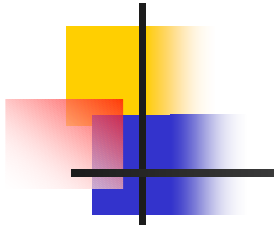
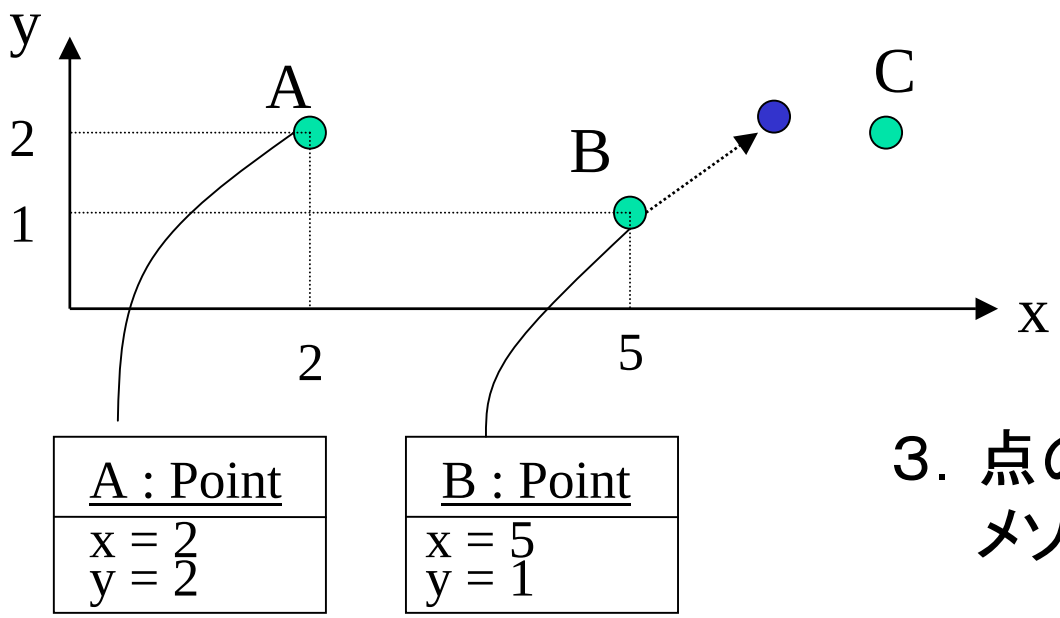


~~プログラムからクラス図へ~~

クラス図（設計）からプログラムへ

図からプログラム表現へ（1）

例：2次元の図形

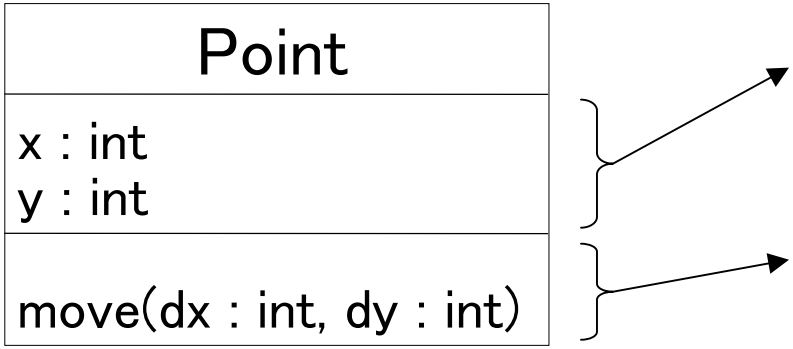


2．新しい点の生成
削除：
コンストラクタ

3．点の移動操作
メソッド

1．属性値（X座標，Y座標）の管理：インスタンス変数

図からプログラム表現へ（2）

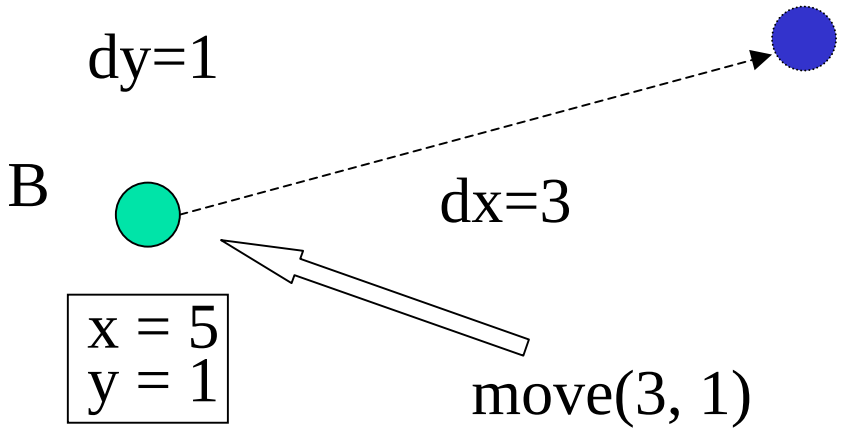



```
class  Point  {

    void  move(int dx, int dy) {
        x = x + dx ;
        y = y + dy ;
    }
}
```

メソッド

図からプログラム表現へ（3）



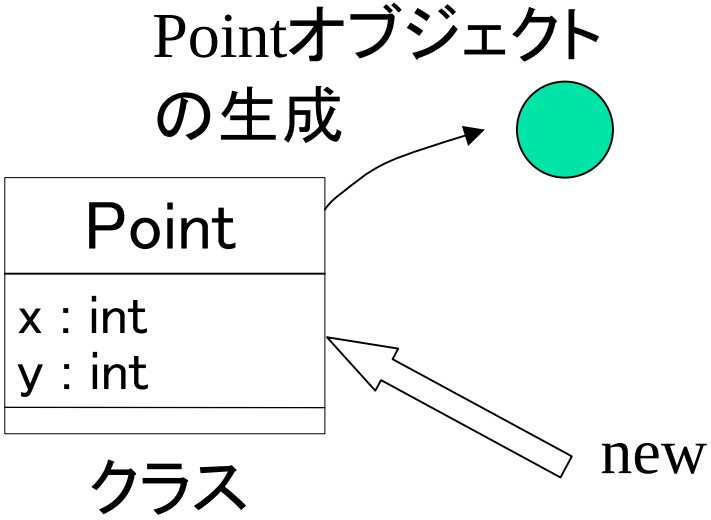
メッセージ通信による計算
メッセージを受け取ると
メソッドが呼ばれる

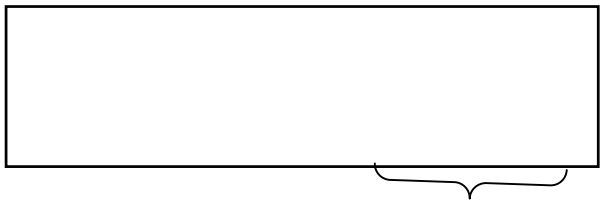
受信オブジェクト
を表す式

メッセージ式

図からプログラム表現へ（4）

オブジェクトの生成





コンストラクタ
(クラス名と同じ名前)
: 特殊なメソッド
戻り値はインスタンス

クラスにnewという特別な
メッセージを送る
クラスオブジェクト

変数ptにPointクラスの
新しいインスタンスが入る

図からプログラム表現へ（5）

x = 3
y = 5

Pointの置く場所
（インスタンス変数の
初期値の設定）

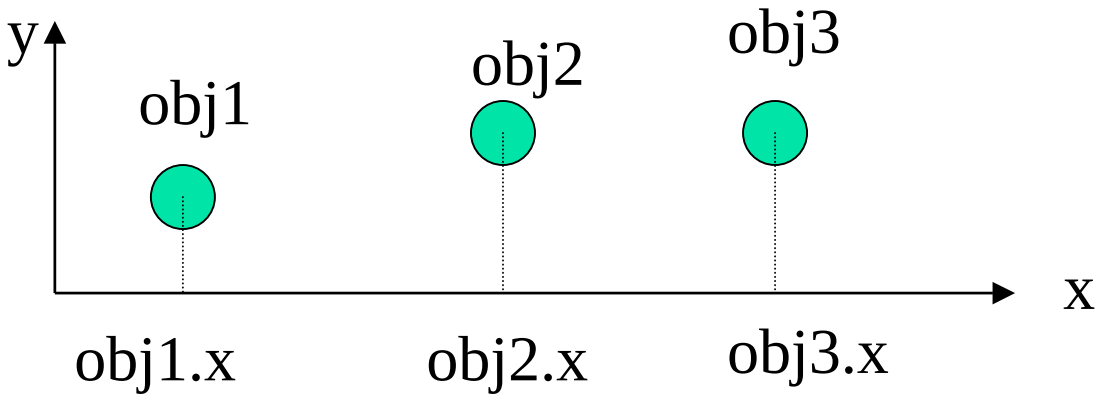
```
class  Point  {
   int x ;
   int y ;
   }
}
```

コンストラクタ
の定義

Point pt = new Point(3, 5)

図からプログラム表現へ（６）

インスタンス変数：インスタンスごとに値が異なる



インスタンス変数の参照

obj.x ： objが表しているオブジェクトのx座標の値

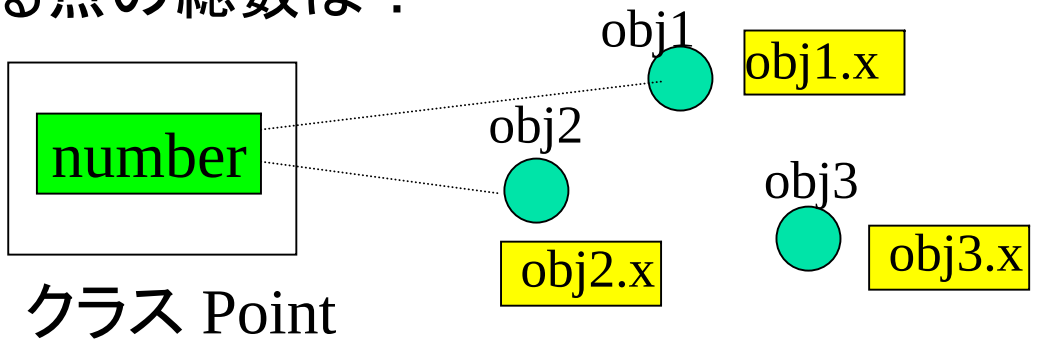
オブジェクトを表す式

インスタンス変数名

図からプログラム表現へ（7）

クラス変数：インスタンスに共通

存在する点の総数は？

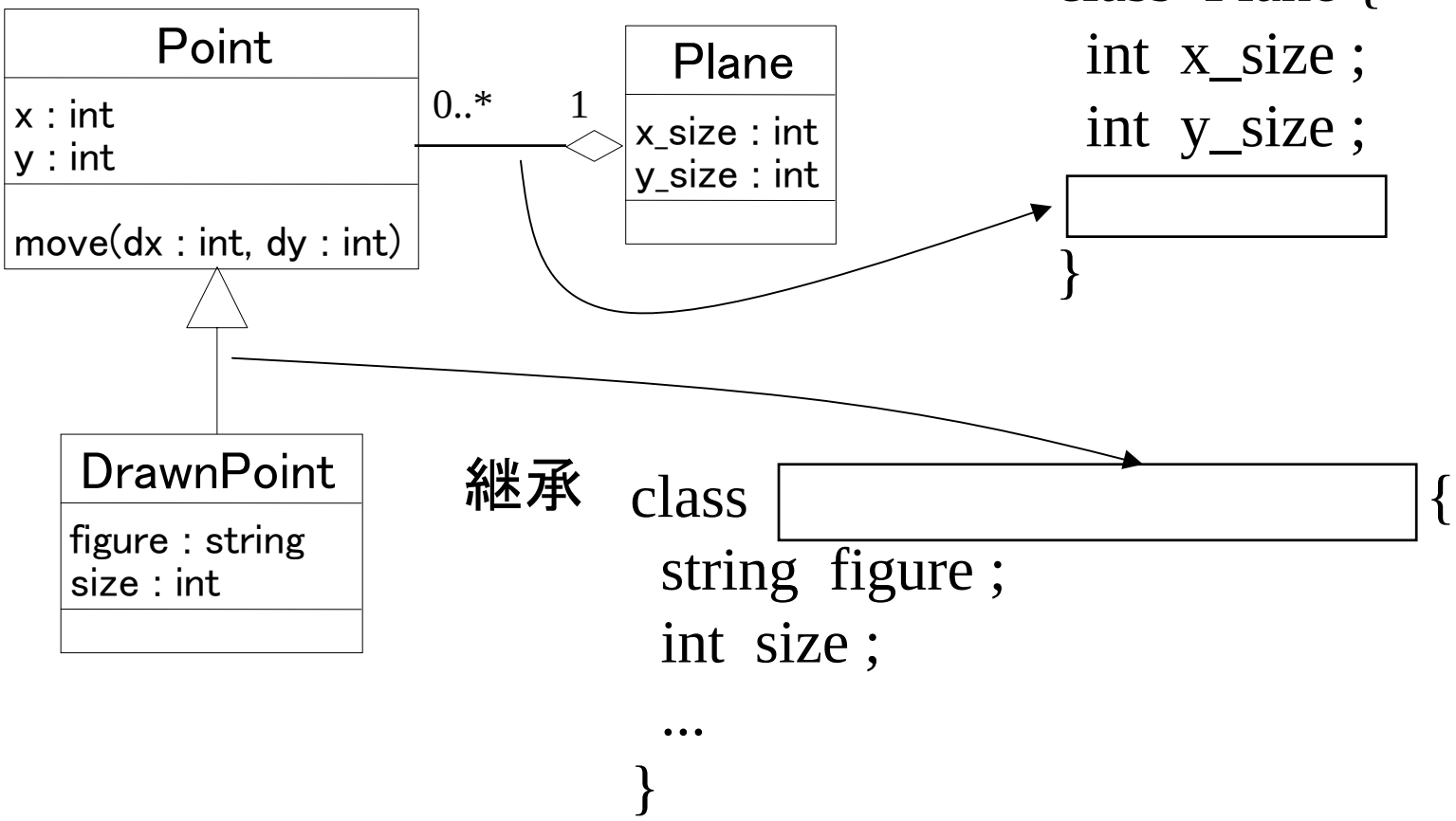



```
class  Point  {
    [                    ]
    int x ;
    int y ;
```

クラス変数

```
Point(int x1, int y1) {
    [                    ]
      x = x1 ;
      y = y1 ;
   }
}
```

図からプログラム表現へ（8）

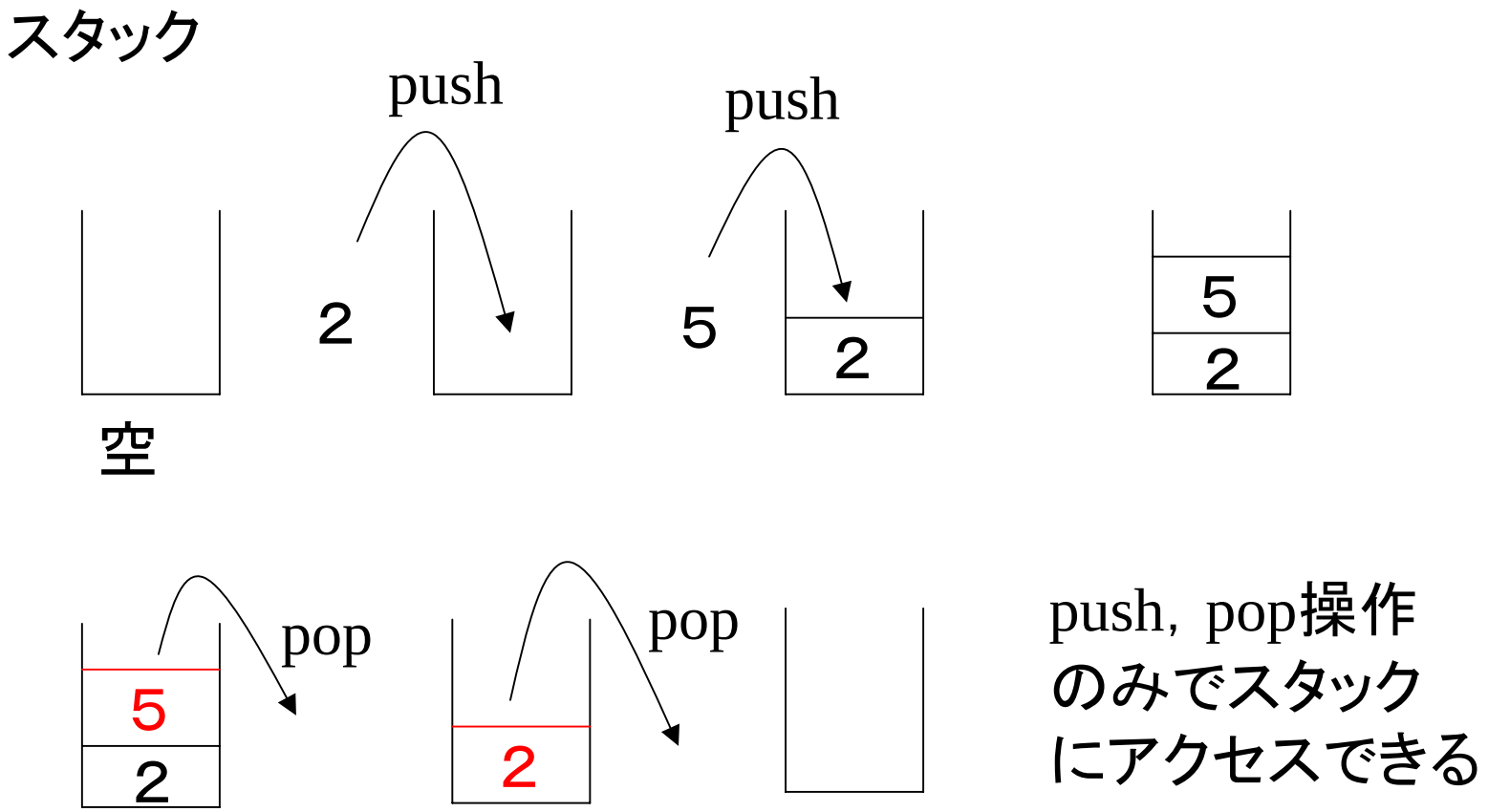


オブジェクト指向のよさ

- データ抽象
- ポリモルフィズム(Polymorphism)
- 継承と関連（集約）に基づく構造化
  - 継承：差分プログラミング
  - 関連：柔軟な変更が可能

  → 再利用性を高める

データ抽象（1）


スタック
push
push
2
5
2
5
2
空
pop
pop
5
2
2
push，pop操作
のみでスタック
にアクセスできる

データ抽象（2）

配列で実現

```
int  Stack[ ]
int  ptr = 0

void push(int  item) {
    [          ]
}
```

```
int  pop( ) {
    if (ptr > 0)
      {
        [          ]
      }
}
```

Stack[5], ptr=ptr+7 など push, popを使わずに直接内部データをアクセスできてしまう

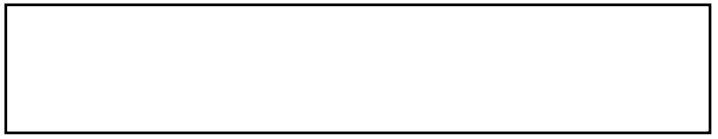
データ抽象（3）

内部データなどデータ構造は重要ではない.
アクセス操作のみが重要. アクセス操作以外を隠蔽

抽象データ型(Abstract Data Type)

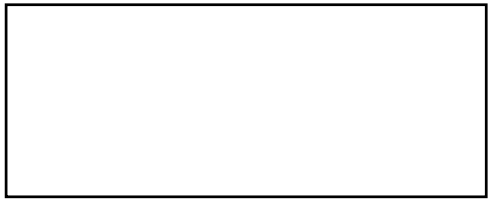
```
class  Stack {
    int  buf[ ] ;
    int  ptr = 0 ;

  public void  push(int item) {
```



```
}
```

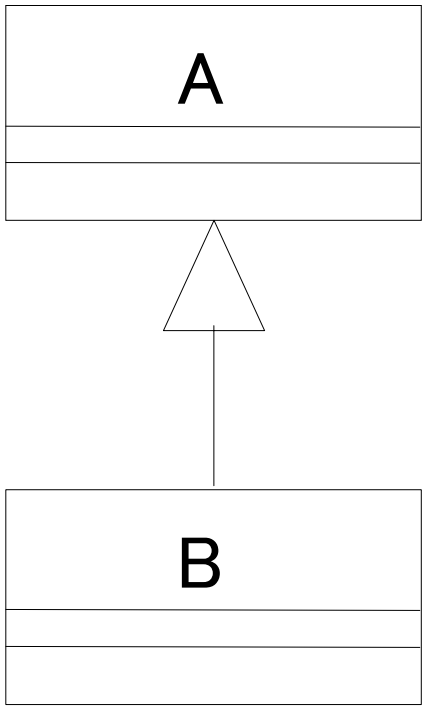
```
  public int pop( ) {
    if (ptr > 0) {
```



```
  }

}
```

可視性

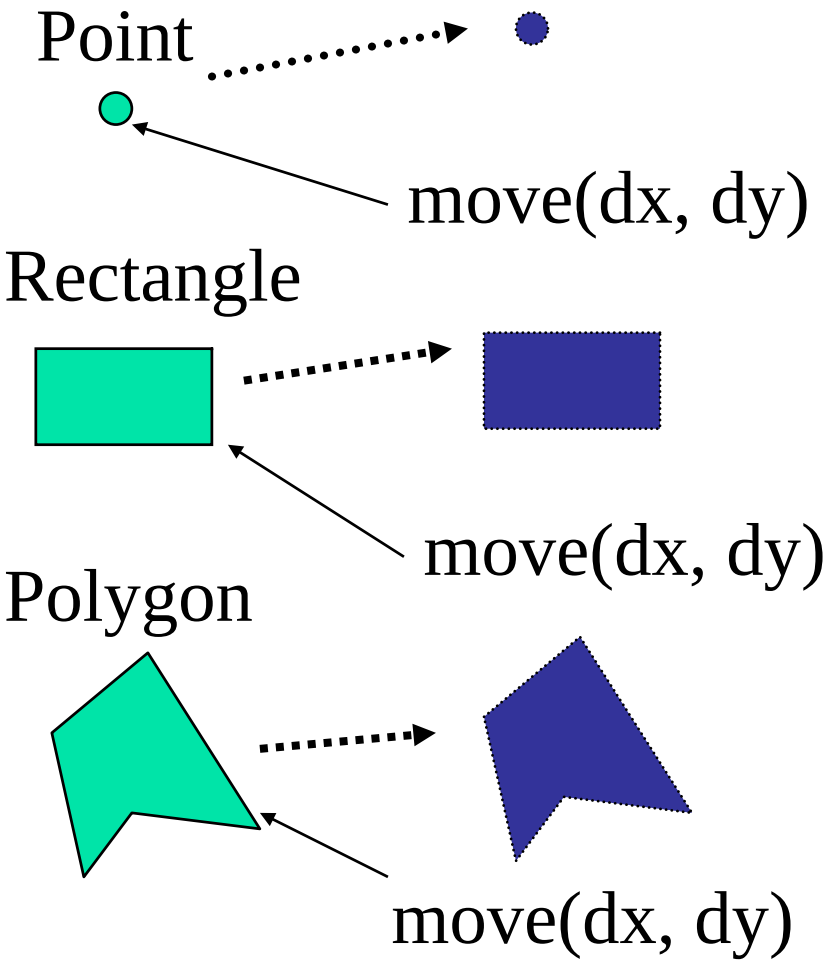


: すべてのオブジェクトから
アクセス可

: そのクラス以外からは
アクセス不可
A内のみアクセス可

: サブクラスからは
アクセス可
A, B内のみアクセス可

ポリモルフィズム（１）



同じmoveで
図形の移動が
できる

中の処理はクラス
ごとに異なる

ポリモルフィズム（2）

```
class  Point  {
   int x ;
   int y ;
    void  move(int dx, int dy) {
   }
}
```

```
class  Polygon  {
   int x[ ] ;
   int y[ ] ;
   int n ;
   void  move(int dx, int dy) {
   }
}
```

メソッドの中の処理が異なる