

Unified Modeling Language (UML)

static members are underlined.

abstract elements are italicized.

Access Levels	
+	public
-	private
#	protected
~	default

fields are given as access level followed by variable name, a :, then type

Example: -x:int

methods are given as access level followed by name(params), a :, then return type


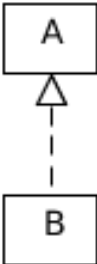
Example: #foo(a:int,b:int):int

constructor is identified via the «constructor» stereotype

Example: +«constructor»MyClass(x:int,y:int)

interfaces are given the «interface» annotation

enumerations are given the «enumeration» annotation

Relationship	Diagram	Meaning
Generalization		A generalizes B B is a subclass of A In Java this is B extends A
Realization		B realizes the interface defined in A In Java this is B implements A

Unified Modeling Language (UML)

```
public class Example {  
    private int x;  
    protected int y;  
    public int z;  
    public Example() { ... }  
    public String toString() { ... }  
    private void foo(int x) { ... }  
    protected int bar(int y, int z) { ... }  
}
```

Example
-x:int #y:int +z:int
+«constructor»Example() +toString():String -foo(x:int) #bar(y:int,z:int):int

```
public abstract class Example2 {  
    public static final double PI = 3.14;  
    public abstract void foo() { ... }  
    protected void bar() { ... }  
}
```

Example2
<u>+PI:double</u>
+foo() #bar()

```
public interface FooListener {  
    public void foo();  
}
```

«interface» FooListener
+foo()