INFERENCIA DE TIPOS EN JAVA - VAR

¿Qué es la inferencia de tipos?

La inferencia de tipos es un proceso en el cual el tipo de datos de una variable se determina automáticamente en tiempo de compilación en función del valor que se le asigna.

La palabra reservada **var** se utiliza para declarar variables locales de forma simplificada, utilizando la inferencia de tipos.

Un ejemplo de esto para entenderlo fácilmente sería:

```
String miString = “Hola Mundo”;
```

Esta sería la manera convencional que hemos usado hasta ahora, pero con la inferencia de tipos se expresaría así:

```
var miVariable = “Hola Mundo”;
```

En este caso, el compilador determina que la variable miVariable es de tipo String, basándose en el valor que se le asigna (“Hola Mundo”). Esto no significa que la variable sea

de tipo Objeto, sino que el compilador determina el tipo de datos más específico posible para que se adapte al valor asignado.

Usos: ¿permitido o no?

La palabra reservada **var** solo se puede utilizar para declarar variables locales y no puede utilizarse para declarar variables de instancia, parámetros de métodos, variables de clase, etc. Las variables de tipo byte y short deben seguir siendo declaradas con su tipo de manera explícita o mediante un casteo.

```
byte num = 1; / var num = (byte)1;
```

Otro ejemplo algo más complejo, usando interface podría ser el siguiente:

```
List<String> lista = new ArrayList<>();
var lista = new ArrayList<>();
```

Simplifica el tener que importar interfaces en una clase con el objetivo de declarar variables inicializadas con una implementación de dicha interface. La inicialización de variables con tipos numéricos nulables () como Integer o Double puede realizarse con el método valueOf:

```
var entero = Integer.valueOf(1);
var doble = Double.valueOf(2d);
```

También simplifica la declaración de variables en los bucles de tipo for/foreach, evitando tener que repetir el tipo de la colección recorrida, y minimizando el impacto de posibles refactorizaciones:

```
for (var elemento : coleccion) {}
```

Variables sin inicializar

Var no se puede utilizar sobre variables que no se inicialicen en el momento de declararse. Realmente es una sintaxis válida, pero Java no sabría qué hacer con ella dado que no puede inferir el tipo de algo que no tiene un valor aún. La idea es que una función x hará las veces de String, en otra de Date y en otra de Long (como en javascript).

```
var x;  // ERROR
```

También podemos encontrar su uso con lambda, pero eso se podrá entender cuando el compañero la explique. Se procede al ejemplo en eclipse de forma inválida:

```
//Método que devuelve un valor ambiguo
static <T> T obtenerResultado() {

    return null;
}
```

```
public static void main(String []args) {

    var resultado = obtenerResultado();
}
```

En este ejemplo, el método obtenerResultado utiliza un tipo genérico <T> y devuelve un valor de ese tipo. Sin embargo, en el método main(), se intenta utilizar la inferencia de tipos con var para declarar la variable resultado y asignarle el resultado de método obtenerResultado(). Dado que el tipo de retorno del método obtenerResultado() es ambiguo y depende de la lógica específica de implementación, el compilador no puede inferir el tipo de dato de la variable resultado, lo que produce un error de compilación.

Veamos pues la forma válida de usar la inferencia:

```
public static void main(String[] args) {

    //Ejemplo con variables simples:

    var numeroEntero = 10;  // inferencia para tipo integer
    var texto = "Prueba de inferencia";  // inferencia para tipo cadena de caracteres
    var pi = 3.14; //inferencia para tipo flotante



    System.out.println("Número entero: "+numeroEntero);
    System.out.println("Texto: "+texto);
    System.out.println("Número doble: "+pi);

}
```

En este ejemplo podemos ver la inferencia de tipos en variable simples.

Veamos un ejemplo algo más complejo:

```
// Inferencia de tipo con ArrayList
var list = new ArrayList<String>();
list.add("Java");
list.add("Python");
// list.add(10); // Esto daría un error de compilación, ya que la lista es de tipo String

// Inferencia de tipo con HashMap
var map = new HashMap<Integer, String>();
map.put(1, "Lunes");
map.put(2, "Martes");

// Iterando sobre el mapa
for (var entry : map.entrySet()) {
    System.out.println(entry.getKey() + " : " + entry.getValue());
}
```

En este ejemplo utilizamos var para declarar una variable list de tipo ArrayList<String> y una variable map de tipo HashMap<Integer, String>. Luego, añadimos elementos a las listas. Y por último iteramos sobre el hashMap utilizando la inferencia de tipo en el bucle for-each.

Si compilamos, obtendremos este resultado:

```
Número entero: 10
Texto: Prueba de inferencia
Número doble: 3.14
1 : Lunes
2 : Martes
```