プログラミング基礎

第9回

参照／文字列の扱い

基本型と参照型

- 型の種類によって
  ・変数を作った時
  ・引数に用いた時
  ・比較の時
  動作が違うので注意

基本型

基本型

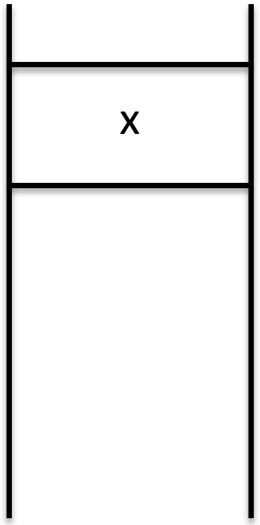
- boolean（trueかfalseを記憶する型）
- char（1文字を記憶する型）
- byte
- short
- int
- long
- float
- double

＊上記の8種類のみ

int x;

プログラムがこの記述を実行した時点で，メモリ上に値を記憶できる領域ができる

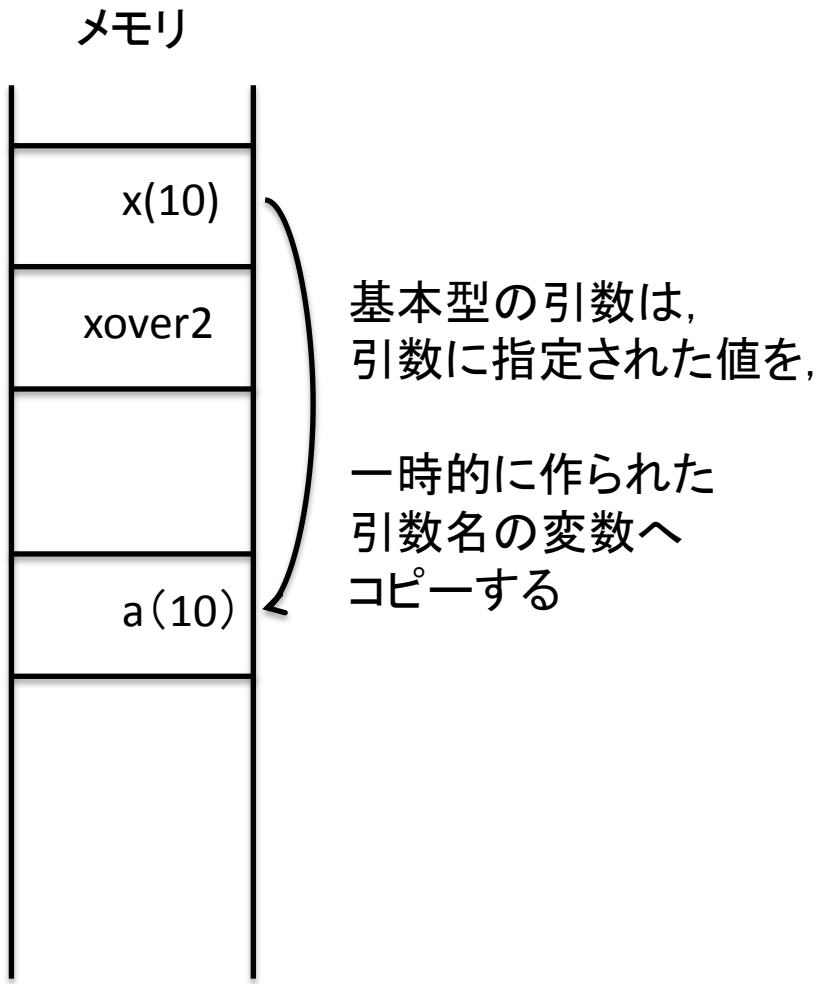
メモリ



基本型と引数

```
static void main(String[] args)
{
   int x;
   x = 10;

   int xover2;
   xover2 = over2(x);
}

int over2(int a)
{
   int r;
   r = a / 2;
   return r;
}
```

プログラムがここへ
到達したとすると

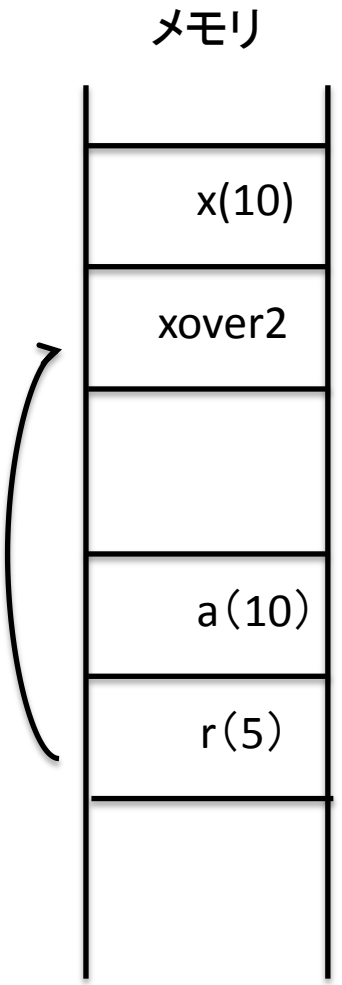


基本型と引数

```
static void main(String[] args)
{
    int x;
    x = 10;

    int xover2;
    xover2 = over2(x);
}

int over2(int a)
{
    int r;
    r = a / 2;
    return r;
}
```

プログラムがover2メソッドを終えたとすると

returnに指定された値が，代入演算子によりxover2へ

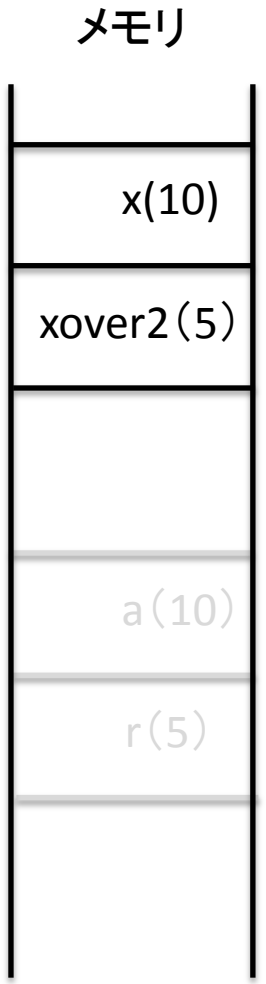


基本型と引数

```
static void main(String[] args)
{
    int x;
    x = 10;

    int xover2;
    xover2 = over2(x);
}

int over2(int a)
{
    int r;
    r = a / 2;
    return r;
}
```

プログラムが
代入演算子
の処理を終えた
とすると

メソッドのために
作られた変数は
自動的に消滅する



基本型と比較

```
int x, y;
x = 10;
y = 10;

boolean b;
b = x == y;
```

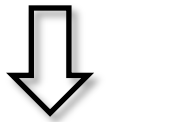
bはtrue（等しい）となる
⇒変数x，yの値が同じか比較される

参照型

参照型

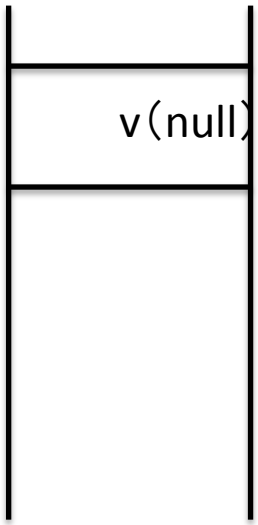
- 基本型以外の型
  （クラス，配列，インタフェース）

Vector2D v;



プログラムが
この記述を
実行した時点では，
まだ値を記憶できない

メモリ

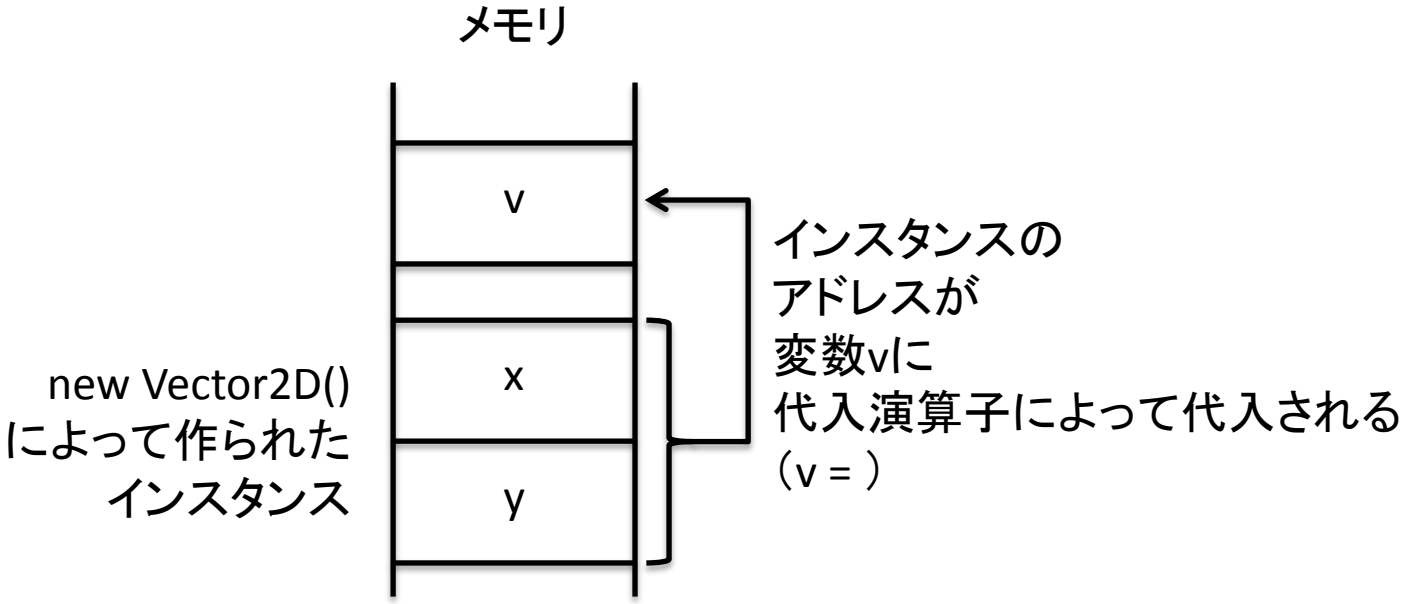


では何ができたのか？

newによって作られた
インスタンスがメモリ上の
どこにできたか（=**アドレス**）
を記憶するための領域ができた
（この時点でvはnull）
（**null**：メモリのどこも
指していないことを表す）

参照型

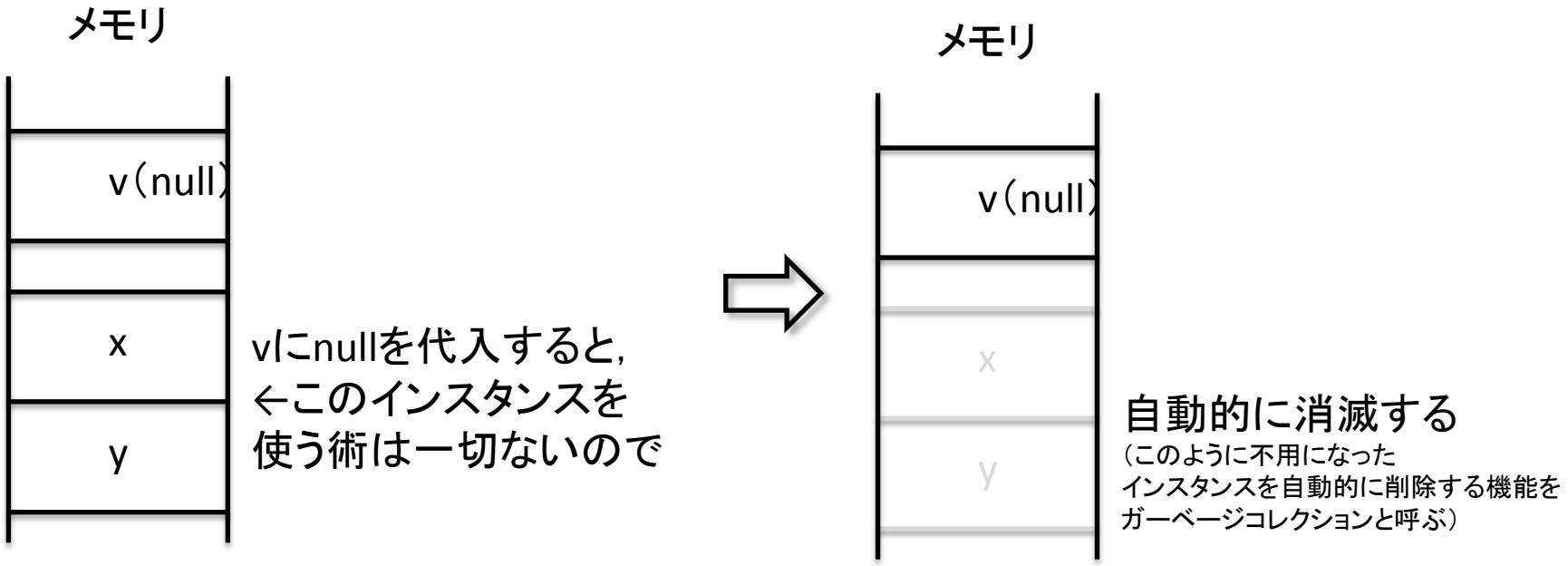
```
Vector2D v;
v = new Vector2D();
```



結果，変数vのアドレスを経由してインスタンスを使用できる
（例：v.x = 1.0; v.x = v.x * 2;）

参照型

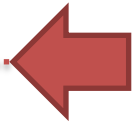
```
Vector2D v;
v = new Vector2D();
v = null;
```



演習

- nullを実際に使用し，プログラムの変化を確かめよ．
（プログラム名：NullTest，mainメソッドがそのまま動くようにVector2Dクラスは各自作成すること）

```
public class NullTest
{
   static public void main(String[] args)
   {
      Vector2D v;
      v = new Vector2D(1.0. 1.0);
      ..........................
      double x, y;
      x = v.getX();
      y = v.getY();
      System.out.println("v=(" + x + ", " + y + ")");
   }
}
```

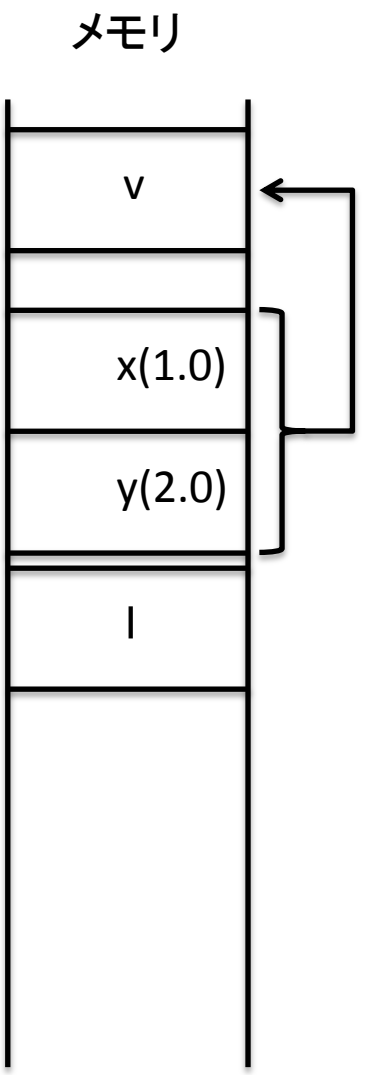


mainメソッドが正しく動いたことを確認した後，
ここに v = null; と記述してみよ

参照型と引数

```
static void main(String[] args)
{
    Vector2D v;
    v = new Vector2D();
    v.x = 1.0; v.y = 2.0;

    double l;
    l = getLength(v);
}

double getLength(Vector2D src)
{
    double l;
    l = Math.sqrt(src.x * src.x + src.y * src.y);
    return l;
}
```

プログラムがここへ
到達したとすると



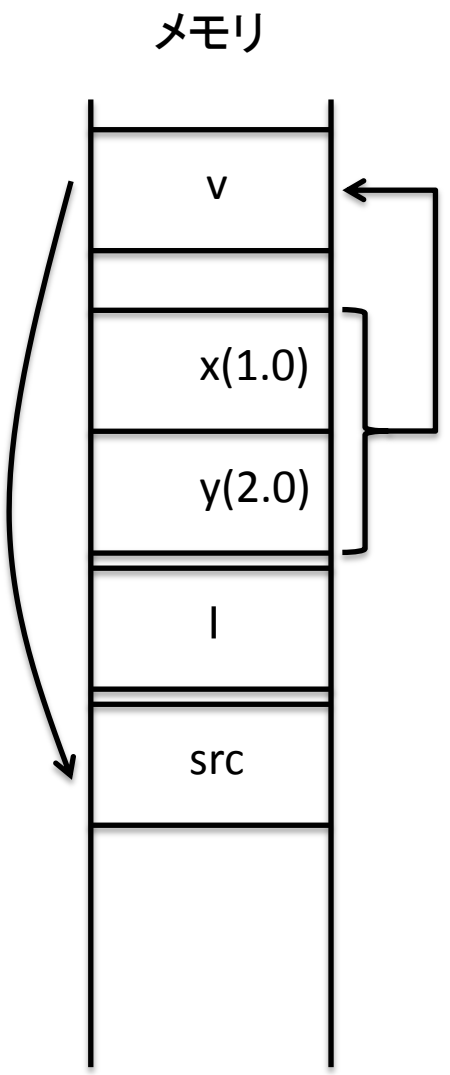
参照型と引数

```
static void main(String[] args)
{
   Vector2D v;
   v = new Vector2D();
   v.x = 1.0; v.y = 2.0;

   double l;
   l = getLength(v);
}

double getLength(Vector2D src)
{
   double l;
   l = Math.sqrt(src.x * src.x + src.y * src.y);
   return l;
}
```

プログラムがここへ
到達したとすると

参照型の引数は
アドレスを

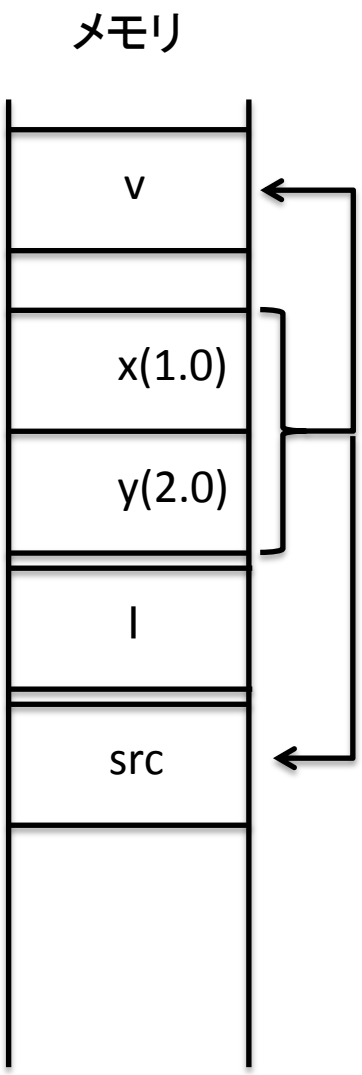
一時的に作られた
引数名の変数へ
コピーする



参照型と引数

```
static void main(String[] args)
{
    Vector2D v;
    v = new Vector2D();
    v.x = 1.0; v.y = 2.0;

    double l;
    l = getLength(v);
}

double getLength(Vector2D src)
{
    double l;
    l = Math.sqrt(src.x * src.x + src.y * src.y);
    return l;
}
```

プログラムがここへ到達したとすると

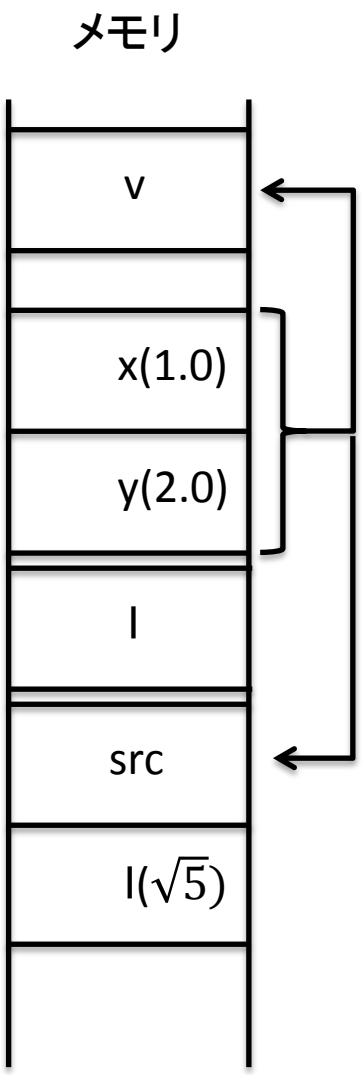


結果vとsrcは同じインスタンスを指す

参照型と引数

```
static void main(String[] args)
{
    Vector2D v;
    v = new Vector2D();
    v.x = 1.0; v.y = 2.0;

    double l;
    l = getLength(v);
}

double getLength(Vector2D src)
{
    double l;
    l = Math.sqrt(src.x * src.x + src.y * src.y);
    return l;
}
```

プログラムがここへ
到達したとすると

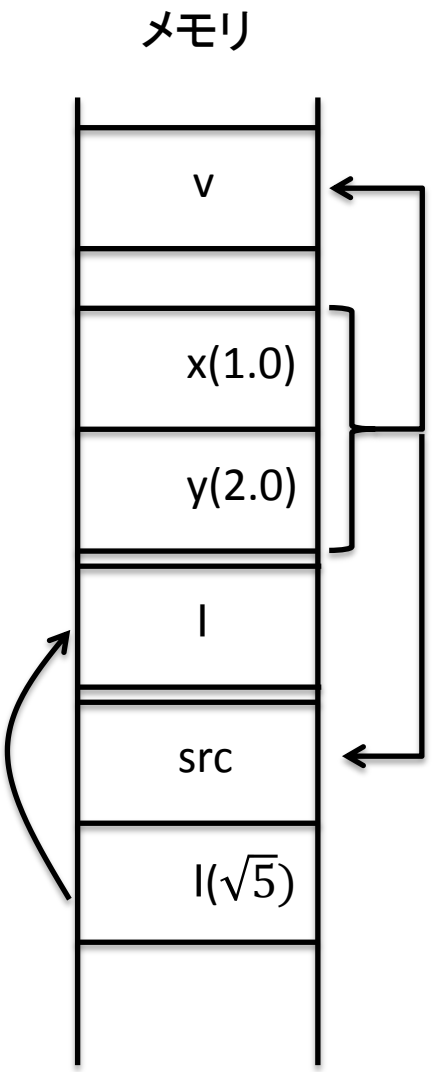


参照型と引数

```
static void main(String[] args)
{
    Vector2D v;
    v = new Vector2D();
    v.x = 1.0; v.y = 2.0;

    double l;
    l = getLength(v);
}

double getLength(Vector2D src)
{
    double l;
    l = Math.sqrt(src.x * src.x + src.y * src.y);
    return l;
}
```

プログラムがgetLength
メソッドを終えたとすると

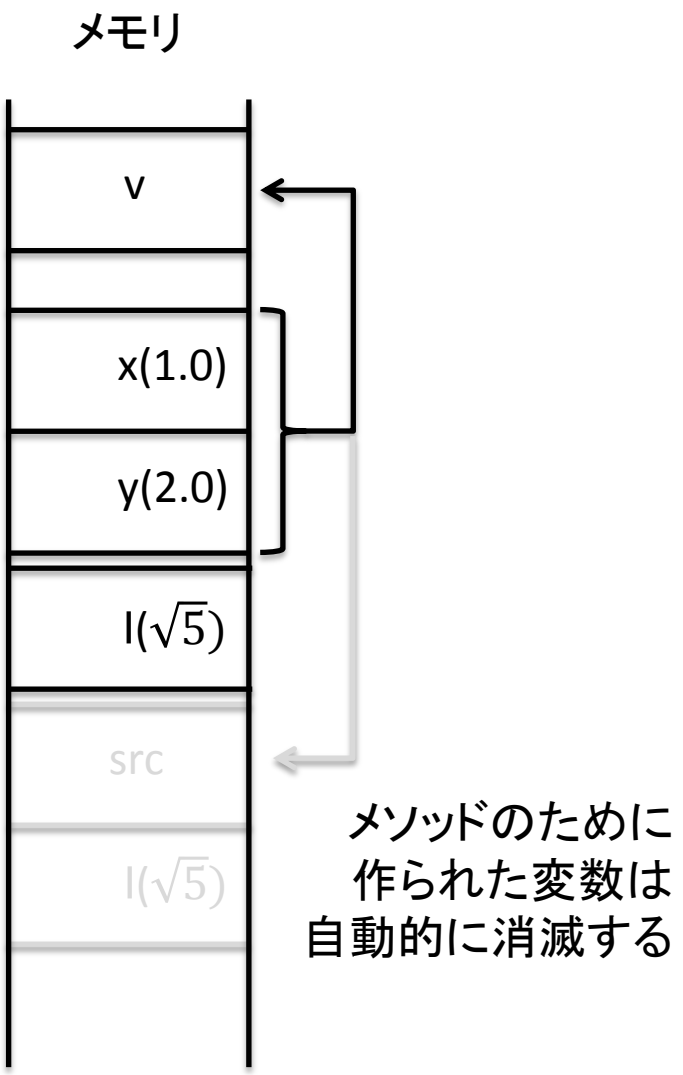
returnに指定された
値が，代入演算子
によりmainメソッド内のlへ



参照型と引数

```
static void main(String[] args)
{
    Vector2D v;
    v = new Vector2D();
    v.x = 1.0; v.y = 2.0;

    double l;
    l = getLength(v);
}

double getLength(Vector2D src)
{
    double l;
    l = Math.sqrt(src.x * src.x + src.y * src.y);
    return l;
}
```

代入演算子の
処理を終えたとすると

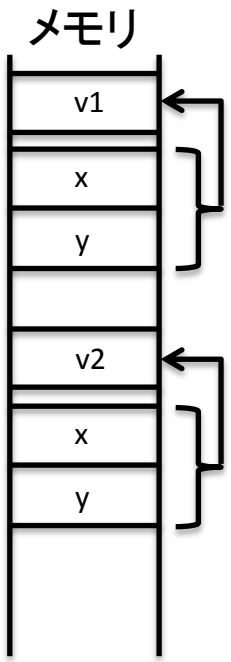


メソッドのために
作られた変数は
自動的に消滅する

参照型と比較

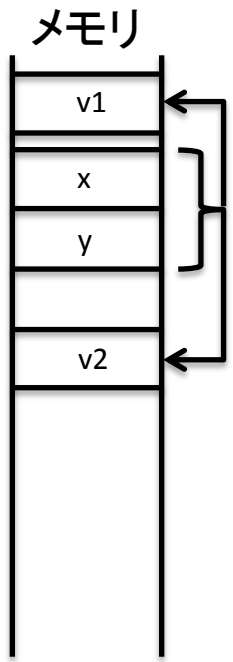
参照型の比較は「アドレスが同じか」を見る

```
Vector2D v1, v2;
v1 = new Vector2D();
v2 = new Vector2D();

v1.x = 1; v1.y = 1;
v2.x = 1; v2.y = 1;

boolean b;
b = v1 == v2;
```



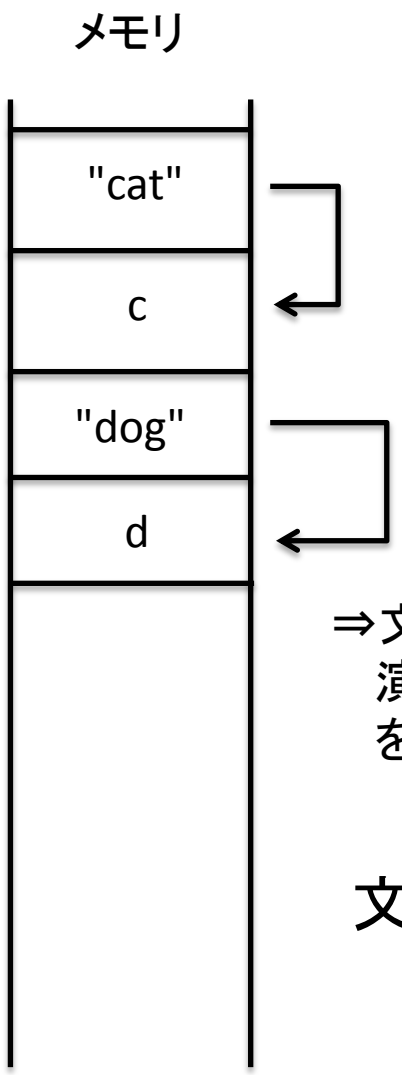
bはfalse（等しくない）という結果になる

```
Vector2D v1, v2;
v1 = new Vector2D();
v1 = v2;

boolean b;
b = v1 == v2;
```



bはtrue（等しい）という結果になる

文字列



String c = "cat";
String d = "dog";

＊文字列も参照型（Stringクラス）であるため，
変数c，dは文字列そのものではなく
文字列へのアドレスであることに注意

⇒文字列の内容を比較する場合は
演算子ではなくequals，もしくはcopareToメソッド
を用いる

文字列（Stringクラス）の各種操作は
Web資料を参照