

Class Diagrams in Enterprise Architect

Department of Software

ENTERPRISE ARCHITECT

a class Diagrams

Classes are represented by rectangles which either carry only the name of that class, or also the attribute and operations. The three compartments - Class name, Attributes, Operations - are each divided by a horizontal line. Class names usually start with a capital letter and are mostly substantive in singular (collection classes, among others, in plural where applicable).

The attributes of a class are noted with at least their names, and can contain additional data pertaining to their type, an initial value, attribute values and constraints. Methods are also noted with at least their name, as well as with possible parameters, their type and initial values, as well as possible attribute values and constraints.

Rectangle

- width: int
- height: int
- area: double

- + Rectangle(width: int, height: int)
- + distance(r: Rectangle): double



ENTERPRISE ARCHITECT

Note :

- class name in top of box
- write < interfaces > on top of interfaces' names
- attributes (optional)
- should include all fields of the object
- operations / methods (optional)
- may (get/set) methods

ENTERPRISE ARCHITECT

To create attribute :

Right click on class → attribute

Class Attributes: b

General | Detail | Constraints | Tagged Values

Name: b

Alias:

Type: int

Scope: Private

Stereotype:

Containment: Not Specified

Initial:

Notes:

☐ Derived ☐ Static
☐ Property ☐ Const

Attributes

Name	Type	Initial Value
a	int	
b	int	

New Copy Save Delete

Close Cancel Help

ENTERPRISE ARCHITECT

create attribute :

- attributes (fields, instance variables)
 - *visibility name : type [count] = default_value*
 - visibility:
 - + public
 - # protected
 - private
 - ~ package (default)

Rectangle	
- width: int	
- height: int	
<hr/>	
+ Rectangle(width: int, height: int)	
+ distance(r: Rectangle): double	

UML has the following types of **visibility**:

Private: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.

package (Default): The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.



ENTERPRISE ARCHITECT

Protected: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.

Public: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

ENTERPRISE ARCHITECT

Access Modifier	within class	within package	outside package by subclass only	outside package
--------------------	-----------------	-------------------	-------------------------------------	--------------------

Private	Y	N	N	N
---------	---	---	---	---

Default	Y	Y	N	N
---------	---	---	---	---

Protected	Y	Y	Y	N
-----------	---	---	---	---

Public	Y	Y	Y	Y
--------	---	---	---	---

ENTERPRISE ARCHITECT

add two number

- a: int
- b: int

To add initial value:

Right click on class → advance → override
attribute initializers

Override Attribute Initializers

Variable	n
Operator	=
Value	5
Note	

OK
Apply
Cancel
Help

add three number

- n = 5
- a: int
 - b: int

ENTERPRISE ARCHITECT

To create operation :
Right click on class → operation

Class Operation: add

General Behavior Pre Post Tagged Values

Name: add

Parameters: b: int, a: int [Edit Parameters](#)

Return Type: int [Advanced](#) ☐ Static



Scope: Public ☐ Abstract ☐ Const

Stereotype: ☐ Return Array ☐ Pure

Concurrency: Sequential ☐ Synchronized ☐ Is Query

Alias:

Notes: **B I U A** $\frac{1}{2}$ $\frac{1}{3}$ x^2 x_2

Operations   [New](#) [Copy](#) [Save](#) [Delete](#)

Name	Return Type	Parameters
add	int	(int, int)



[Close](#) [Cancel](#) [Help](#)

Parameters

Name: Type: boolean Default: <none>

Stereotype: Kind: in ☐ Fixed

Alias: ☐ Add new to end

  [New](#) [Save](#) [Delete](#) [Close](#)

Parameters

Name	Type	Default
b	int	
a	int	

ENTERPRISE ARCHITECT

Where a is private and integer number

b is private and integer number

add if function received two integer
number and return integer number(a+b)

add two number

- a: int
- b: int

+ add(int, int) : int

area

pi = 13,4

- x: int
- y: int

+ compute area(int, int) : int

ENTERPRISE ARCHITECT

Code Generation

To generate code :

Right click on class → generate code

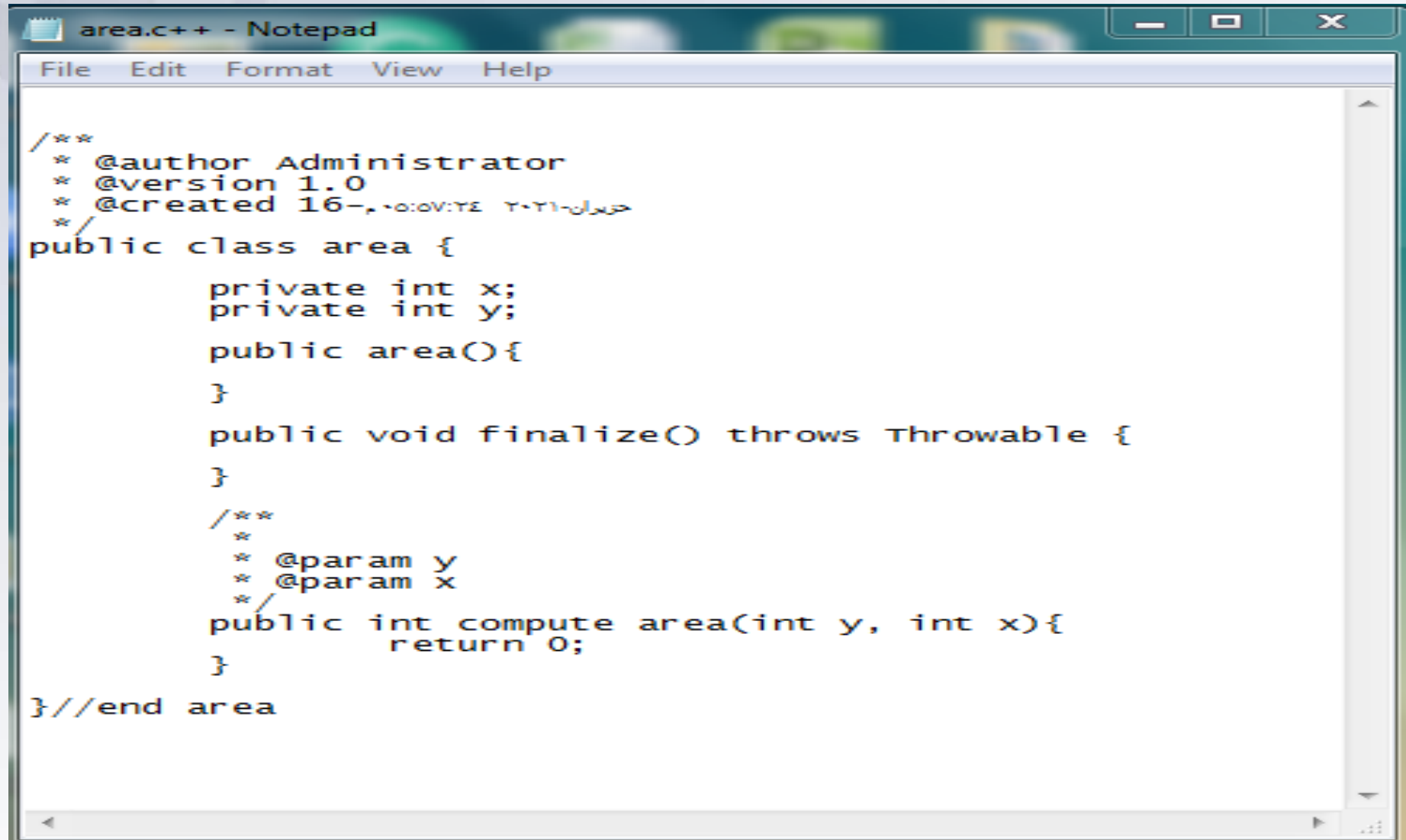
The screenshot shows the 'Generate Code' dialog box in Enterprise Architect. The dialog has a title bar with a close button (X). The main area is divided into several sections:

- Path:** A text field containing 'C:\Users\Administrator\Desktop\areba.java' and a blue button with three dots (browse).
- Target language:** A dropdown menu currently set to 'Java'.
- Details:** A text field containing 'area'.
- Import(s) / Header(s):** Two empty text areas with vertical scrollbars.

On the right side of the dialog, there is a vertical stack of buttons: 'Generate', 'Advanced', 'View', 'Save', 'Close', and 'Help'.

ENTERPRISE ARCHITECT

Code Generation



The image shows a Notepad window titled 'area.c++ - Notepad'. The window contains C++ code for a class named 'area'. The code includes a header comment with author, version, and creation date information. The class 'area' is defined with two private integer variables, 'x' and 'y'. It has a public constructor 'area()' and a public method 'finalize()' that throws a 'Throwable'. There is also a public method 'compute area(int y, int x)' that returns 0. The code is enclosed in a namespace or module declaration 'area'.

```
area.c++ - Notepad
File Edit Format View Help

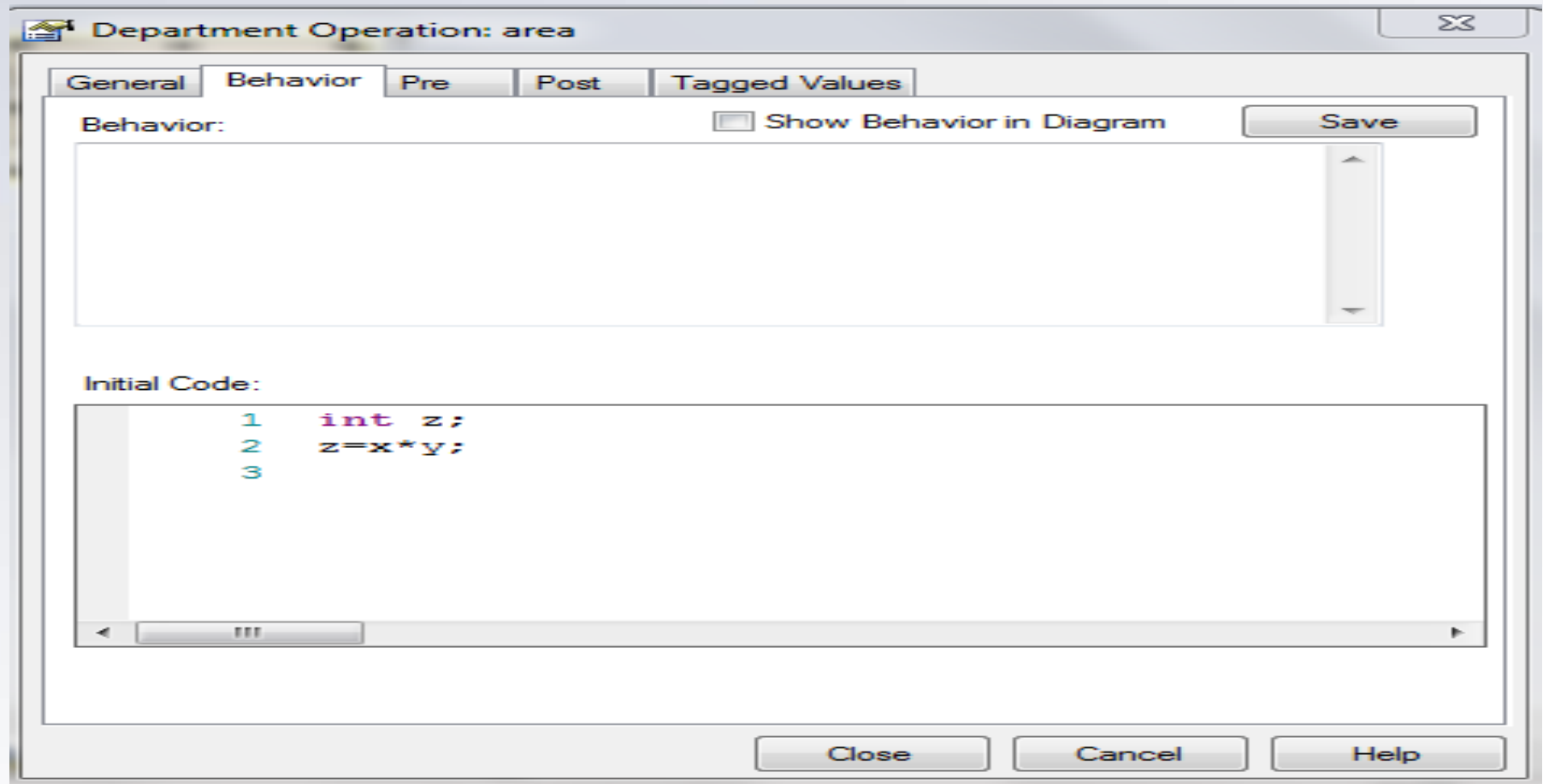
/**
 * @author Administrator
 * @version 1.0
 * @created 16-٠٥:٥٧:٢٤ ٢٠٢١-٠٥-١٦
 */
public class area {
    private int x;
    private int y;
    public area(){
    }
    public void finalize() throws Throwable {
    }
    /**
     *
     * @param y
     * @param x
     */
    public int compute area(int y, int x){
        return 0;
    }
} //end area
```

ENTERPRISE ARCHITECT

Code Generation

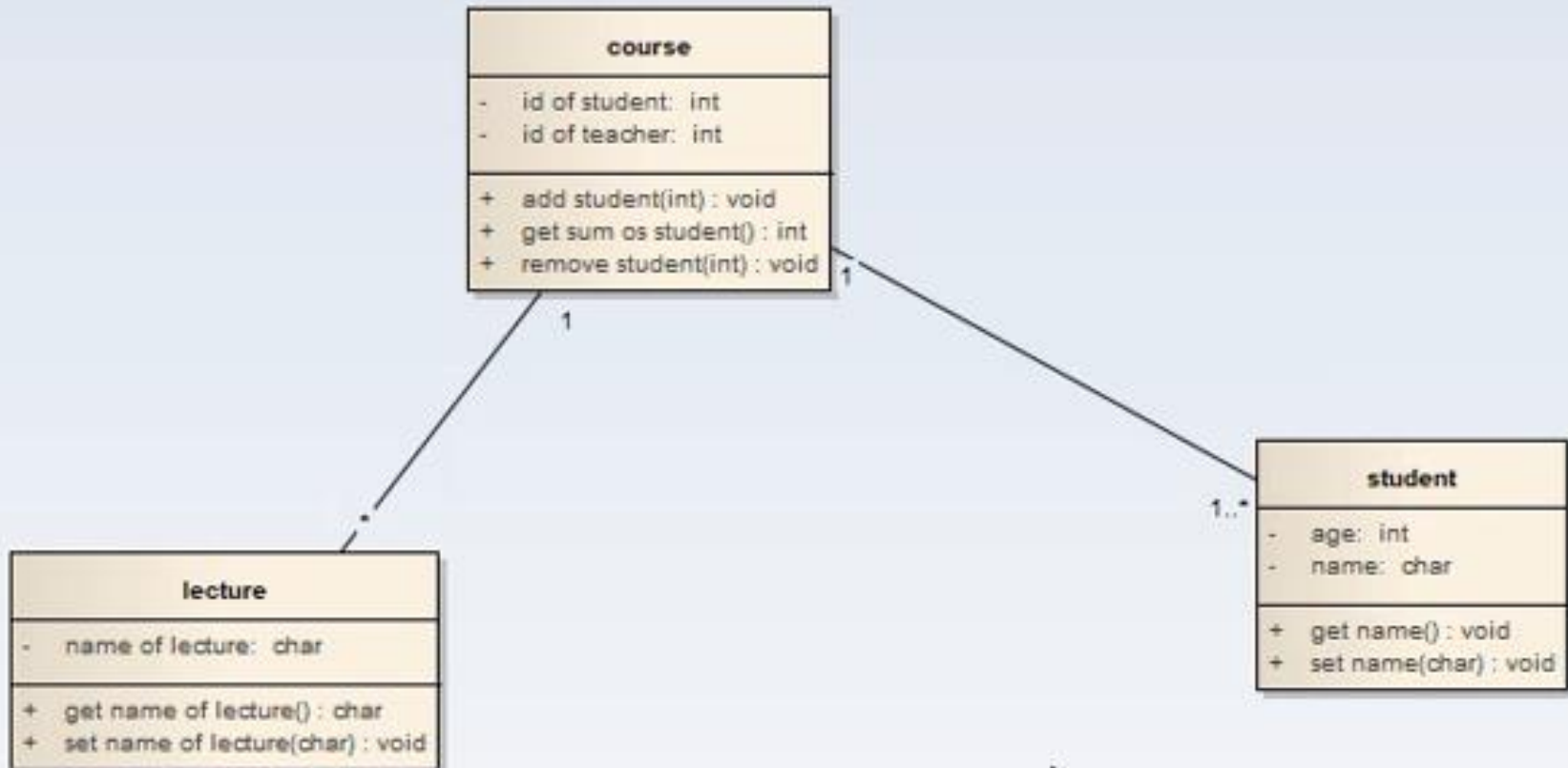
To add code in method :

Select operation → behavior



ENTERPRISE ARCHITECT

example



Example

Design class Diagram for Car that contain:

- 1-car model
- 2-car
- 3-Engine
- 4-GearBox
- 6-Door
- 7- Brake
- 8- tire
- 9-wheel

and create attribute and operation for each class



Thank You