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

que añada su parámetro a una colección. ¿Este método debe tener el método `add`? Parece coherente ya que hay otros muchos métodos `add`, pero en este caso hay una diferencia semántica, de modo que debemos usar un nombre como `insert` o `append`. Llamar `add` al nuevo método sería un juego de palabras.

Nuestro objetivo, como autores, es facilitar la comprensión del código. Queremos que el código sea algo rápido, no un estudio exhaustivo. Queremos usar un modelo en el que el autor sea el responsable de transmitir el significado, no un modelo académico que exija investigar el significado mostrado.

Usar nombres de dominios de soluciones

Recuerde que los lectores de su código serán programadores. Por ello, use términos informáticos, algoritmos, nombres de patrones, términos matemáticos y demás. No conviene extraer todos los nombres del dominio de problemas ya que no queremos que nuestros colegas tengan que preguntar el significado de cada nombre en especial cuando ya conocen el concepto bajo otro nombre diferente.

El nombre `AccountVisitor` tiene mucho significado para un programador familiarizado con el patrón `VISITOR`. ¿Qué programador no sabe lo que es `JobQueue`? Hay cientos de cosas técnicas que los programadores tienen que hacer y elegir nombres técnicos para dichas cosas suele ser lo más adecuado.

Usar nombres de dominios de problemas

Cuando no exista un término de programación para lo que esté haciendo, use el nombre del dominio de problemas. Al menos el programador que mantenga su código podrá preguntar el significado a un experto en dominios.

Separar los conceptos de dominio de soluciones y de problemas es parte del trabajo de un buen programador y diseñador. El código que tenga más relación con los conceptos del dominio de problemas tendrá nombres