Los métodos deben tener nombres de verbo como `postPayment`, `deletePage` o `save`. Los métodos de acceso, de modificación y los predicados deben tener como nombre su valor y usar como prefijo `get`, `set` e `is` de acuerdo al estándar de javabean[9].

```
string name = employee.getName();
customer.setName("mike");
if (paycheck.isPosted())…
```

Al sobrecargar constructores, use métodos de factoría estáticos con nombres que describan los argumentos. Por ejemplo:

```
Complex fulcrumPoint = Complex.FromRealNumber(23.0);
```

es mejor que:

```
Complex fulcrumPoint = new Complex(23.0);
```

Refuerce su uso convirtiendo en privados sus constructores correspondientes.

No se exceda con el atractivo

Si los nombres son demasiado inteligentes, sólo los recordarán los que compartan el sentido del humor de su autor, y sólo mientras se acuerden del chiste. ¿Sabrán qué significa la función `HolyHandGrenade`? Sin duda es atractiva, pero en este caso puede que `DeleteItems` fuera más indicado. Opte por la claridad antes que por el entretenimiento. En el código, el atractivo suele aparecer como formas coloquiales o jergas. Por ejemplo, no use `whack()` en lugar de `kill()`. No recurra a bromas culturales como `eatMyShorts()` si quiere decir `abort()`.

Diga lo que piense. Piense lo que diga.

Una palabra por concepto

Elija una palabra por cada concepto abstracto y manténgala. Por ejemplo,

resulta confuso usar `fetch`, `retrieve` y `get` como métodos equivalentes de clases distintas. ¿Cómo va a recordar qué método se corresponde a cada clase? Desafortunadamente, tendrá que recordar qué empresa, grupo o individuo ha creado la biblioteca o clase en cuestión para recordar qué término se ha empleado. En caso contrario, perderá mucho tiempo buscando en encabezados y fragmentos de código.

Los entornos de edición modernos como Eclipse e IntelliJ ofrecen pistas sensibles al contexto, como la lista de métodos que puede invocar en un determinado objeto. Pero esta lista no suele incluir los comentarios de nombres de funciones y listas de parámetros. Tendrá suerte si muestra los nombres de parámetros de las declaraciones de funciones. Los nombres de funciones deben ser independientes y coherentes para que pueda elegir el método correcto sin necesidad de búsquedas adicionales.

Del mismo modo, resulta confuso tener un controlador, un administrador y un control en la misma base de código. ¿Cuál es la diferencia entre `DeviceManager` y `ProtocolController`? ¿Por qué no son los dos controladores o administradores? ¿Son controladores? El nombre hace que espere que dos objetos tengan un tipo diferente y clases diferentes.

Un léxico coherente es una gran ventaja para los programadores que tengan que usar su código.

No haga juegos de palabras

Evite usar la misma palabra con dos fines distintos. Suele hacerse en juegos de palabras. Si aplica la regla de una palabra por conceptos, acabará con muchas clases que por ejemplo tengan un método `add`. Mientras las listas de parámetros y los valores devueltos de los distintos métodos `add` sean semánticamente equivalentes, no hay problema.

Sin embargo, alguien puede decidir usar la palabra `add` por motivos de coherencia, aunque no sea en el mismo sentido. Imagine que hay varias clases en las que `add` crea un nuevo valor sumando o concatenando dos valores existentes. Imagine ahora que crea una nueva clase con un método