Usar nombres que se puedan pronunciar

A los humanos se nos dan bien las palabras. Gran parte de nuestro cerebro se dedica al concepto de palabras. Y, por definición, las palabras son pronunciables. Sería una pena malgastar esa parte de nuestro cerebro dedicada al lenguaje hablado. Por tanto, cree nombres pronunciables. Si no lo puede pronunciar, no podrá explicarlo sin parecer tonto. Es un factor importante, ya que la programación es una actividad social.

Conozco una empresa que usa `genymdhms` (fecha de generación, año, mes, día, hora, minuto y segundo) y lo pronuncian tal cual. Yo tengo la costumbre de pronunciar todo tal y como lo veo escrito, de forma que muchos analistas y diseñadores acabaron por llamarme algo como «genimedemes». Era un chiste y nos parecía divertido, pero en realidad estábamos tolerando el uso de nombres pobres. Teníamos que explicar las variables a los nuevos programadores y cuando las pronunciaban, usaban palabras inventadas en lugar de nombres correctos. Compare:

```
class DtaRcrd102 {
   private Date genymdhms;
   private Date modymdhms;
   private final String pszqint = “102”;
   /*… */
};
```

con:

```
class Customer {
   private Date generationTimestamp;
   private Date modificationTimestamp;
   private final String recordId = “102”;
   /*… */
};
```

Ahora se puede mantener una conversación inteligente: «Eh, Mikey, fíjate en este registro. La marca de tiempo de generación es para mañana. ¿Cómo es posible?»

Usar nombres que se puedan buscar

Los nombres de una letra y las constantes numéricas tienen un problema: no son fáciles de localizar en el texto. Se puede detectar `MAX_CLASSES_PER_STUDENT`, pero el número 7 resulta más complicado. Las búsquedas pueden devolver el dígito como parte de nombres de archivo, otras definiciones de constantes o expresiones en las que se use con otra intención. Mucho peor si la constante es un número extenso y alguien ha intercambiado los dígitos, lo que genera un error inmediato y no aparece en la búsqueda.

Del mismo modo, el nombre `e` es una opción muy pobre para variables que el programador tenga que buscar. Es la letra más usada en inglés y aparece en la práctica totalidad de los textos de un programa. A este respecto, los nombres extensos superan a los breves y cualquier nombre que se pueda buscar supera a una constante en el código.

Personalmente prefiero nombres de una letra que sólo se puedan usar como variables locales dentro de métodos breves. *La longitud de un nombre debe corresponderse al tamaño de su ámbito* [N5]. Si una variable o constante se usa en varios puntos del código, debe asignarle un nombre que se pueda buscar. Compare:

```
for (int j=0; j<34; j++) {
   s += (t[j]*4)/5;
}
```

con:

```
int realDaysPerIdealDay = 4;
const int WORK_DAYS_PER_WEEK = 5;
int sum = 0;
for (int j = 0; j < NUMBER_OF_TASKS; j++) {
   int realTaskDays = taskEstimate[j] * realDaysPerIdealDay;
   int realTaskWeeks = (realdays / WORK_DAYS_PER_WEEK);
   sum += realTaskWeeks;
}
```

En este ejemplo, `sum` no es un nombre especialmente útil, pero al menos se puede buscar. Se usa una función más extensa, pero comprobará que resulta mucho más fácil buscar `WORK_DAYS_PER_WEEK` que todas las instancias de `5` y filtrar la lista a los casos con el significado adecuado.

Evitar codificaciones