Errores de programación típicos en Java

El nombre de la clase pública no coincide con el nombre del archivo

Cada archivo *.java* puede contener sólo una clase pública (***public class***). El nombre de dicha clase **debe coincidir exactamente** con el nombre del archivo con la extensión *.java* **respetando incluso las mayúsculas y minúsculas**. Por ejemplo, la clase pública ***Lab1Ejer1*** debe estar en el archivo llamado ***Lab1Ejer1.java***. Este es un error de sintaxis.

Una clase no está en el directorio correcto

Este error ocurre cuando el comando *javac* no puede localizar el archivo *.java* correspondiente a una clase en el directorio esperado. Si una clase está en el paquete por defecto (no tiene la declaración del ***package*** en la primera línea), entonces *javac* buscará el fichero correspondiente a dicha clase en la carpeta en la que se ha indicado que está el código fuente (o en el directorio de trabajo actual si no se ha especificado esa opción). Si una clase pertenece a un paquete *packPaquete*, entonces *javac* buscará el archivo correspondiente a dicha clase en el subdirectorio *packPaquete* en la carpeta en la que se ha indicado que está el código fuente (o en el directorio de trabajo actual si no se ha especificado esa opción).

Si el fichero no se encuentra en la ubicación correcta, se produce un error de compilación (ver The declared package … does not match the expected package …*).*

Error en mayúsculas y minúsculas

Java es un lenguaje *case sensitive*, es decir, un carácter en minúsculas (e.g. a) es distinto de su equivalente en mayúsculas. Por lo tanto, el identificador ***miVariable*** es distinto a ***MiVariable***. Esto implica que hay que prestar gran atención sobre todo con las palabras reservadas del lenguaje de programación (e.g. **import, package, …)**. Alterar las mayúsculas y minúsculas puede ocasionar errores de compilación de distinto tipo si se producen en palabras reservadas o en identificadores.

Comparador vs Asignación

Para comprobar si 2 valores son iguales se usa el operador == (el operador de comparación). Para asignar el valor de la derecha a la variable de la izquierda se usa el operador = (operador de asignación).

Un error típico de los programadores novatos a la hora de evaluar la igualdad entre dos valores suele ser el siguiente:

if (miValor = valorEsperado)

Este código intenta evaluar el valorEsperado como un valor booleano en lugar de intentar la evaluación de igualdad entre ***miValor*** y ***valorEsperado***. Si ***miValor*** y ***valorEsperado*** no son valores booleanos, se producirá un error de compilación debido a que el valor de la expresión no es del tipo correcto (*boolean*).

El operador de asignación devuelve el valor asignado, por lo que si ambos valores son booleanos, no se producirá un error de compilación, pero si un error lógico, por lo que el programa no funcionará como se espera.

Lista de errores de compilación más típicos

… cannot be resolved

Indica que la variable no está definida en el ámbito dónde se intenta utilizar. Puede deberse a:

Causa 1: El identificador utilizar en la declaración y en el uso de la variable no coinciden. Comprueba las mayúsculas, minúsculas y otros posibles errores tipográficos.

Causa 2: Puede que la variable esté definida en un bloque (*{}*) y se intente utilizar fuera de ese bloque. La variable sólo es accesible para el bloque en el que está definida.

Causa 3: Puede que la variable esté definida en otro método. No se puede utilizar una variable de otro método (hay que definirla).

… cannot be resolved to a type

Indica que el tipo que se desea utilizar (ya sea una clase o una interfaz) no corresponde a ningún tipo disponible. Puede deberse a:

Causa 1: El nombre del tipo está mal escrito. Comprueba las mayúsculas, minúsculas y otros posibles errores tipográficos.

Causa 2: Falta el ***import*** del paquete que contiene el tipo.

Causa 3: Falta incluir el fichero *.jar* que contiene el tipo en el *classpath*.

The declared package … does not match the expected package …

El nombre del paquete no coincide con el del paquete esperado. La causa más habitual es que el nombre del directorio donde se encuentra la clase no coincide con el nombre del paquete especificado en la primera línea del fichero *java* (***package***) o no se ha especificado el paquete en la clase.

Duplicate local variable …

Este error se produce cuando se declara dos veces una variable dentro del mismo ámbito (e.g., dentro del mismo método o el mismo bloque).

Cannot make a static reference to the non-static field …

Este error se suele producir cuando se intenta acceder a un atributo **no estático** (no lleva el modificador ***static***) desde la parte estática de la clase (método ***main*** o cualquier otro método que lleve el modificador ***static***). Comprueba que el atributo o variable al que se intenta acceder lleve el modificador ***static*** o crea un objeto de la clase que contiene dicho atributo.

The method … is undefined for type …

El método que se está intentando invocar no existe. Comprueba el nombre del método considerando mayúsculas, minúsculas y cualquier otro error tipográfico.

Comprueba que el número y el tipo de los parámetros de la declaración concuerden con los utilizados al invocar el método.

This method must return a result of type …

Este error se puede dar por las siguientes razones:

Causa 1: El tipo del valor devuelto en la instrucción ***return*** no es compatible con el tipo indicado en la declaración del método.

Causa 2: Falta la sentencia ***return***.

Causa 3: Si el método tiene estructuras condicionales (e.g. ***if*** *(){}* ***else****{}*) o bloques ***try****{}* ***catch****(){}*, es posible que se haya omitido la sentencia ***return*** en alguno de los bloques (*{}*). Asegúrate de que haya un ***return*** en todos los bloques. Una forma de evitar este error es poner sólo un ***return*** al final del método y utilizar una variable que contenga el valor a devolver.

Unhandled exception … must be caught or declared to be thrown

No se está gestionando una excepción que puede producirse al ejecutar la aplicación. Debes incluir el código que puede lanzar la excepción dentro de un bloque ***try****{}* ***catch****(){}* o añadir un ***throws*** *NombreExcepcion* en la cabecera del método.

The constructor …() is undefined

Se intenta invocar el constructor por defecto (sin parámetros) de una clase y éste no existe. Cuando se definen constructores específicos para una clase (con parámetros) se pierde el constructor por defecto. Debes definirlo explícitamente.

Lista de errores de ejecución típicos

Los errores que se describen a continuación se producen al ejecutar la aplicación. **No los detecta el compilador**.

NullPointerException

Este error se produce cuando se intenta realizar una operación o acceder a un atributo de un objeto (e.g. ***String***), y éste no ha sido inicializado (en el ejemplo este error se producirá al intentar obtener la longitud con el método ***length()***).

```
String cadena;
System.out.println("La lóngitud de la cadena " + cadena + " es: " +
cadena.length());
```

Asegúrate de que has asignado un valor o has creado un nuevo objeto (***new*** *Tipo()*) del tipo correspondiente antes de realizar cualquier operación sobre el mismo.

IndexOutOfBounds, ArrayIndexOutOfBounds

Este error se produce cuando se intenta accede a una posición no válida de una cadena de caracteres (***IndexOutOfBounds***) o un array (***ArrayIndexOutOfBounds***). Recuerda que en *Java* las posiciones están en el rango [0, *numElementos* - 1].