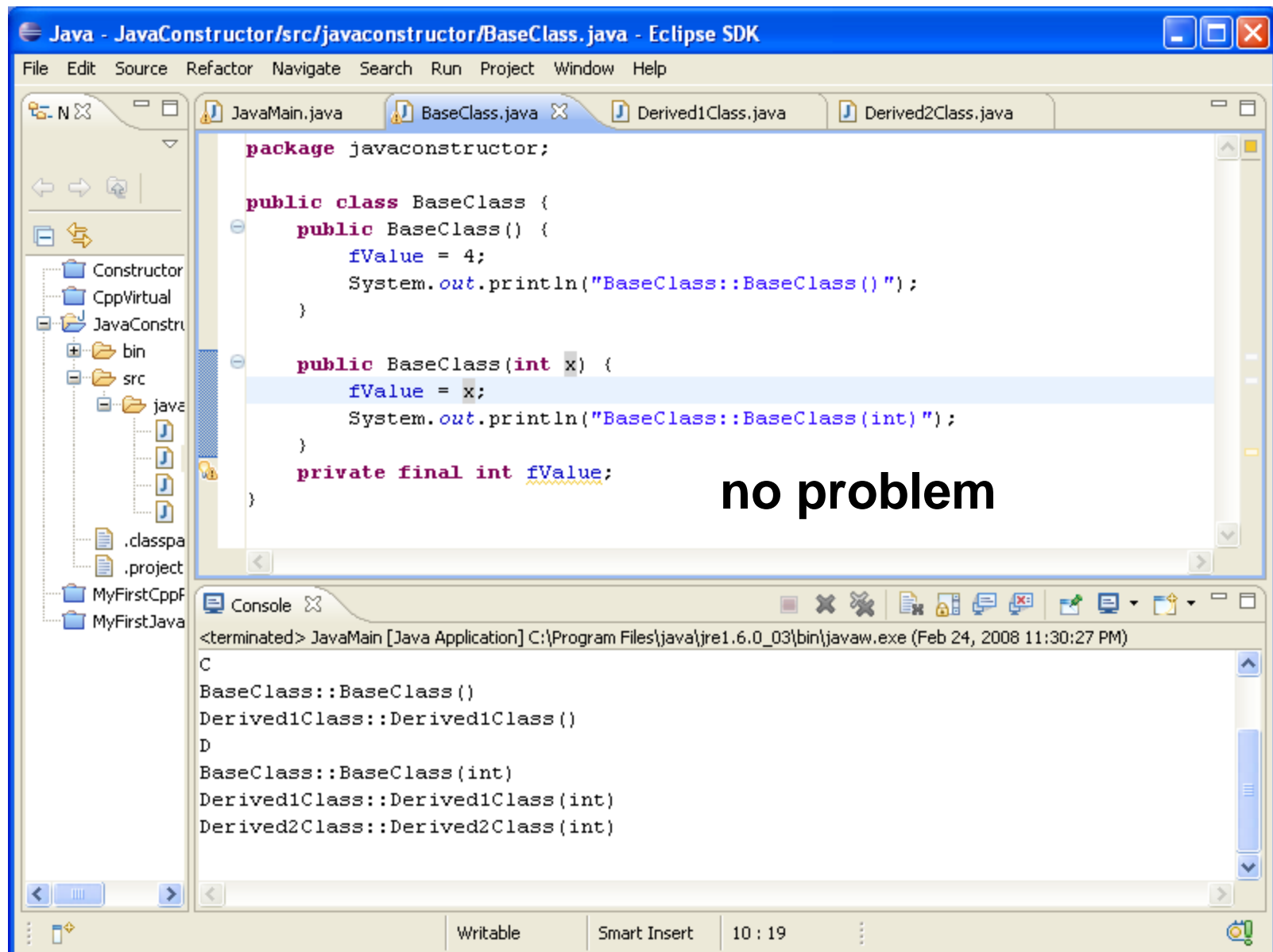


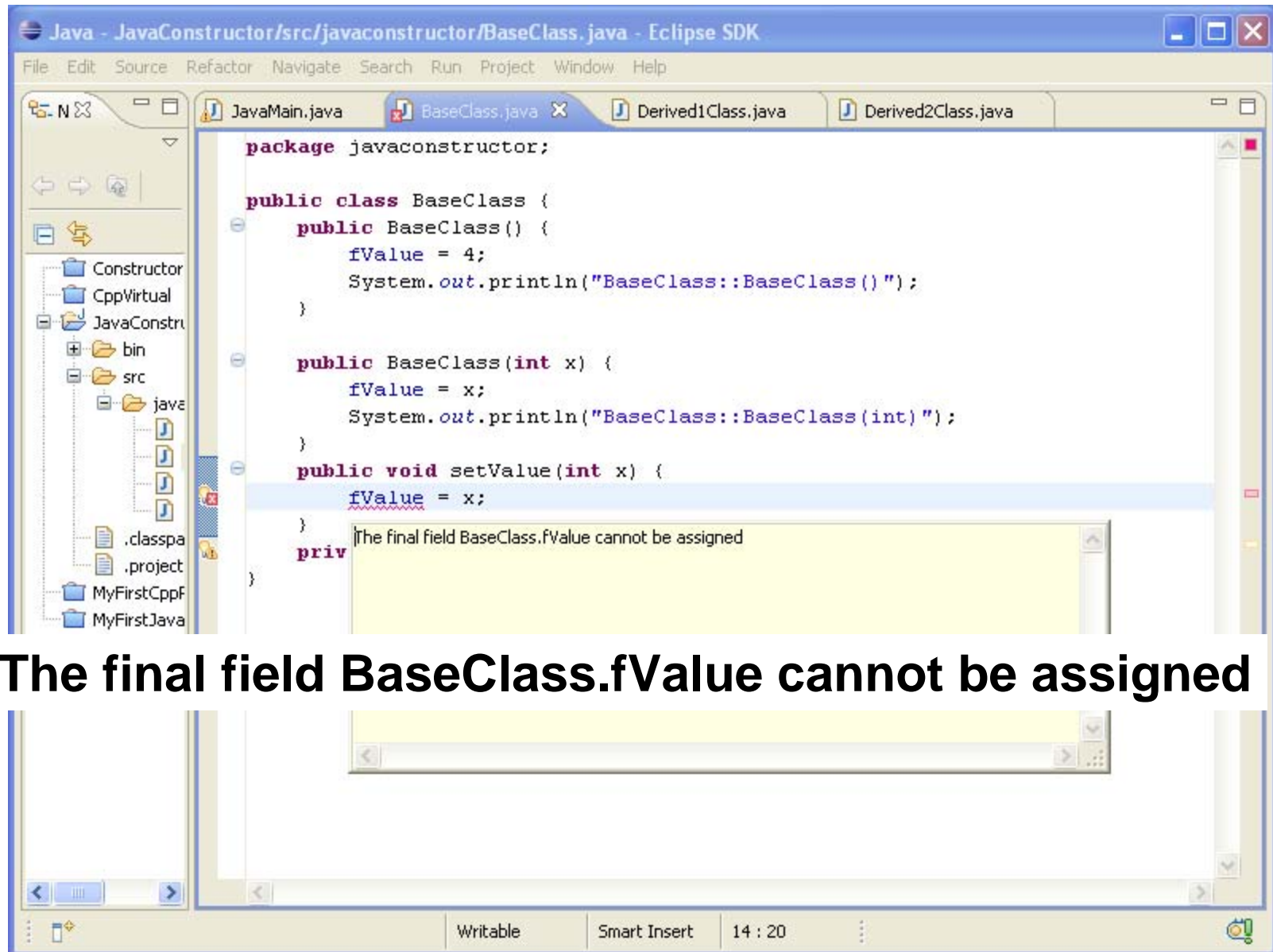












The final field BaseClass.fValue cannot be assigned

Self Test