

# CAPÍTULO 1

/ /

## » Declaraciones y Control de Acceso Java Refresher

Un programa Java es principalmente una colección de objetos que se comunican con otros objetos invocando los métodos de cada uno. Cada objeto es de un cierto tipo, y ese tipo está definido por una clase o una interfaz. La mayoría de los programas Java utilizan una colección de objetos de diferentes tipos.

→ **Clase:** Un modelo que describe el tipo de estado y comportamiento que los objetos de su tipo soportan.

→ **Objeto:** Cuando la máquina virtual de Java (JVM) encuentra nuevas palabras, esta las usa para la apropiación de la clase para crear un objeto que es una instancia de esa clase.

→ **Estado (Instancia de Variables):** Cada objeto (instancia de una clase) que tendrá su propio conjunto único de instancia de variables como definido en la clase.

→ **Métodos:** Cuando un programador crea clases, se crean métodos para la clase. Los métodos son donde la lógica de las clases se guardan. Son donde los algoritmos son ejecutados y manipulan los datos.

## » Identificadores y Palabras Clave

Todos los componentes de Java, justamente hablando acerca de clases, variables y métodos necesitan nombres. En Java esos nombres son llamados identificadores.

## » Herencia

Definido como una clase se reusa en otra clase. En Java, uno puede definir en general (más abstracto) superclases y estos se extienden con subclases más específicas. La superclase no sabe nada de las clases que la heredan, pero todas las subclases que heredan las superclases deben declarar explícitamente la relación de herencia.

Una subclase que hereda desde una superclase es automáticamente dado accesible a variables de instancia y métodos definidos por la superclase, pero también es libre de anular métodos de las superclases para definir más comportamientos en específico.

## » Interfaces

Un poderoso compañero para la herencia es el uso de las interfaces. Las interfaces son como una superclase abstracta al 100% que define los métodos que una subclase debe admitir, pero no como ellos deben ser soportados. En otras palabras, una interfaz Animal podría declarar que todas las clases de implementación de animales tienen un método comer(), pero la interfaz

Animal no proporciona alguna lógica para el método comer().

### » Encuentro Otras Clases

Para cada clase debe tener enfocada su responsabilidad, para el que fue creado.

### » Identificadores y JavaBeans

#### → Identificadores Legales

Las reglas del compilador las usa para determinar ya sea si un nombre es legal o no.

### » Reglas de Denominación de JavaBean Listener

- \* Los nombres de los métodos más utilizados para registrar un listener con un origen de evento deben usar el prefijo add, seguido del tipo listener. Por ejemplo: add ActionListener(), es un nombre válido para eventos de acción.
- \* Los nombres de los métodos más utilizados para eliminar (anular el registro) un listener debe usar el prefijo remove, seguido del tipo listener (usando las mismas reglas que el método de agregar registro).
- \* El tipo listener que se agregará o eliminará se debe pasar como el argumento al método.
- \* Los nombres de los métodos de escucha deben terminar con la palabra "Escucha".

### » Declaración de Reglas en Archivos Fuente

- \* Allí puede estar solo una clase pública por archivo código fuente.

- \* Los comentarios pueden aparecer en el inicio o al final de cualquier linea en el archivo código fuente.
- \* Si está ahí una clase pública en el archivo, el nombre del archivo debe tener el nombre de la clase.
- \* Si la clase es parte de un paquete, la declaración del paquete debe ser la primera linea en el archivo de código fuente, antes de cualquier declaración de importación.
- \* Un archivo puede tener más de una clase no pública.
- \* Los archivos sin clases públicas pueden tener un nombre que no coincida con ningún nombre de los del archivo.

## » Declaraciones de Clases y Modificadores

- **Modificadores de Acceso:** public, protected, private
- **Modificadores no de Acceso:** strictfp, final, abstract y etcétera.

## → Clases de Acceso

Acceso significa visibilidad, el ejemplo más común es cuando una clase accede a otra clase y puede instanciar con las variables o métodos dependiendo de su modificador de acceso.

## → Acceso Default

Ningún modificador que lo precede en la declaración, para esto se obtiene el acceso de control cuando no se escribe un modificador en la declaración de clase.

Piensa del Acceso Default como un acceso de nivel paquete, porque unas clases con acceso default solo pueden ser clases con el mismo paquete.

## → Acceso Público

Todas las clases en el universo Java tienen acceso a las clases públicas. Si tratas de usar un paquete diferente desde la clase tu necesitas importarlo de la clase pública.

Ejemplo:

```
Package cert;
public class Beverage {
    //insert todo logic here
    :
}
```

- Los identificadores deben iniciar con una letra, el símbolo de pesos (\$), o un conector de carácter como el guion bajo (\_). Los identificadores nunca se pueden iniciar con un número.
- Después del primer carácter, pueden contener alguna combinación de letras, símbolos de moneda o números.
- En la práctica, no hay límite para el número de caracteres que un identificador puede tener.
- No puedes usar las palabras clave de Java.
- Los identificadores de Java son distinguibles de mayúsculas y minúsculas.

## → Ejemplos de Identificadores legales

```
int -d;
int $c;
int ---2_w;
int this_is_a_very_detailed_name_for_an_identifier;
```

## → Ejemplos de Identificadores Ilegales

int :b;

int -d;

int 7g;

## → Clases e Interfaces

La primera letra debe ser en mayúsculas, y si varias palabras están unidas para formar el nombre, la primera letra de la siguiente palabra debe ser en mayúscula.

## → Métodos

La primera letra debe ser en minúscula, y para las palabras siguientes que forman el nombre, la primera letra se va en mayúscula. A continuación unos ejemplos:

getBalance

hacerCalculo

## → Variables

Al igual que los métodos, se debe utilizar el formato camelCase, comenzando con una letra minúscula. Se recomienda nombres cortos y significativos, que suenan bien para nosotros. A continuación algunos ejemplos:

buttonWidth

accountBalance

myString

## → Constantes

Las constantes de Java se crean marcando variables estáticas y final. Deben nombrarse usando letras mayúsculas con subrayado caracteres como separadores: MIN\_HIGHT

## » Estándares de JavaBeans

La especificación JavaBeans está destinada a ayudar a los desarrolladores de Java a crear componentes de Java que pueden ser utilizados fácilmente por otros desarrolladores de Java en un desarrollo integrado visual Herramienta de entorno (IDE) (como Eclipse o NetBeans). Como programador de Java, desea poder utilizar componentes de la API de Java, pero sería fantástico si también podría comprar el componente Java que deseé de "Beans 'R Us", ese software empieza por la calle. Una vez que haya encontrado los componentes, le gustaría ser accesible a ellos a través de una herramienta de desarrollo de tal manera que no tenga que escribir todo tu código desde cero.

Los métodos que cambian el valor de una propiedad se denominan métodos de establecimiento, y los métodos que recuperan el valor de una propiedad se denominan métodos getter. Las reglas de nomenclatura de JavaBean que necesitan son:

## » Reglas de Denominación de Propiedades de JavaBean

- † Si la propiedad no es booleana, el prefijo del método getter debe ser get. Por ejemplo `getTalla()` es un nombre válido para una propiedad llamada "Talla".

El nombre de la propiedad se infiere del getter y setters, no a través de ninguna variable en su clase. Lo que devuelves from getSize() depende de uno.

- \* Si la propiedad es un valor boolean, el prefijo del método getter es get o is. Por ejemplo getStopped() o isStopped() son nombres válidos para una propiedad boolean.
- \* Se debe establecer el prefijo del método de establecimiento. Por ejemplo setSize() es el valor válido para una propiedad denominada tamaño.
- \* Para completar el nombre de un método getter o setter, cambie la primera letra del nombre de la propiedad en mayúsculas y luego agregarlo al prefijo apropiado (get, is o set).
- \* Las firmas de métodos de establecimiento deben estar marcadas como públicas, con un tipo de retorno nulo, y un argumento que representa el tipo de propiedad.
- \* Las firmas del método getter deben marcarse como públicas, no admitir argumentos y tener un tipo de retorno que coincida con el tipo de argumento del método setter para esa propiedad.

En segundo lugar, la especificación JavaBean admite eventos, que permiten que los componentes notifiquen unos a otros cuando pasa algo. El modelo de eventos se usa a menudo en GUI aplicaciones cuando un evento como un clic del mouse es multifusión a muchos otros objetos que pueden tener cosas que hacer cuando se produce el clic del mouse. Los objetos que reciben la información de que ocurrió un evento llamado listeners.

## » Miembros Públicos

Cuando un método o miembro de variable se declara público, significa todas las demás clases, independientemente del paquete al que pertenecen, pueden acceder al miembro (asumiendo que la clase así mismo es visible).

Para una subclase, si un miembro de su superclase se declara público, la subclase hereda ese miembro independientemente de si ambas clases están en el mismo paquete:

```
package cert;
public class Zoo {
    public String doZooThings() {
        //imagine the fun code that goes here
        return "fun";
    }
}
```

## » Miembros Privados

No se puede acceder a los miembros marcados como privados mediante código en ninguna clase que no sea la clase en la que se declaró el miembro privado. Por ejemplo:

```
package cert;
public class Zoo {
    private String doZooThings() {
        //imagine the fun code that goes here, but only
        //the Zoo
        //Class Knows
        return "fun";
    }
}
```

## » Miembros Protegidos y Predeterminados

Los niveles de control de acceso protegidos y predeterminados son casi idénticos, pero con una diferencia crítica. Se puede acceder a un miembro predeterminado solo si la clase que accede al miembro pertenece al mismo paquete, mientras que se puede acceder a un miembro protegido (a través de la herencia) por una subclase incluso si la subclase está en un paquete diferente.

El comportamiento predeterminado y protegido difieren solo cuando hablamos de subclases. Si la palabra clave protegida se usa para definir un miembro, cualquier subclase de la clase declarando el miembro puede acceder a él a través de la herencia. No importa si la superclase y subclase están en paquetes diferentes, el miembro de superclase protegido todavía es visible para la subclase (aunque visible solo de una manera muy específica como veremos un poco más adelante).

Esto contrasta con el comportamiento predeterminado, que no permite que una subclase acceda a un miembro de superclase a menos que la subclase esté en el mismo paquete que la superclase.