Доклад-презентация

Серьезный доклад для серьезных людей,

Использованы ресурсы астрала, интернета и кадры из сериала “IT crowd”

Декомпиляция и обфускация

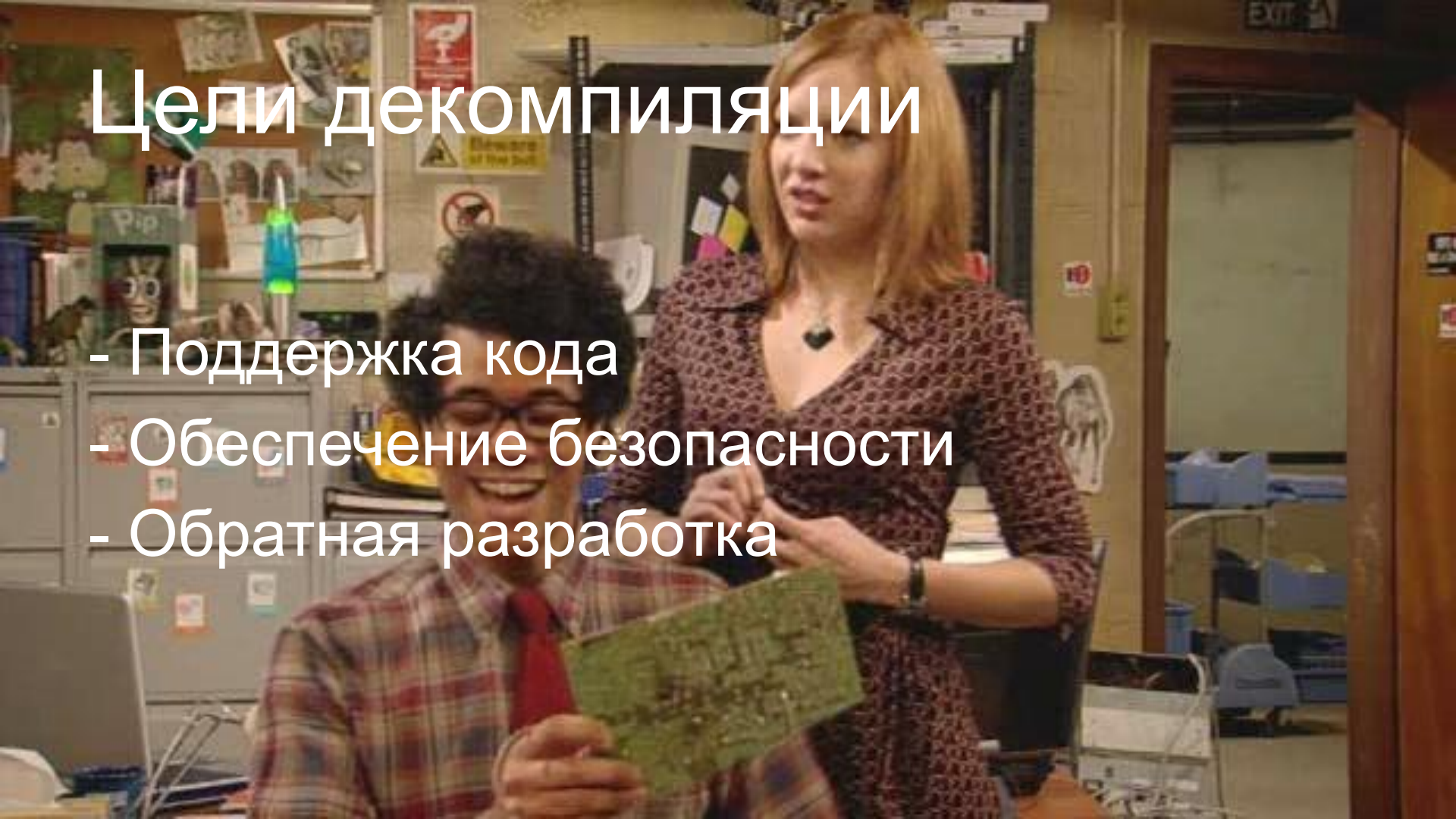
Декомпиляция

Трансляция исполняемого кода в исходный.

Возможна благодаря:

- Массе лишней информации об оригинальном коде в скомпилированной проге
- Однотипности языков в\у
- Информации о компиляторе
- Наличию промежуточных кодов

Цели декомпиляции



- Поддержка кода
- Обеспечение безопасности
- Обратная разработка

Декомпиляторы Java

- “Jad” Java-декомпилятор
- “Cavaj” Java-декомпилятор
- “Mocha” Java-декомпилятор
- “DeCafePro” от DeCafe, Франция
- “SourceTech Java-декомпилятор” от SourceTech Corp
- “SourceAgain” от Ahpah Corp
- “Class Cracker” от Mayon Software, Австралия
- “IceBreaker” от BreakerTech corp, Великобритания
- “NMI Java-декомпилятор” от NMI
- DJ Java Decompiler - декомпилятор Java-классов.

Обфускация

Запутывание кода программы с целью осложнения декомпиляции.

Возможна на уровнях:

- Алгоритма
- Исходного текста
- Машинных кодов

Обоснованность обфускации

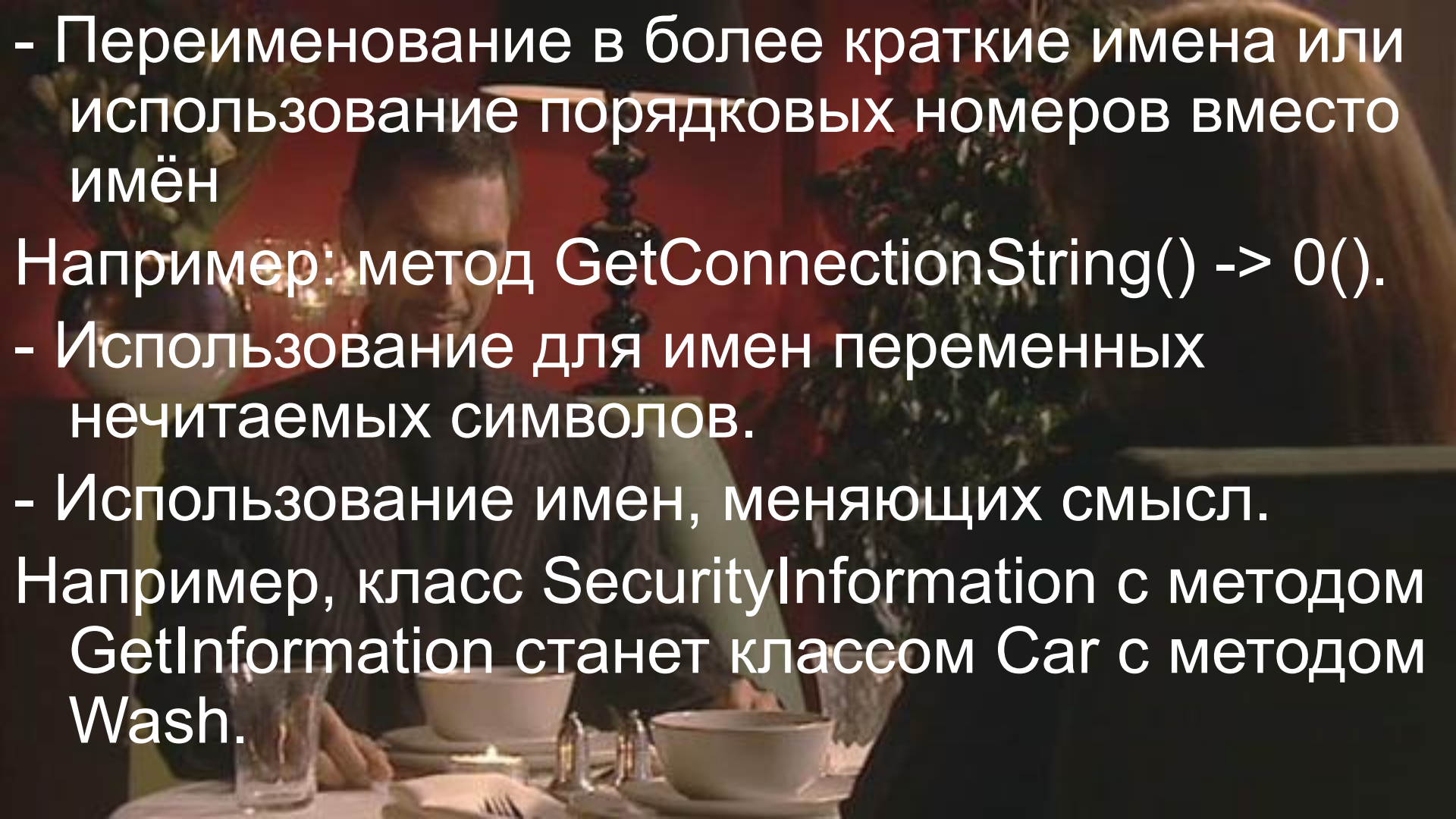
- Время на понимание > времени актуальности алгоритма
- Цена обфускации > цены продукта

Методы обфускации

- Переименование методов, переменных и т.д. в набор бессмысленных символов.

Например: метод GetPassword() после обфускации будет иметь имя KJHS92DSLKaf().

Проблема: многие декомпиляторы, встречая на своем пути подобного рода имена, заменяют их наболее осмысленные (method_1, method_2), тем самым сводя всю работу обфускатора на нет.

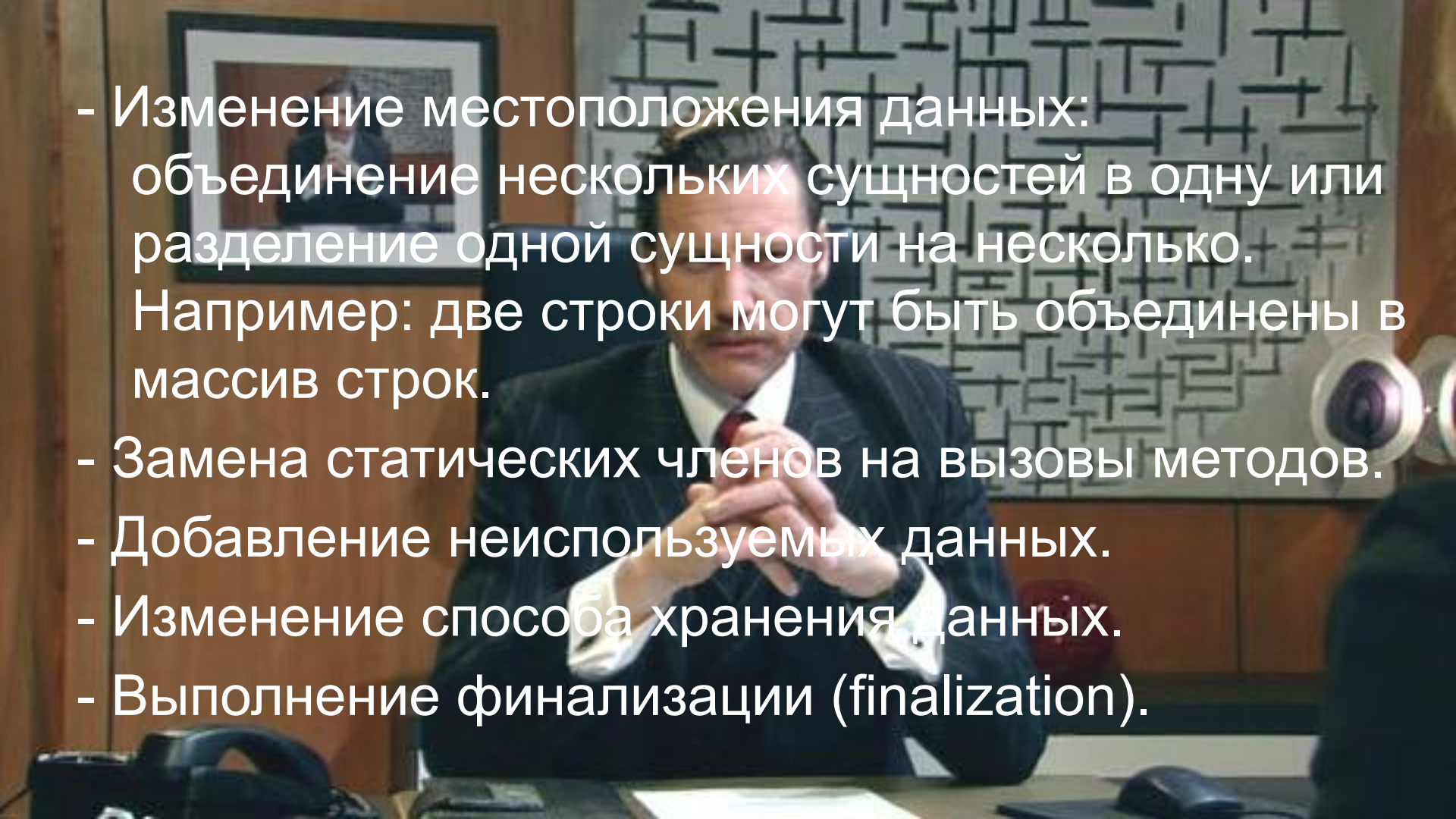


- Переименование в более краткие имена или использование порядковых номеров вместо имён

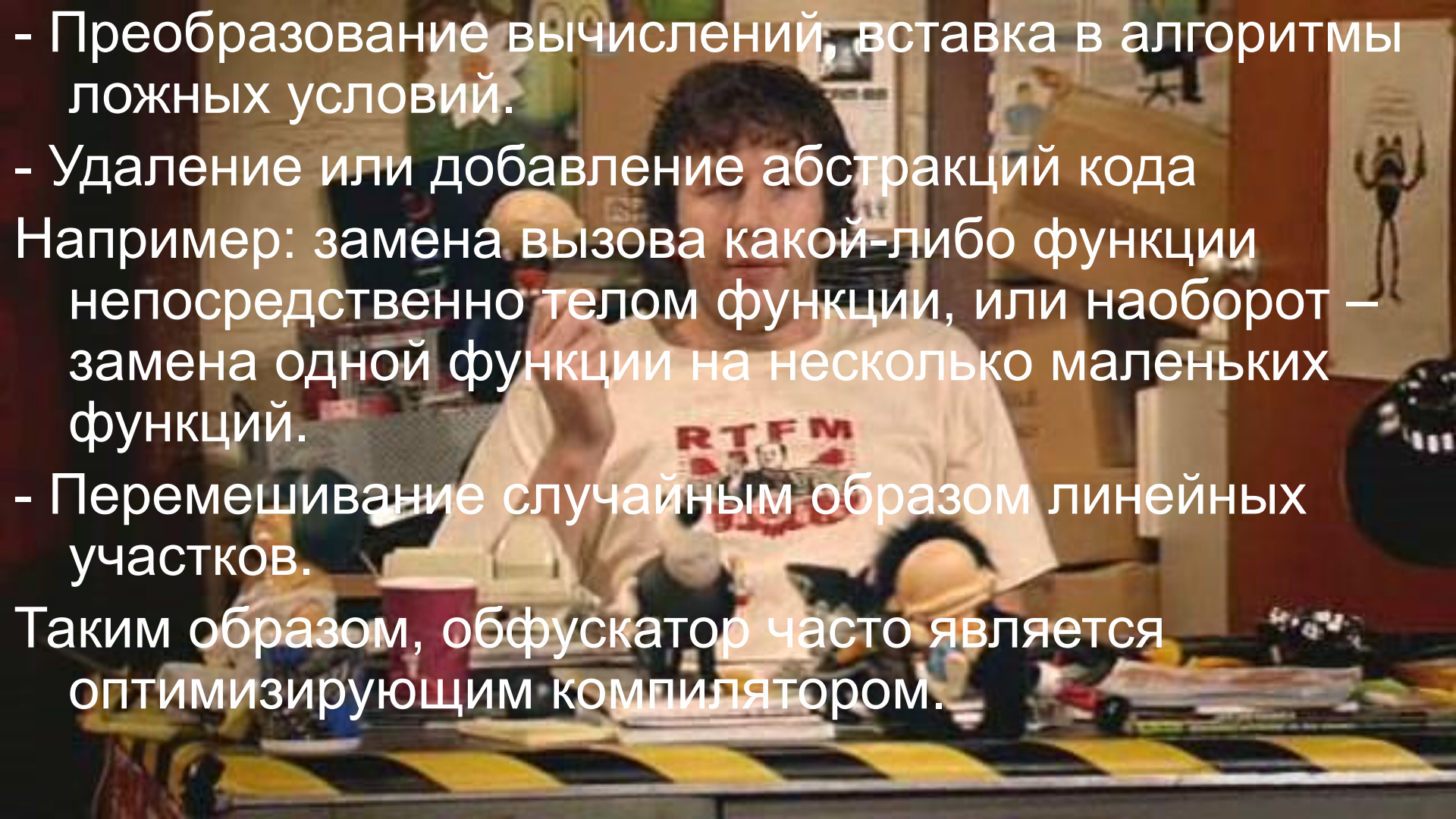
Например: метод GetConnectionString() -> 0().

- Использование для имен переменных нечитаемых символов.
- Использование имен, меняющих смысл.

Например, класс SecurityInformation с методом GetInformation станет классом Car с методом Wash.



- Изменение местоположения данных: объединение нескольких сущностей в одну или разделение одной сущности на несколько. Например: две строки могут быть объединены в массив строк.
- Замена статических членов на вызовы методов.
- Добавление неиспользуемых данных.
- Изменение способа хранения данных.
- Выполнение финализации (finalization).



- Преобразование вычислений, вставка в алгоритмы ложных условий.
- Удаление или добавление абстракций кода

Например: замена вызова какой-либо функции непосредственно телом функции, или наоборот – замена одной функции на несколько маленьких функций.

- Перемешивание случайным образом линейных участков.

Таким образом, обфускатор часто является оптимизирующим компилятором.

Пример обфускации

```
private void loadPixie( URL url ) throws Exception
{
URLConnection connection = url.openConnection();
connection.setUseCaches( true );
in = new DataInputStream( connection.getInputStream() );
// Verify file format via magic numbers and version number.
if (in.readInt() != Constants.MAGIC)
throw new Exception( "Bad Pixie header" );
int v = in.readShort();
…}
private void mth015E(void 867 % static 931)
{void short + = 867 % static 931.openConnection(); short +.setUseCaches(true);
private01200126013D = new DataInputStream(short
+.getInputStream());
if(private01200126013D.readInt() != 0x5daa749) throw new Exception("Bad Pixie header");
void do const throws = private01200126013D.readShort();
…}
```