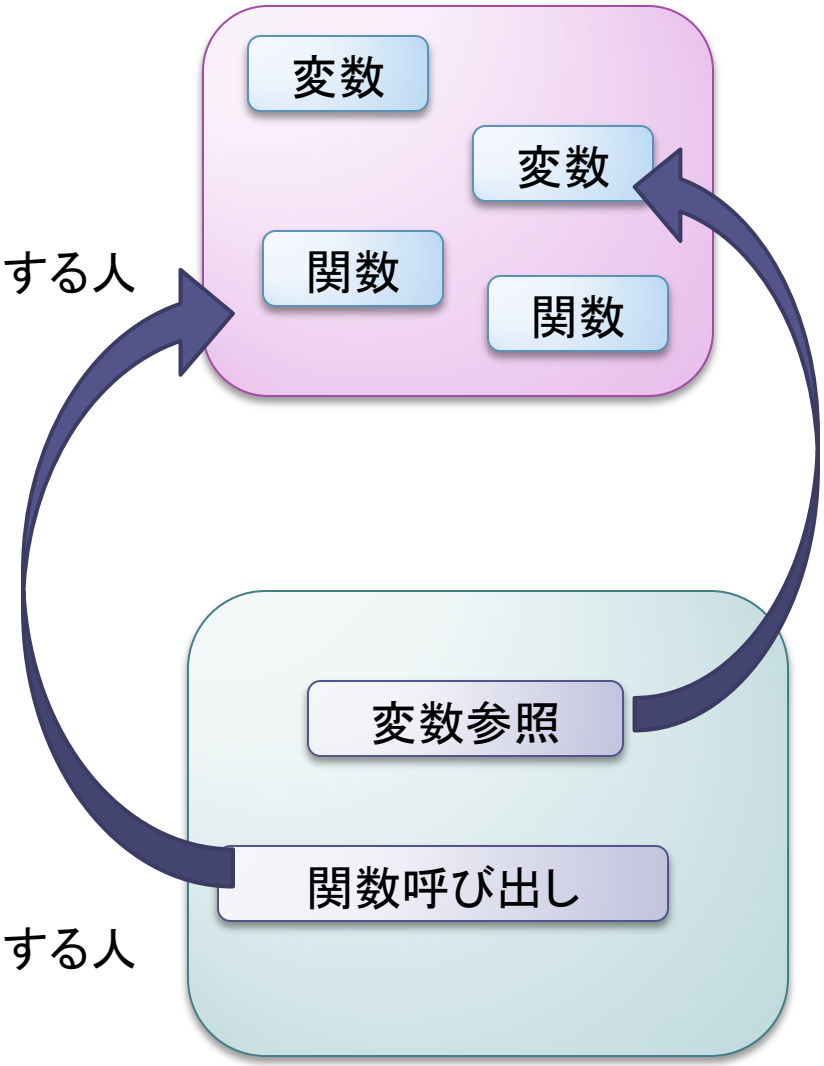
オブジェクト指向プログラミング（OOP:Object-oriented programming)

- データ隠ぺい（カプセル化）
- 継承（インヘリタンス）
- 多態（ポリモーフィズム）

- プログラムの巨大化・複雑化と、これにともなうバグの増加・プログラムの見通しの悪さを解決
- プログラム部品（コンポーネント）の再利用性を高める
- オブジェクト指向言語を学ぶには、その機能を「覚える」のではなく、それが「なぜ」必要になったかを理解しよう

カプセル化はなぜ必要か

コンポーネントを提供する人
Alice さん



コンポーネント（まとまった機能）を、変数や関数の形で提供

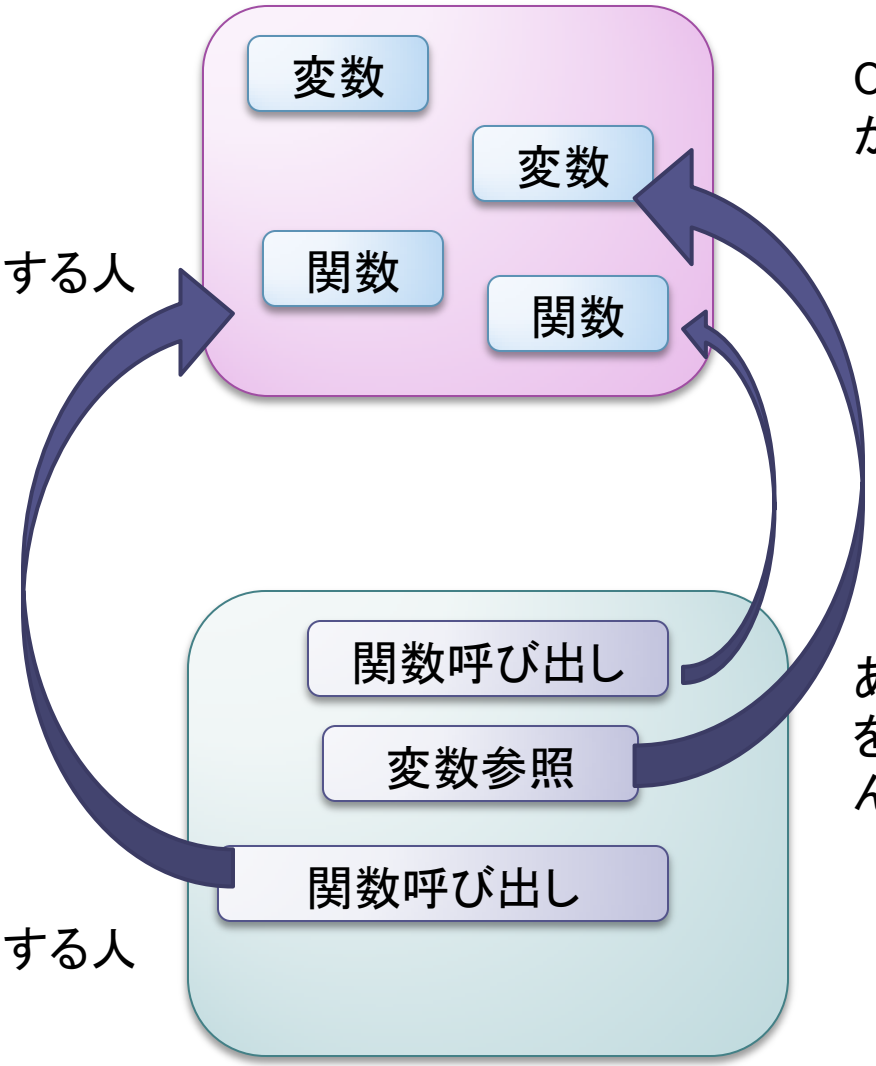
コンポーネントを利用する人
Bob さん

コンポーネントを使う人は、与えられた変数を参照したり、関数を呼び出したりして利用

オブジェクト指向「ではない」シナリオ

コンポーネントを提供する人
Alice さん

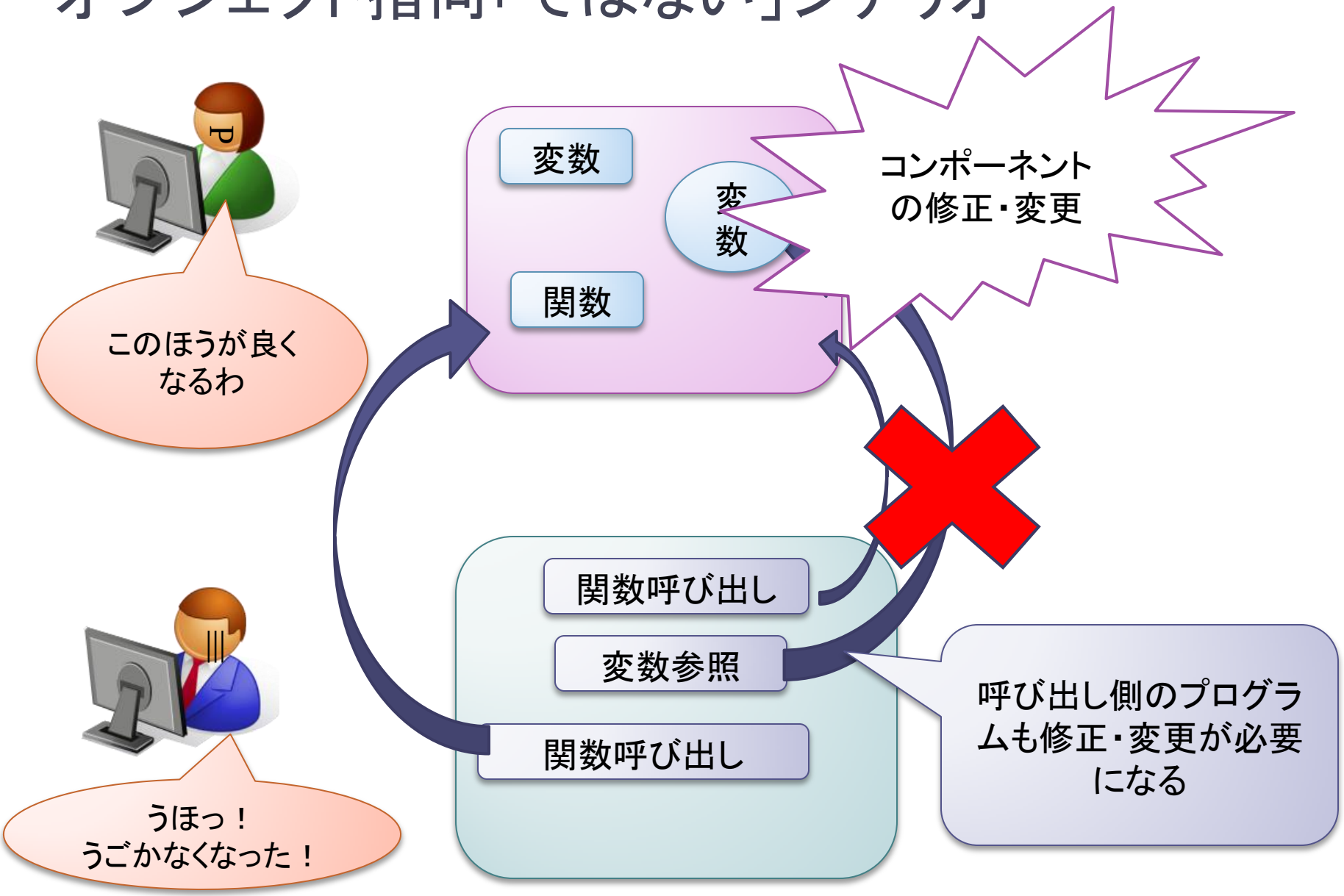
コンポーネントを利用する人
Bob さん



OOPは意識せずに、とにかく機能を実装

あたえられた関数や変数を、便利そうであればどんどん利用

オブジェクト指向「ではない」シナリオ



オブジェクト指向な実現方法

カプセル化した部分は、いつ修正しても平気だわ

private
関数
変数
public
関数
変数

あとで修正が必要そうなオブジェクトの中身はなるべく見せない・触らせない（隠ぺい）

オブジェクトの操作に必要な部分だけを公開＝インターフェース

インターフェースだけ触っていれば安心だ

変数参照
関数呼び出し

インターフェースのみを使ってコンポーネントにアクセス
内部構造は知る必要が無い・知ってはいけない

C++におけるデータ隠ぺい（復習）