Un ejemplo de nombre desinformativo sería el uso de la `L` minúscula o la `O` mayúscula como nombres de variables, sobre todo combinados. El problema, evidentemente, es que se parecen a las constantes `1` y `0` respectivamente:

```
int a = l;
if ( O == l )
   a = O1;
else
   l = 01;
```

El lector puede pensar que es una invención, pero hemos visto código con abundancia de estos elementos. En un caso, el autor del código, sugirió usar una fuente distinta para que las diferencias fueran más evidentes, una solución que se hubiera transmitido a todos los futuros programadores como tradición oral o en un documento escrito. El problema se resolvió con carácter definitivo y sin necesidad de crear nuevos productos, con tan sólo cambiar los nombres.

Realizar distinciones con sentido

Los programadores se crean un problema al crear código únicamente dirigido a un compilador o intérprete. Por ejemplo, como se puede usar el mismo nombre para hacer referencia a dos elementos distintos en el mismo ámbito, puede verse tentado a cambiar un nombre de forma arbitraria. En ocasiones se hace escribiéndolo incorrectamente, lo que provoca que los errores ortográficos impidan la compilación[7].

No basta con añadir series de números o palabras adicionales, aunque eso satisfaga al compilador. Si los nombres tienen que ser distintos, también deben tener un significado diferente.

Los nombres de series numéricas (`a1`, `a2`… `aN`) son lo contrario a los nombres intencionados. No desinforman, simplemente no ofrecen información; son una pista sobre la intención del autor. Fíjese en lo siguiente:

```
public static void copyChars(char a1[], char a2[]) {
   for (int i = 0; i < a1.length; i++) {
     a2[i] = a1[i];
   }
}
```

Esta función se lee mejor cuando se usa `source` y `destination` como nombres de argumentos.

Las palabras adicionales son otra distinción sin sentido. Imagine que tiene la clase Product. Si tiene otra clase con el nombre `ProductInfo` o `ProductData`, habrá creado nombres distintos, pero con el mismo significado. `Info` y `Data` son palabras adicionales, como `a`, `an` y `the`.

No es incorrecto usar prefijos como `a` y `the` mientras la distinción tenga sentido. Imagine que usa `a` para variables locales y `for` para argumentos de funciones[8]. El problema aparece cuando decide invocar la variable `theZork` porque ya tiene otra variable con el nombre `zork`.

Las palabras adicionales son redundantes. La palabra `variable` no debe incluirse nunca en el nombre de una variable. La palabra `table` no debe incluirse nunca en el nombre de una tabla. ¿Es mejor `NameString` que `Name`? ¿Podría ser `Name` un número de coma flotante? En caso afirmativo, incumple la regla anterior sobre desinformación. Imagine que encuentra una clase con el nombre `Customer` y otra con el nombre `CustomerObject`. ¿Cuál sería la distinción? ¿Cuál representa mejor el historial de pagos de un cliente?

Existe una aplicación que lo ilustra. Hemos cambiado los nombres para proteger al culpable. Veamos el error exacto:

```
getActiveAccount();
getActiveAccounts();
getActiveAccountInfo();
```

¿Cómo saben los programadores de este proyecto qué función deben invocar?

En ausencia de convenciones concretas, la variable `moneyAmount` no se distingue de `money`, `customerInfo` no se distingue de `customer`, `accountData` no se distingue de `account` y `theMessage` no se distingue de `message`. Debe diferenciar los nombres de forma que el lector aprecie las diferencias.