```
    return flaggedCells;
  }
```

Con estos sencillos cambios de nombre, es fácil saber qué sucede. Es la ventaja de seleccionar nombres adecuados.

Evitar la desinformación

Los programadores deben evitar dejar pistas falsas que dificulten el significado del código. Debemos evitar palabras cuyo significado se aleje del que pretendemos. Por ejemplo, `hp`, `aix` y `sco` son nombres de variables pobres ya que son los nombres de plataformas o variantes de Unix. Aunque se trate del código de una hipotenusa y `hp` parezca la abreviatura correcta, puede no serlo.

No haga referencia a un grupo de cuentas como `accountList` a menos que realmente sea una lista (`List`). La palabra lista tiene un significado concreto para los programadores. Si el contenedor de las cuentas no es realmente una lista, puede provocar falsas conclusiones[6]. Por tanto, resulta más adecuado usar `accountGroup`, `bunchOfAccounts` o simplemente `accounts`.

Evite usar nombres con variaciones mínimas. ¿Cuánto se tarda en apreciar la sutil diferencia entre `XYZControllerForEfficientHandlingOfStrings` y `XYZControllerForEfficientStorageOfStrings` en un módulo? Ambas palabras tienen una forma similar.

La ortografía similar de conceptos parecidos es información; el uso de ortografía incoherente es desinformación. En los entornos modernos de Java, el código se completa de forma automática. Escribimos varios caracteres de un nombre y pulsamos varias teclas para obtener una lista de posibles opciones de un nombre. Es muy útil si los nombres de elementos similares se ordenan alfabéticamente de forma conjunta y si las diferencias son muy evidentes, ya que es probable que el programador elija un objeto por nombre sin fijarse en los comentarios o la lista de métodos proporcionados por una clase.

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: