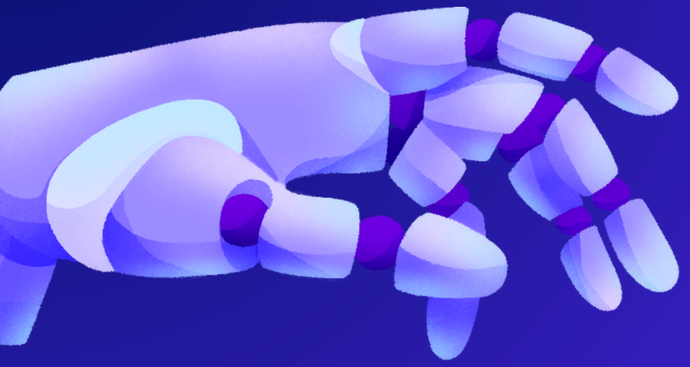
UNIVERSIDAD DE LAS FUERZAS ARMADAS "ESPE"

OBJECT-ORIENTED PROGRAMMING

NRC: 1940
DATE: 05/11/2024

1922
ECUADOR
ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA



Correct use of identifiers

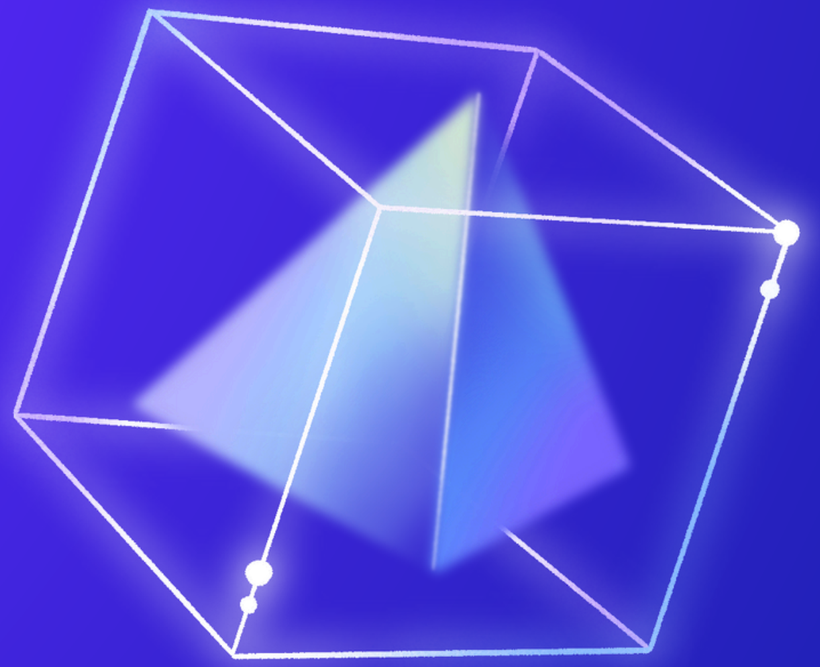
Group Members:

-CHASIPANTA IAN

-CALVOPIÑA DAVID

-CHASIPANTA SAUL

-ROJAS JONATHAN



Correct use of identifiers

In programming, identifiers are the names we assign to variables, functions, classes, methods, etc., to refer to them in the code. Using identifiers correctly is essential to creating code that is clear, easy to understand, and maintain.

Access modifiers

In object-oriented programming, access modifiers are keywords that control the visibility of a class's properties and methods from other classes or outside of it. These modifiers help define access levels to elements of a class, promoting encapsulation and data protection.

The main access modifiers and the implementation of classes in Java

Java has three main access modifiers:

-Public (public): Allows access to attributes or methods from anywhere in the program.

-Private (private): Restricts access only to the class itself. Private attributes and methods cannot be accessed or modified directly from outside the class.

-Protected (protected): Allows access from the same class, subclasses and classes within the same package.

IDENTIFIERS

Class implementation

The class implementation tries to adapt to people's way of thinking and adjust to their way of analyzing.

This stage comprises two components which are the declaration and the body of the class.

ClassDeclaration

{

ClassBody

}

HOW TO...

The class declaration must contain the keyword class and the name of the class that is being defined, for example class imaginarynumber {}

Within the class declaration you can

- Declare what the class is
- List the interfaces implemented by the class
- Declare if the class is public, private

SUPERCLASS

Declare the superclass of the class
If a superclass is not specified, it is assumed to be the object class. To explicitly specify the super class of a class, you must put the extends keyword plus the name of the superclass between the name of the class that has been created classClassName extends SuperClassName{
}.
A subclass inherits the variables and methods of the superclass

LIST THE INTERFACES IMPLEMENTED BY THE CLASS

When declaring a class you can specify what interface.
An interface declares a set of methods and constants without specifying their implementation for any method. When a class requires the implementation of an interface, it must provide the implementation for all methods declared in the interface.

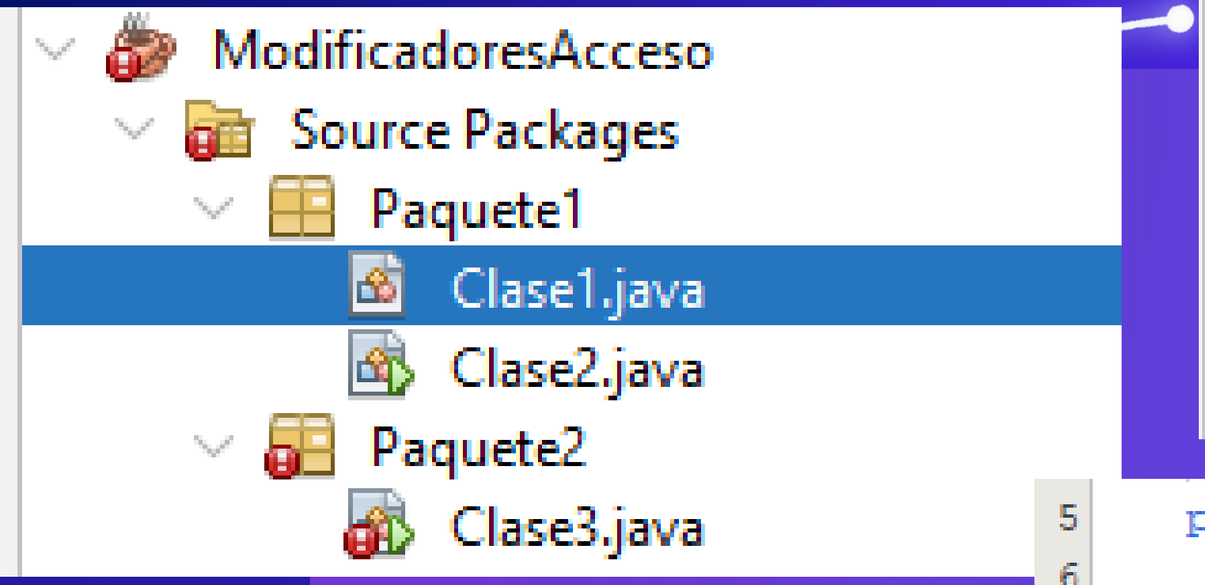
To declare that a class implements an interface, the keyword implements must be entered followed by the list of interfaces implemented by the class, delimited by commas, for example an interface called Arithmetic that defines the methods called addition (), subtraction (), etc. The Imaginary Number class can declare that it implements the Arithmetic interface like this.class Imaginary Number extends Number implements Arithmetic.

DECLARE

Declare if the class is public, abstract or final
The public modifier declares that the class can be used by objects outside the current package. When you use the word public you must ensure that it is the first item in the declaration. public class ImaginaryNumber extends Number implements Arithmetic {
}.

ACCESS MODIFIER

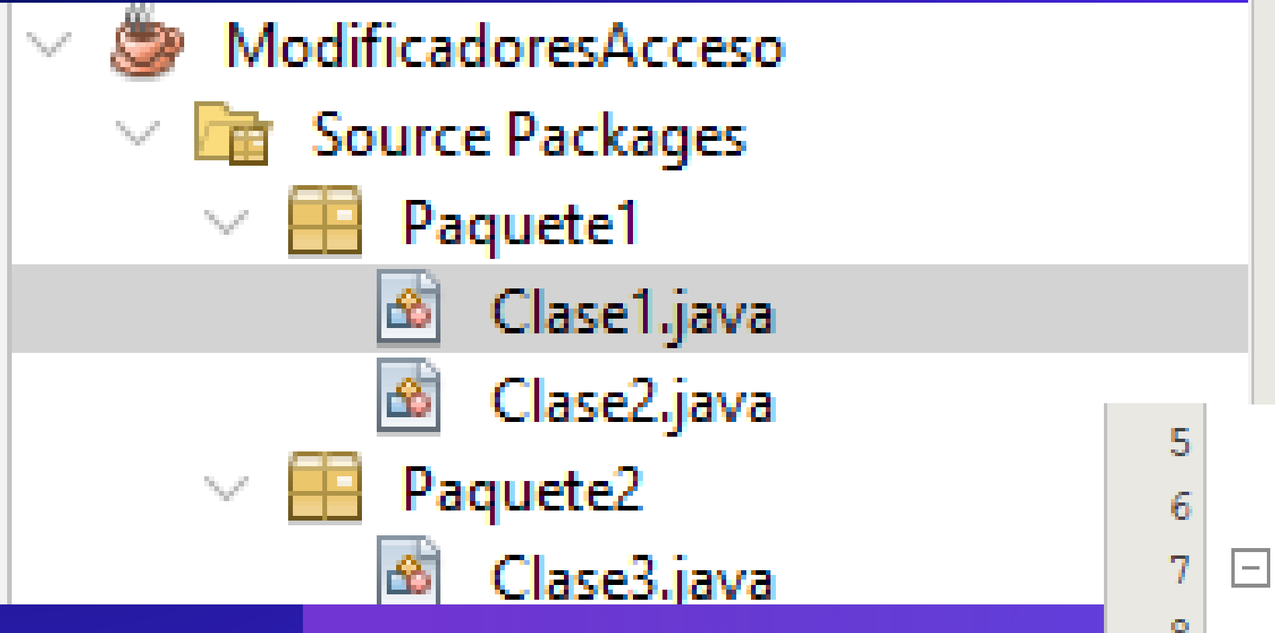
MODIFICADOR	CLASE	PACKAGE	SUBCLASE	TODOS
Public	Sí	Sí	Sí	Sí
Protected	Sí	Sí	Sí	No
Private	Sí	No	No	No
Por defecto	Sí	Sí	No	No




```
5   package Paquete1;
6
7   public class Clase2 {
8       public static void main(String[] args){
9           Clase1 obj1= new Clase1();
10
11          obj1.atributo1 = 15;
12      }
```

```
5   package Paquete1;
6
7   public class Clase1 {
8       int atributo1;
9
10
11  }
12
```

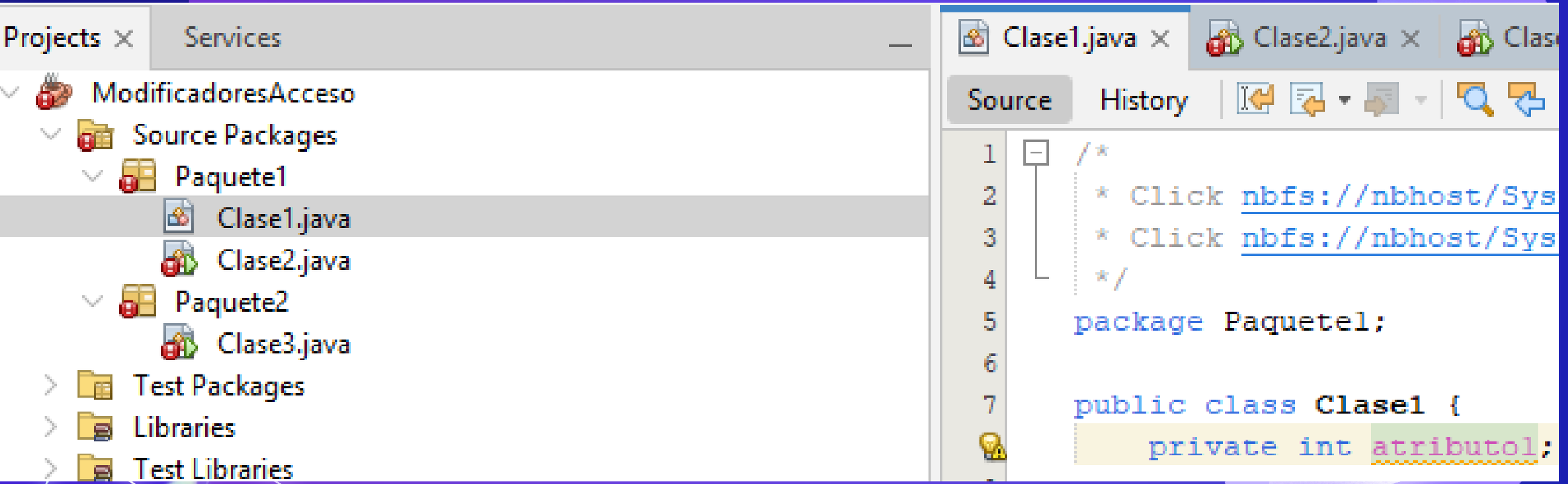
```
5   package Paquete2;
6
7   import Paquete1.Clase1;
8
9   public class Clase3 {
10      public static void main(String[] args){
11          Clase1 obj1 = new Clase1();
12          atributo1 is not public in Clase1; cannot be accessed from outside package
13          ----
14          (Alt-Enter shows hints)
            obj1.atributo1 = 15;
16
17      }
18
19  }
```

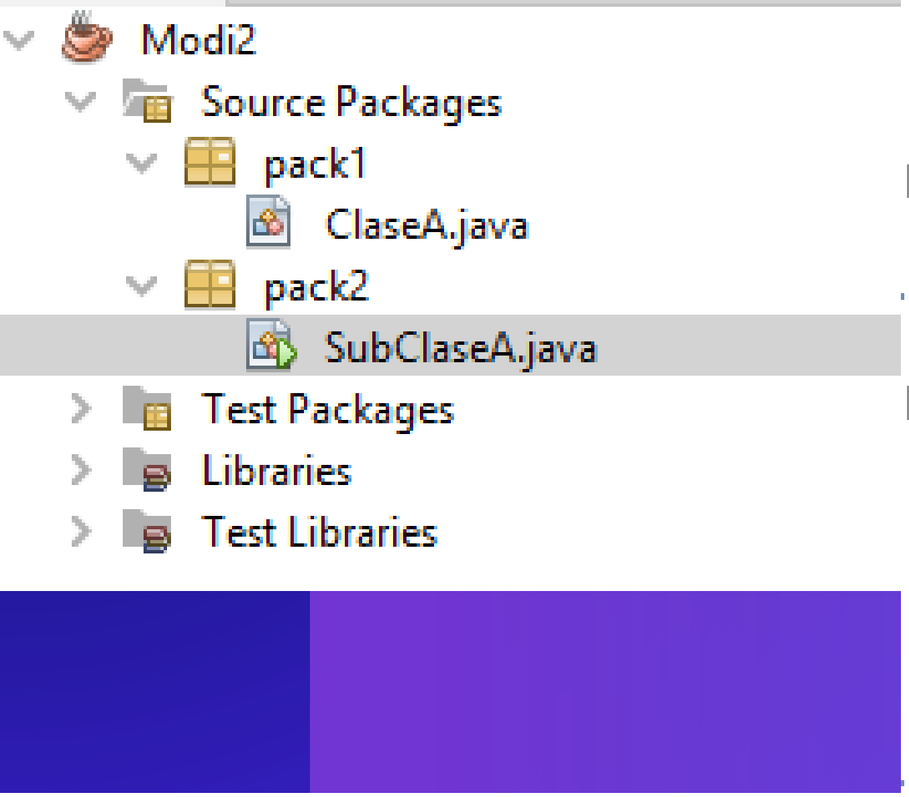



```java
package Paquete1;

public class Clase1 {
    public int atributo1;


}
```

```java
package Paquete1;

public class Clase2 {
    public static void main(String[] args){
        Clase1 obj1= new Clase1();

        obj1.atributo1 = 15;
```

```java
package Paquete2;

import Paquete1.Clase1;

public class Clase3 {
    public static void main(String[] args){
        Clase1 obj1 = new Clase1();



        obj1.atributo1 = 15;

    }

}
```


Projects
Services
ModificadoresAcceso
Source Packages
Paquete1
Clase1.java
Clase2.java
Paquete2
Clase3.java
Test Packages
Libraries
Test Libraries
Clase1.java
Clase2.java
Source
History
1 /*
2 * Click nbfs://nbhost/Sys
3 * Click nbfs://nbhost/Sys
4 */
5 package Paquete1;
6
7 public class Clase1 {
private int atributo1;




```java
package pack2;

import pack1.ClaseA; // Importar la clase del otro paquete

public class SubClaseA extends ClaseA { // Extiende ClaseA

    public static void main(String[] args) {
        // Creamos un objeto de SubClaseA
        SubClaseA asub = new SubClaseA();

        // Accedemos al atributo protegido de la clase base
        System.out.println( x: asub.MensajeProtegido);
    }
}
```

```java
package pack1;

public class ClaseA {
    protected String MensajeProtegido = "Esto está protegido";
}
```

Output - Modi2 (run)

```
run:
Esto está protegido
BUILD SUCCESSFUL (total time: 0 seconds)
```

THANKS