ソフトウェア難読化技術

- ソフトウェアの流れ

開発者

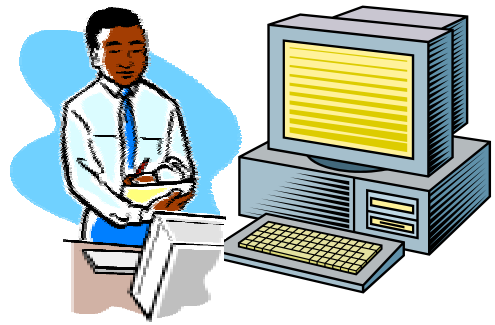
```
read(a);
read(b);
c=a+b;
print(c);
```

ソースコード

コンパイル →

```
01010101
10101110
11101101
00010110
10011001
```

実行可能ファイル →



利用者

アルゴリズム・データの盗用ソフトウェアの改ざん

```
read(a);
read(b);
c=a+b;
print(c);
```

← 逆コンパイル

```
01010101
10101110
11101101
00010110
10011001
```

悪意ある利用者

- 問題点

逆コンパイラなどのツールにより，ソースコードの復元は容易になりつつある

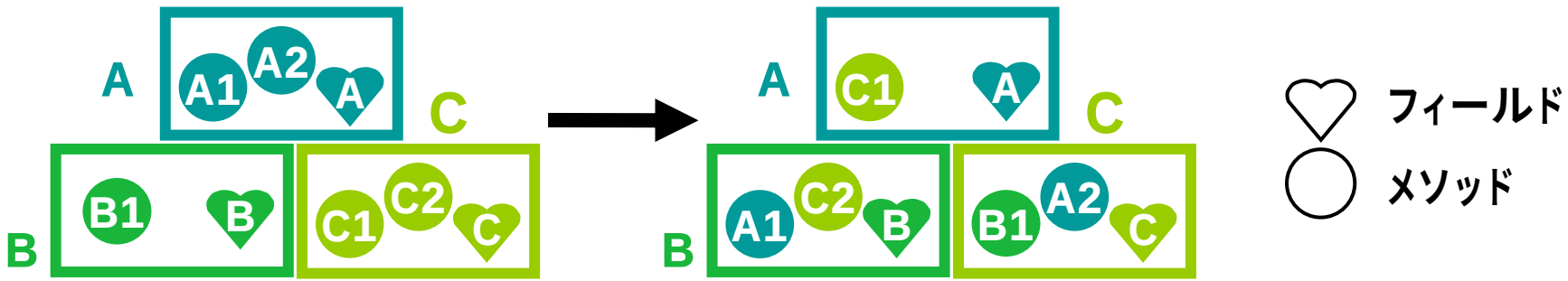
→ ソフトウェアの知的著作権の保護が困難に

JAVAにおいてはこの傾向が顕著である

外部からの観測・改ざんが困難なソフトウェアが必要

JAVAに対する難読化手法

- **難読化手法**
  - JAVAではデータ構造とアルゴリズムがクラスファイルとして，一まとめに定義される
  - メソッド（アルゴリズム）をクラスファイル間に分散させることにより，データ構造とアルゴリズムの分離が実現できる．



全てのフィールド，メソッドを外部に公開する
⇒他のクラスファイルから各フィールドへのアクセスが可能に

- **難読化の効果**
  - ソフトウェアの可読性の低下
    ソフトウェアを理解するためには，クラスファイル間の解析が必要になる
  - クラスファイルの盗用の防止
    アルゴリズムとデータ分離されており，単体ではもはや意味をなさない