1. **Негласные правила именования переменной/функции/класса.**

**Имена должны передавать намерения программиста.**

Имя переменной, функции или класса должно отвечать на все главные вопросы. Оно должно сообщить, почему эта переменная (и т. д.) существует, что она делает и как используется. Если имя требует дополнительных комментариев, значит, оно не передает намерений программиста.

**Избегайте дезинформации**

Программисты должны избегать ложных ассоциаций, затемняющих смысл кода. Не используйте слова со скрытыми значениями, отличными от предполагаемого. Например, переменным не стоит присваивать имена hp, aix, and sco, потому что они ассоциируются с платформами и разновидностями Unix. Даже если в переменной хранится длина гипотенузы и имя hp кажется хорошим сокращением, оно может ввести в заблуждение читателя кода

**Используйте осмысленные различия**

Недостаточно добавить в имя серию цифр или неинформативные слова, даже если компилятору этого будет достаточно. Если имена различаются, то они должны обозначать разные понятия. «Числовые ряды» вида (a1, a2, .. aN) являются противоположностью сознательного присваивания имен. Такие имена не дезинформируют — они просто не несут информации и не дают представления о намерениях автора. Пример:

public static void copyChars(char a1[], char a2[]) {

for (int i = 0; i < a1.length; i++) {

a2[i] = a1[i];

}

}

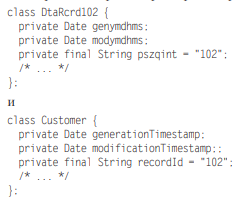
Такая функция будет читаться намного лучше, если присвоить аргументам имена source и destination.

Неинформативные слова также применяются для создания бессодержательных различий. Допустим, у вас имеется класс Product. Создав другой класс с именем ProductInfo или ProductData, вы создаете разные имена, которые по сути обозначают одно и то же. Info и Data не несут полезной информации, как и артикли a, an и the.

**Используйте удобопроизносимые имена**

В одной известной мне компании используется переменная genymdhms (дата генерирования, год, месяц, день, час, минуты и секунды), поэтому программисты упоминали в своих разговорах «ген-уай-эм-ди-эйч-эм-эс». У меня есть противная привычка произносить все так, как написано, поэтому я начал говорить «генъямадда-химс». Потом переменную начали так называть многие проектировщики и аналитики, и это звучало довольно глупо.

Новым разработчикам приходилось объяснять смысл переменных, после чего они начинали изъясняться дурацкими неестественными словами вместо нормальной разговорной речи. Сравните:



**Выбирайте имена, удобные для поиска**

Строка MAX\_CLASSES\_PER\_STUDENT отыскивается легко, а с числом 7 могут возникнуть проблемы. Система поиска находит эту цифру в именах файлов, в определениях констант и в различных выражениях, где значение используется с совершенно другим смыслом.

Также не стоит присваивать имя e переменной, которая может использоваться при поиске. Самая распространенная буква английского алфавита с большой вероятностью встречается в любом текстовом фрагменте каждой программы

Лично я считаю, что однобуквенные имена могут использоваться ТОЛЬКО для локальных переменных в коротких методах. Длина имени должна соответствовать размеру его области видимости [N5]. Если переменная или константа может встречаться или использоваться в нескольких местах кодового блока, очень важно присвоить ей имя, удобное для поиска.

**Префиксы членов классов**

Префиксы m\_, которыми когда-то снабжались переменные классов, тоже стали ненужными. Классы и функции должны быть достаточно компактными, чтобы вы могли обходиться без префиксов.

Кроме того, люди быстро учатся игнорировать префиксы (и суффиксы), чтобы видеть содержательную часть имени. Чем больше мы читаем код, тем реже замечаем префиксы.

**Имена классов**

Имена классов и объектов должны представлять собой существительные и их комбинации: Customer, WikiPage, Account и AddressParser. Старайтесь не использовать в именах классов такие слова, как Manager, Processor, Data или Info. Имя класса не должно быть глаголом.

**Имена методов**

Имена методов представляют собой глаголы или глагольные словосочетания: postPayment, deletePage, save и т. д. Методы чтения/записи и предикаты образуются из значения и префикса get, set и is согласно стандарту javabean1 .

string name = employee.getName();

customer.setName("mike");

if (paycheck.isPosted())...

При перегрузке конструкторов используйте статические методы-фабрики с именами, описывающими аргументы. Например, запись

Complex fulcrumPoint = Complex.FromRealNumber(23.0);

обычно лучше записи

Complex fulcrumPoint = new Complex(23.0);

Рассмотрите возможность принудительного использования таких методов; для этого соответствующие конструкторы объявляются приватными.