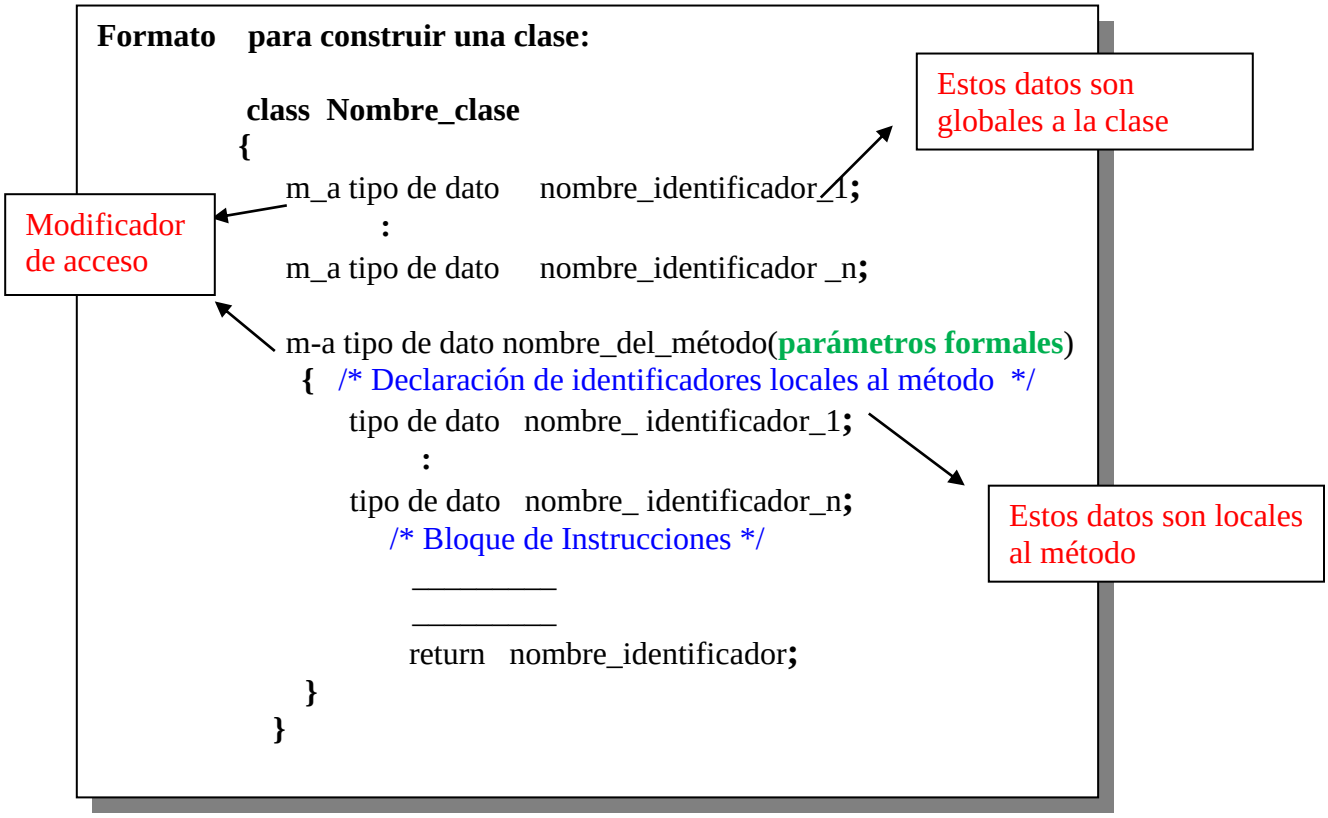
2.4 Construcción de clases y objetos en Java

2.4.1. Definición de una clase

2.4.1.1 Miembros de una clase:



Los parámetros formales:

- son valores que se reciben en el método en forma de **variables u objetos**
- deben ser declarados con su tipo de dato correspondiente y separados por coma.,
- son locales al método, lo cual significa que serán reconocidos sólo en el método donde están declarados. (Esto se explicó en el curso de Desarrollo de Software I).

Paso de parámetros por valor: Todos los tipos de datos primitivos (int, long, float, double, boolean), se pasan en los métodos por valor. En otras palabras, sus valores se copian en nuevas posiciones, que se pasan al método; como consecuencia de esto, si un argumento se cambia dentro de un método no se cambiará en el programa llamador original. (Esto se explicó en el curso de Desarrollo de Software I).

Paso de parámetros por referencia: Los objetos, los arreglos y cadenas pasan por referencia. El paso de un objeto significa que la referencia del objeto se pasa a un parámetro formal. Cualquier cambio al objeto local que suceda dentro del cuerpo del método afectará al objeto original que se pasa como argumento. (Esto lo explicaré en el capítulo de arreglos).

2.4.3 Modificadores de acceso.

Con el fin de mantener la característica de **ocultar** en POO, se debe considerar cómo se accede a los miembros de la clase. Normalmente, es una buena práctica restringir el acceso a los datos y métodos de una clase que se utiliza para definir el objeto. Esta práctica de limitación del acceso a cierta información interna se llama **ocultamiento.**

A continuación, se presenta un cuadro completo de los modificadores de acceso en Java, donde se pueden especificar y cómo funcionan en la clase, objeto, paquete, subclase.
Por el momento veremos solamente su efecto en una clase y objeto.

Modificadores	Clase	Métodos	Datos	Funcionamiento
Por defecto	**si**	**si**	**si**	• Un ***dato*** o ***método*** es visible en esta **clase** • **Un *método* o *dato* es visible en el objeto** • La ***clase*** es visible en este **paquete**
public	**si**	**si**	**si**	• Un ***dato*** o ***método*** es visible en esta **clase** **Un *método* o *dato* es visible en el objeto** • Una ***clase*** es visible a todos los programas de cualquier **paquete**
private	**no**	**si**	**si**	• Un ***método*** o ***dato*** es visible en esta **clase y no visible en la subclase** • **Un *método* o *dato* no es visible en el objeto**
protected	**no**	**si**	**si**	• Un ***método*** o ***dato*** es visible en esta **clase** y en las **subclases** • **Un *método* o *dato* es visible en el objeto**

Cuando especificamos modificadores de acceso (public o por defecto) a una clase su funcionamiento es de la siguiente manera:

public

```
/*La clase es accesible dentro de cualquier
paquete, todos los paquetes la pueden usar */

public class BufferedReader        extends
Reader
{
   . . .
}
```

por defecto

```
/*La clase es accesible dentro del paquete
donde esta la clase */

      class  Ejemplo
      {


       }
```

Cuando creamos un objeto, lo hacemos instanciándolo a una clase.
A continuación, se presentan unos ejemplos de los modificadores de acceso en Java y como funcionan en el objeto:

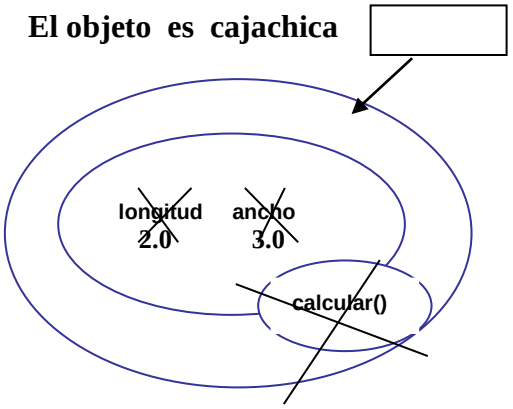
El símbolo ⊗ significa que no es visible para el objeto o en otras palabras que el objeto no lo puede usar.

Ejemplo de acceso privado:

```
class    Rectangulo
{   private   int  longitud;
    private   int ancho ;
    private   calcular( )
    {

    }
}

   .    .    .
```




```
cajachica.longitud = 80.45;  /* ERROR…longitud no es visible para el objeto  */
cajachica.calcular( );       /* ERROR…calcular no es visible para el objeto caja */
```

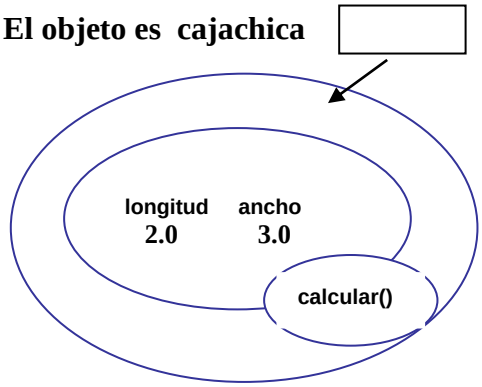
Ejemplo de acceso por defecto, público o protegido:

```
class    Rectangulo
{                int longitud;
     protected int ancho;
     public       void calcular( )
     {

     }
}


   .    .    .

   cajachica.calcular( );
```



Ver ejemplo de modificadores de acceso

2.4.4 Declaración y Creación de un Objeto.

Un objeto es una variable por referencia, este tiene la capacidad de guardar una dirección de memoria.
Una vez definida la clase se pueden crear muchas instancias de una clase.
Todos los objetos deben ser declarados antes de ser utilizados.

Existen 2 formatos para los objetos en Java:

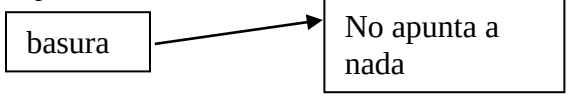
Formato # 1 permite declarar y después crear el objeto:

```
Nombre_clase     nombre_objeto_1 ;   //declara el objeto y contiene basura

nombre_objeto_1 =  new  Constructor (lista de párametros ) ;   //crea el objeto
```

Ejemplo:

```
//declara el objeto
Rectangulo   cajachica;
```




```
//crea el objeto
cajachica = new Rectangulo( );
```

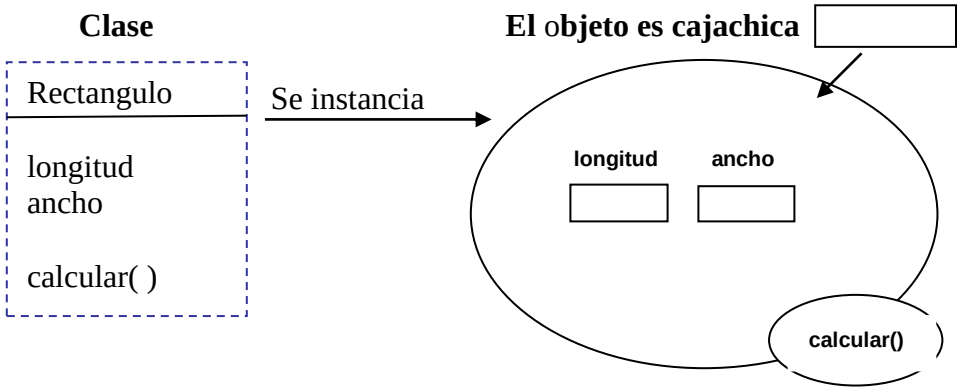
Formato # 2 permite declarar y crear el objeto:

```
Nombre_clase   nombre_objeto_1  =  new  constructor( ) ;//declara y crea el objeto
```

Es el constructor

Ejemplo :

Rectangulo cajachica = new Rectangulo ( ) ;

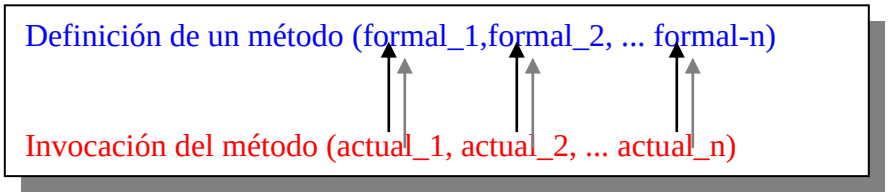


2.4.5 Acceso a datos y métodos

Después que se ha creado un objeto, se puede acceder a sus miembros con el punto (**.**) como se muestra en el formato siguiente:

nombre_objeto.nombre_del_método ([lista parámetros actuales])

Donde **lista de parámetros actuales** pueden ser variables, objetos y/o constantes, que serán enviados al método llamado. Los argumentos deben corresponder en tipo, orden y cantidad con la lista de parámetros del método.



Correspondencia de parámetros actuales y formales

(Esto se explicó en el curso de Desarrollo de Software I).

2.3.1.2.3. Métodos Especiales (constructor y destructor)

Constructor: es un método que sirve para construir un nuevo objeto y asignar valores iniciales a sus miembros datos. Al no existir un constructor, el compilador genera uno automáticamente o por defecto.

Los constructores se caracterizan por:

-tener el mismo nombre de la clase que se usó para instanciar el objeto
-no devuelven valores
- puede tener parámetros formales como cualquier método
- puede declarar más de un constructor
- no deberán desarrollarse para otra tarea que no sea de inicializar un objeto

Ejemplo de un constructor declarado por el programador:

```
clase  Rectangulo
{     private float longitud ;
      Private float ancho;
      Rectangulo(float lon, float a)
      { longitud = lon ;
        ancho = a;
       }
            :
}
Rectangulo  Cajachica  = new  Rectangulo (2.0, 3.0) ; //declaración y creación
                                                       del objeto
```

Destructor: es un método que libera o limpia el almacenamiento asignado a los objetos cuando se crean. Toda clase tiene el método constructor por defecto.