Resumen de las dudas más comunes

Esta unidad proporciona un resumen de las dudas más comunes que surgen al trabajar con Java, incluyendo temas como la entrada y salida de datos, los tipos de datos primitivos, el casting y el estilo de codificación. Se explican conceptos clave como el uso de los métodos print(), println() y printf() para dar formato a la salida, y el uso de la clase Scanner para la entrada de datos. También se cubren las convenciones de nomenclatura y el uso adecuado de los comentarios en el código.

by Ferran Vilanova

Entrada y salida de datos

Si no queremos dar un "formato" a nuestro texto utilizaremos los métodos print() o println(), siendo println() el que incluye el retorno de carro al final de la salida. Si queremos dar formato, se usa printf() que tiene una sintaxis distinta:

```
System.out.printf("La suma es: %d %n", resultado);
```

Para mostrar decimales, si resultado es 4,2, se usaría:

```
System.out.printf("La suma es: %f4,2 %n", resultado); // Esto muestra 4,200
```

Para la entrada de datos, se utiliza la clase Scanner:

```
Scanner sc = new Scanner(System.in); //Esto crea un objeto de tipo Scanner llamado sc y que leerá del buffer de System.in (origen de los datos)
```

La clase Scanner tiene definidas funciones para recoger del buffer de entrada distintos tipos de datos: sc.nextInt(); sc.nextDouble(); sc.nextLine().

Es importante tener en cuenta que después de recoger un entero de la consola, el usuario puede pulsar enter, quedando en el buffer ese salto de línea que no se puede guardar en un entero. Por eso, la siguiente instrucción de Scanner se encontrará un \\n como primer dato. Para evitar esto, después de nextInt() siempre debemos vaciar el buffer con nextLine().

Además, si la entrada de datos no es correcta, el programa se interrumpirá lanzando un error, por ejemplo, si se espera un entero y el usuario introduce una letra. Otro aspecto importante es que la consola toma el lenguaje de nuestro entorno de desarrollo, por lo que interpreta que los decimales van detrás de una coma (2,1), si introducimos un decimal con un punto, no funcionará.

Tipos de datos primitivos

Es importante tener en cuenta las reglas para los identificadores en Java:

1 Distinción entre mayúsculas y minúsculas

Java hace distinción entre mayúsculas y minúsculas, por lo que var1, Var1 y VAR1 son distintas variables.

2 Palabras reservadas

Los identificadores no pueden ser palabras reservadas del lenguaje, como true, for, if, etc.

3 Unicidad

Los identificadores no pueden ser iguales a otro declarado en el mismo ámbito.

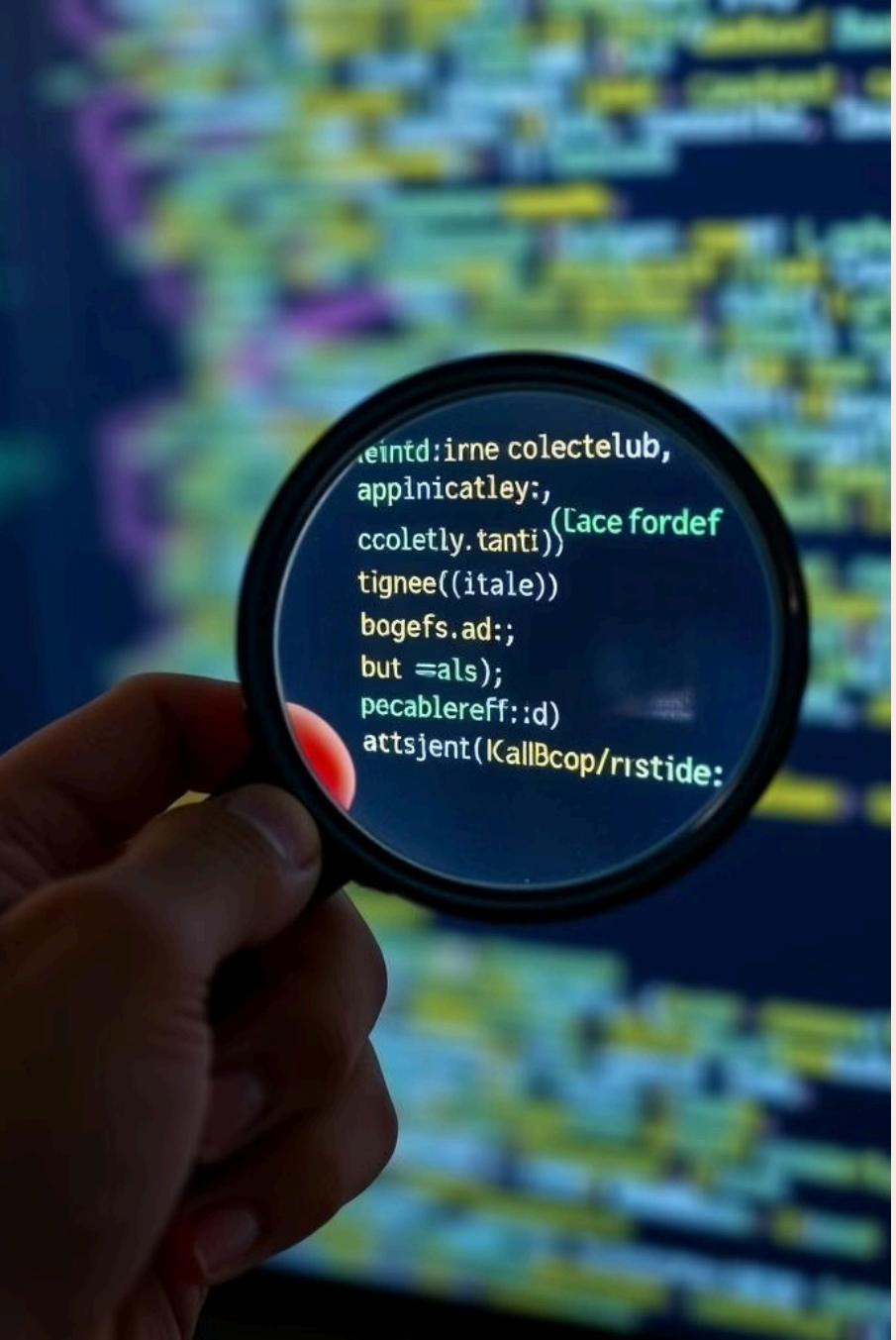
4 Convenciones de nomenclatura

Los nombres de las variables y los métodos deberían empezar por una letra minúscula, y los de las clases por mayúscula. Si el identificador está formado por varias palabras, la primera se escribe en minúsculas (excepto para las clases) y el resto de palabras se empiezan por mayúscula, por ejemplo: añoDeCreación.

Además, es importante tener en cuenta la declaración de los diferentes tipos de datos primitivos:

- Un tipo float se declara con una F al final: float precioPatata =1.2F;
- Un tipo Double se declara con un '.' para separar los decimales, incluso si no son significativos: double precioPatata = 1.0;
- Un tipo carácter se declara entre comillas simples: char car='A';
- Un tipo cadena de texto se declara entre comillas dobles: String txt="Hola!";

Los tipos primitivos NO son objetos, por lo que NO tienen métodos asociados. Sin embargo, existen las llamadas clases Wrapper o Envoltorio, en las que se definen métodos habituales para ese tipo de dato. Estas clases empiezan por mayúscula y no se instancian creando objetos.



Uso de las clases Wrapper

1 Obtener valores máximos y mínimos

```
int n = Integer.MAX_VALUE; //en n se guarda el valor más grande que puede almacenar una variable de tipo entero.
```

2 Conversión de tipos

```
int n = Integer.parseInt("32"); //el texto se convierte a entero si es posible
```

3 Operaciones con tipos

```
mayor = Integer.max(x, y); //asigna a mayor el valor más grande entre x e y
```

Casting o conversión de tipos de datos

A veces es necesario convertir un tipo de datos en otro, por ejemplo, un entero a decimal. Esto se conoce como casting. En una operación de división, por ejemplo, al operador / se le pueden pasar valores enteros y hará una División Entera, truncando los decimales (Ejemplo: 3/ 2 -> 1). Si se quiere que el operador / haga una división con todos los decimales, se deben poner los operandos como valores decimales, por ejemplo, 3.0 / 2.0 --> 1.50000...

Pero si en lugar de los literales 3.0 y 2.0 se tienen esos valores guardados en variables, se deberá especificar que se quieren convertir a entero antes de dividir, así:

```
resultado = (double) x / (double) y;
```

Esto se debe a que los operadores en Java están sobrecargados, lo que significa que se invoca a un método diferente en función de los parámetros que se le pasan. Esto también ocurre con otros operadores, como el signo +, que puede significar tanto concatenación de texto como suma de valores numéricos.

Estilo de codificación

Al igual que cada persona tiene su propio estilo de escritura, cada programador tiene su estilo de codificación. Algunos ponen espacios entre líneas de código, usan identificadores de variables muy cortos, o añaden tabulaciones extra. Sin embargo, es importante seguir ciertas convenciones para que nuestro código sea legible y no cree confusiones:

1 Declaración de variables

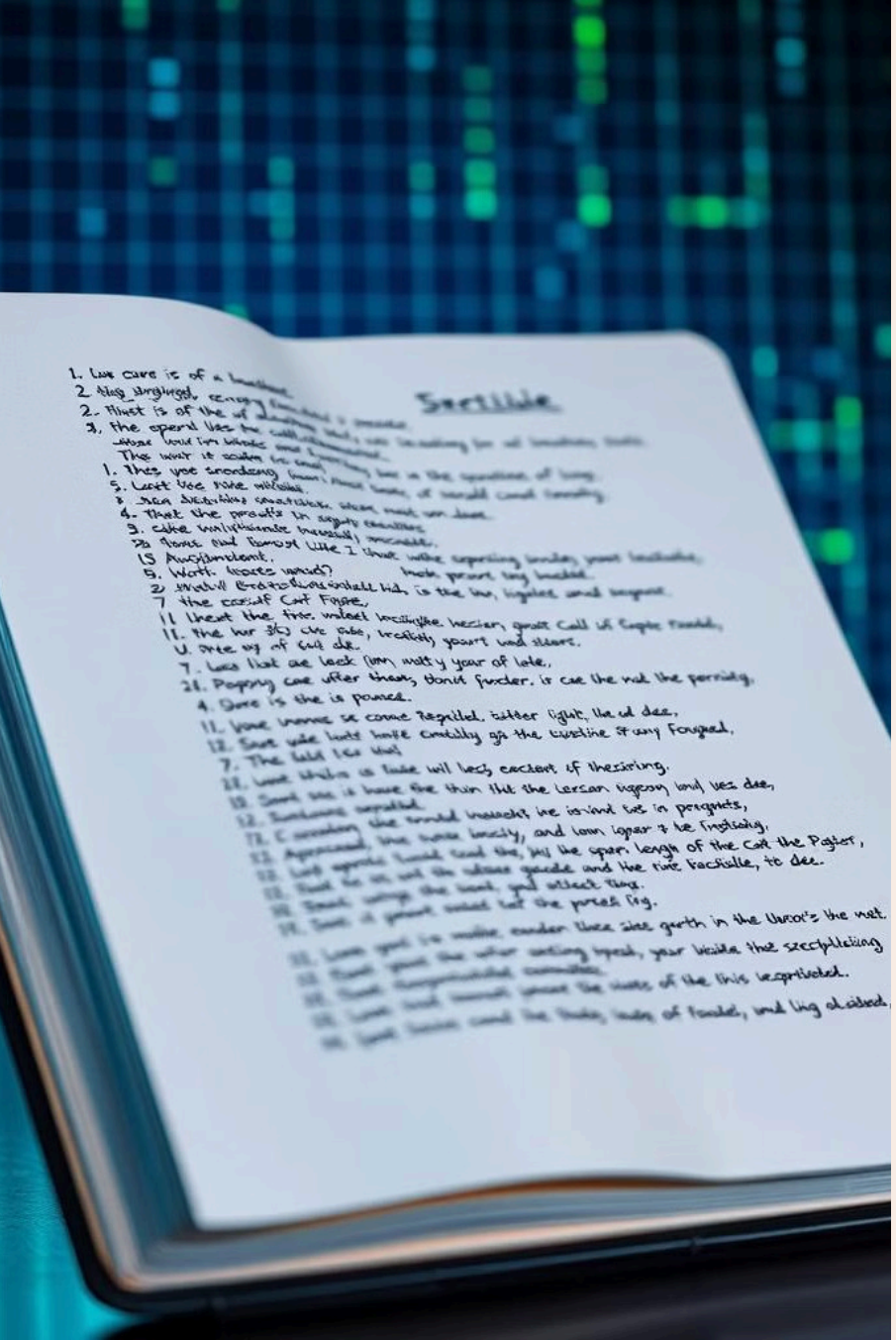
Las variables se deben declarar al principio del código.

2 Comentarios

El código debe estar bien comentado, describiendo en lenguaje natural lo que hace, especialmente en las líneas clave como condiciones de salida o errores encontrados.

3 Indentación

El código debe estar bien indentado. Si no se sabe cómo, la mayoría de entornos de desarrollo, como Eclipse, permiten indentar automáticamente con un atajo de teclado.



Tipos de comentarios

Existen dos tipos principales de comentarios en Java:

1 Comentarios de varias líneas

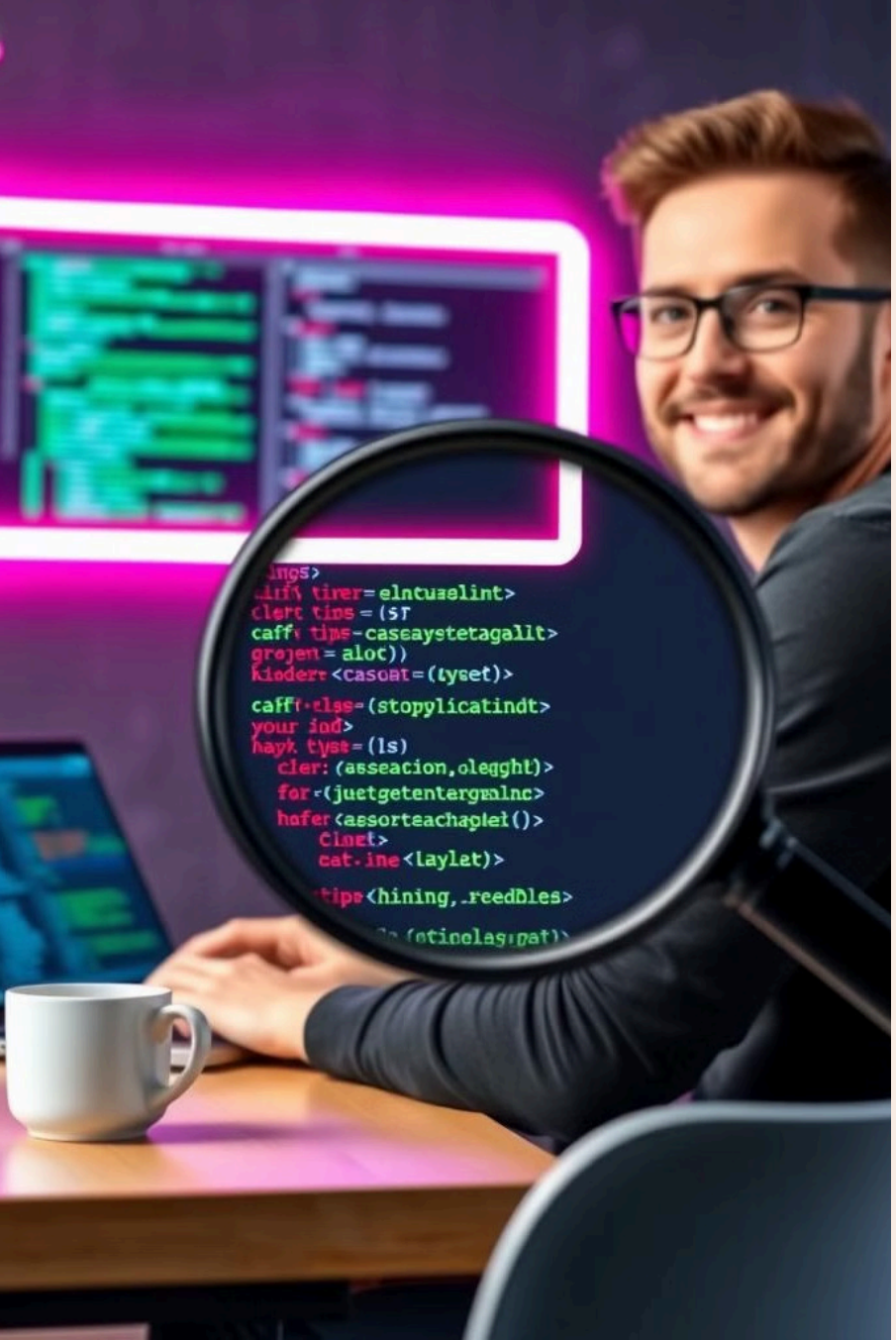
```
/* ... diversas líneas ... */
```

Estos comentarios suelen incluir explicaciones generales del código, como qué problema resuelve, qué datos devuelve o cómo termina.

2 Comentarios de una línea

```
// ... Una sola línea
```

Estos comentarios explican valores importantes, como en un bucle while(x<20) //Para x==20 salimos del bucle.



Resumen

En resumen, esta unidad ha cubierto los conceptos clave relacionados con la entrada y salida de datos en Java, los tipos de datos primitivos y su declaración, el uso de las clases Wrapper, el casting de tipos de datos y las convenciones de estilo de codificación. Estos temas son fundamentales para comprender el funcionamiento básico de Java y escribir código de calidad y legible.