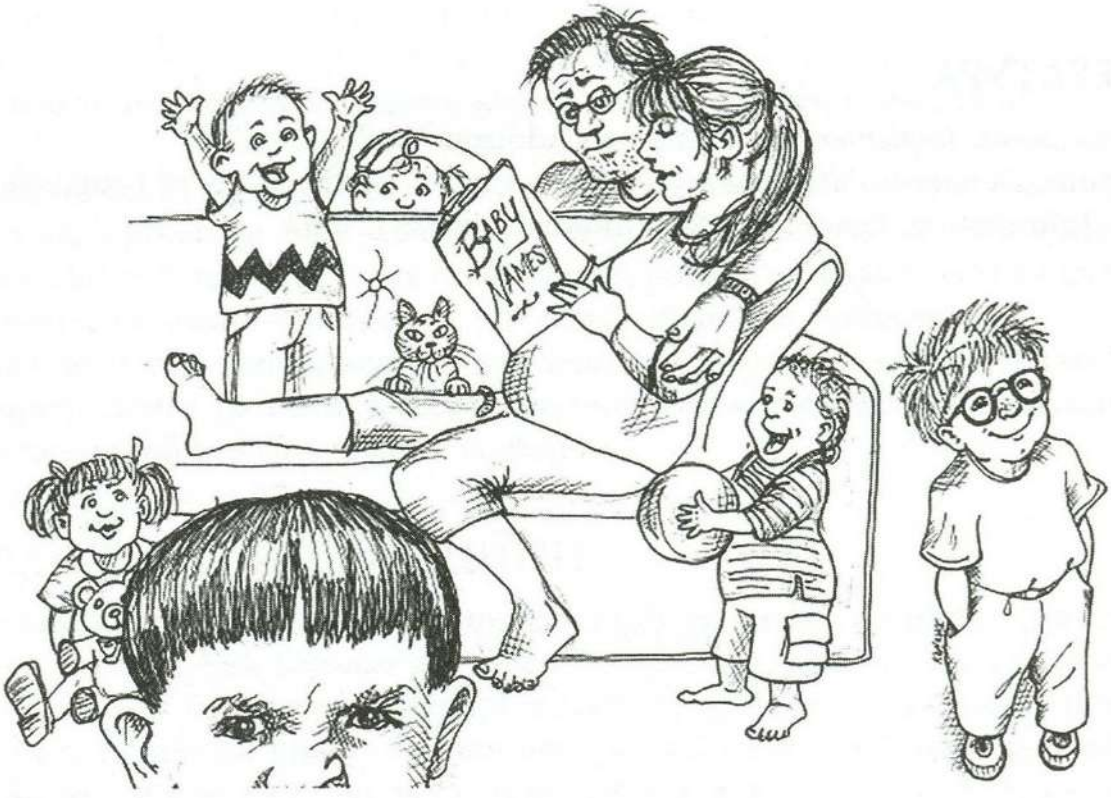
Зміст

Розділ 2

ЗМІСТОВНІ ІМЕНА

Тім Оттінгер



Імена трапляються в програмуванні повсюдно. Ми присвоюємо імена своїм змінним, функціям, аргументам, класам і пакетам. Ми присвоюємо імена вихідним файлам і каталогам, у яких вони зберігаються. Ми присвоюємо імена файлам `jar`, `war` та `ear`. Імена, імена, імена... Але те, що роблять так часто, слід робити добре. Далі наводяться деякі прості правила створення хороших імен.

Імена мають передавати наміри програміста

Легко сказати: імена мають передавати наміри програміста. І все ж до вибору імен слід ставитися *серйозно*. Щоб вибрати гарне ім'я, знадобиться час, але економія окупить витрати. Отже, стежте за іменами у своїх програмах і змінюйте їх, якщо знайдете більш вдалі варіанти. Цим ви спрощуєте життя кожному, хто читатиме ваш код (і собі самому також).

Ім'я змінної, функції або класу має відповідати на всі головні питання. Воно має повідомити, чому ця змінна (або щось інше) існує, що вона робить і як використовується. Якщо ім'я вимагає додаткових коментарів, воно зазвичай не передає намірів програміста.

```
int d; // elapsed time in days
```

Ім'я `d` не передає зовсім нічого. Воно не асоціюється ні з часовими інтервалами, ні з днями. Його слід замінити іншим ім'ям, яке вказує, що саме вимірюють і в яких одиницях:

```
int elapsedTimeInDays;
int daysSinceCreation;
int daysSinceModification;
int fileAgeInDays;
```

Змістовні імена істотно спрощують розуміння і модифікацію коду. Наприклад, що робить нижченаведений фрагмент?

```
public List<int[]> getThem() {
  List<int[]> list1 = new ArrayList<int[]>();
  for (int[] x : theList)
    if (x[0] == 4)
      list1.add(x);
  return list1;
}
```

Чому ми не можемо негайно сказати, що робить цей код? У ньому немає складних виразів. Пробіли й відступи розставлені грамотно. У коді задіяні тільки три змінні й дві константи. У ньому немає жодних хитромудрих класів або поліморфних методів, тільки список масивів (принаймні позір).

Проблема криється не в складності коду, а в його *неочевидності*, тобто в ступені, у якому контекст не походить із власне коду. Код неявно вимагає, щоби ми знали відповіді на такі запитання:

1. Які дані зберігаються в `theList`?
2. Чим так важливий елемент `theList` із нульовим індексом?
3. Який особливий сенс має значення `4`?
4. Як будуть використовувати список, що повертається?