プログラミング基礎

第5回　クラスを用いたプログラム

9月26日の問題を振り返る

- 「二つの2次元ベクトルを加算減算して画面に結果を表示しなさい」

```
static public void main(String[] args)
{
        double x0 = Double.parseDouble(args[0]);
        double y0 = Double.parseDouble(args[1]);
        double x1 = Double.parseDouble(args[2]);
        double y1 = Double.parseDouble(args[3]);

        double addx = x0 + x1;
        double addy = y0 + y1;
        double subx = x0 - x1;
        double suby = y0 - y1;

        System.out.println("Add:(" + addx + ", " + addy + ")");
        System.out.println("Sub:(" + subx + ", " + suby + ")");
}
```

9月26日の問題を振り返る

- 2次元ベクトルは必ずxとyがセットになったもの
  ⇒いちいちdouble型で別々に扱うのは煩雑

ベクトルV0
```
double x0 = Double.parseDouble(args[0]);
double y0 = Double.parseDouble(args[1]);
```
ベクトルV1
```
double x1 = Double.parseDouble(args[2]);
double y1 = Double.parseDouble(args[3]);
```

- しかし，double型ではどうやっても1つの値しか記憶できない

さてどうしよう

- 今困っていること:
  double型は1つの値しか記憶できない

⇒2つの値を記憶できる型
（仮に**ベクトル型**と名付けておく）
を自分で作れれば問題解決

ベクトル型導入

```
static public void main(String[] args)
{
        ベクトル v0, v1;
        v0のx = Double.parseDouble(args[0]);
        v0のy = Double.parseDouble(args[1]);
        v1のx = Double.parseDouble(args[2]);
        v1のy = Double.parseDouble(args[3]);

        double addx = v0のx + v1のx;
        double addy = v0のy + v1のy;
        double subx = v0のx - v1のx;
        double suby = v0のy - v1のy;

        System.out.println("Add:(" + addx + ", " + addy + ")");
        System.out.println("Sub:(" + subx + ", " + suby + ")");
}
```

というわけで

⇒「必要になった型（クラス）を自分で
どうやって作るのか，使うのか」
が今後数回の話題

Java（というよりは最近のプログラミング言語全般）を使いこなすためには
必ず理解しなければならない話題

実際に型（クラス）を作ってみる

「これから型（クラス）を
作り（定義し）ますよ」
という意味で必ず書く

```
class Vector2D
{
    double x;
    double y;
}
```

型（クラス）名
⇒任意
分かりやすい名前，
もしくは指示された名前を

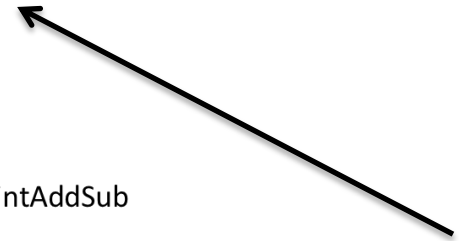
Vector2D型（クラス）は，
double型の変数x，
double型の変数y
を持つ
という理解で構わない
x，yと名付けたが
もちろん違う名前でも構わない

Vector2D型（クラス）
の内容を{}内に記述する

⇒クラスが持つ（に含まれる）変数
⇒「**属性**」「**フィールド**」と呼ばれる

作った型クラスはどこに置く

```
class Vector2D
{
    double x;
    double y;
}

public class VectorsPrintAddSub
{
        static public void main(String[] args)
        {
            …
        }
}
```

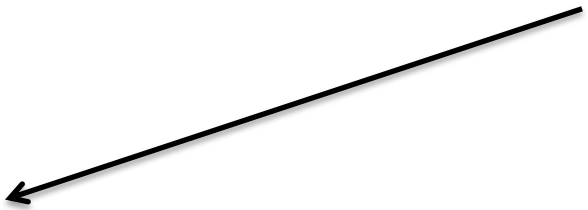


mainメソッドを含んでいるクラス
（⇒実はいままでクラスを散々作ってた）
の外ならどこでも可

作った型クラスはどこに置く

```
public class VectorsPrintAddSub
{
        static public void main(String[] args)
        {
            …
        }
}

class Vector2D
{
    double x;
    double y;
}
```

mainメソッドを含んでいるクラス
（⇒実はいままでクラスを散々作ってた）
の外ならどこでも可



演習

- 9月26日の課題，VectorsAddSubを改良していく
- Vector2Dクラスを作成せよ

ベクトル型導入

```
static public void main(String[] args)
{
        ベクトル v0, v1;
        v0のx = Double.parseDouble(args[0]);
        v0のy = Double.parseDouble(args[1]);
        v1のx = Double.parseDouble(args[2]);
        v1のy = Double.parseDouble(args[3]);

        double addx = v0のx + v1のx;
        double addy = v0のy + v1のy;
        double subx = v0のx - v1のx;
        double suby = v0のy - v1のy;

        System.out.println("Add:(" + addx + ", " + addy + ")");
        System.out.println("Sub:(" + subx + ", " + suby + ")");
}
```

作ったクラスを使う

```
static public void main(String[] args)
{
        Vector2D v0, v1;
        v0のx = Double.parseDouble(args[0]);
        v0のy = Double.parseDouble(args[1]);
        v1のx = Double.parseDouble(args[2]);
        v1のy = Double.parseDouble(args[3]);

        double addx = v0のx + v1のx;
        double addy = v0のy + v1のy;
        double subx = v0のx - v1のx;
        double suby = v0のy - v1のy;

        System.out.println("Add:(" + addx + ", " + addy + ")");
        System.out.println("Sub:(" + subx + ", " + suby + ")");
}
```

まだ不十分

作ったクラスを使う

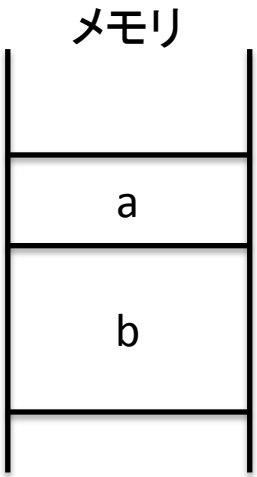
```
static public void main(String[] args)
{
        Vector2D v0, v1;
        v0 = new Vector2D();
        v1 = new Vector2D();
        v0のx = Double.parseDouble(args[0]);
        v0のy = Double.parseDouble(args[1]);
        v1のx = Double.parseDouble(args[2]);
        v1のy = Double.parseDouble(args[3]);

        double addx = v0のx + v1のx;
        double addy = v0のy + v1のy;
        double subx = v0のx - v1のx;
        double suby = v0のy - v1のy;

        System.out.println("Add:(" + addx + ", " + addy + ")");
        System.out.println("Sub:(" + subx + ", " + suby + ")");
}
```

作ったクラスの変数を作る場合は，このように書く

インスタンス

intやdoubleは以下のように記述するだけで自動的にメモリ上に値を記憶する領域ができる（確保される）

例：
int a;
double b;



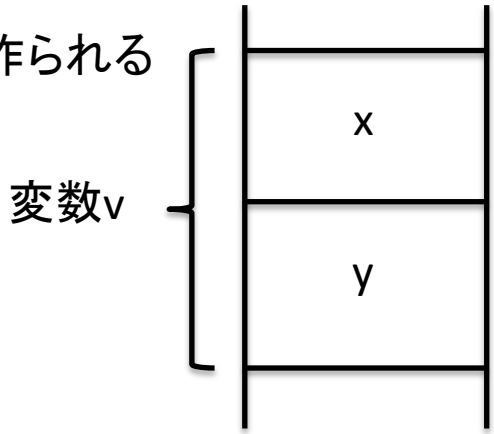
クラスは同じように書いても値を記憶する領域はできない

例：
Vector2D v0;

以下のように記述してはじめて**値を記憶する領域**が作られる

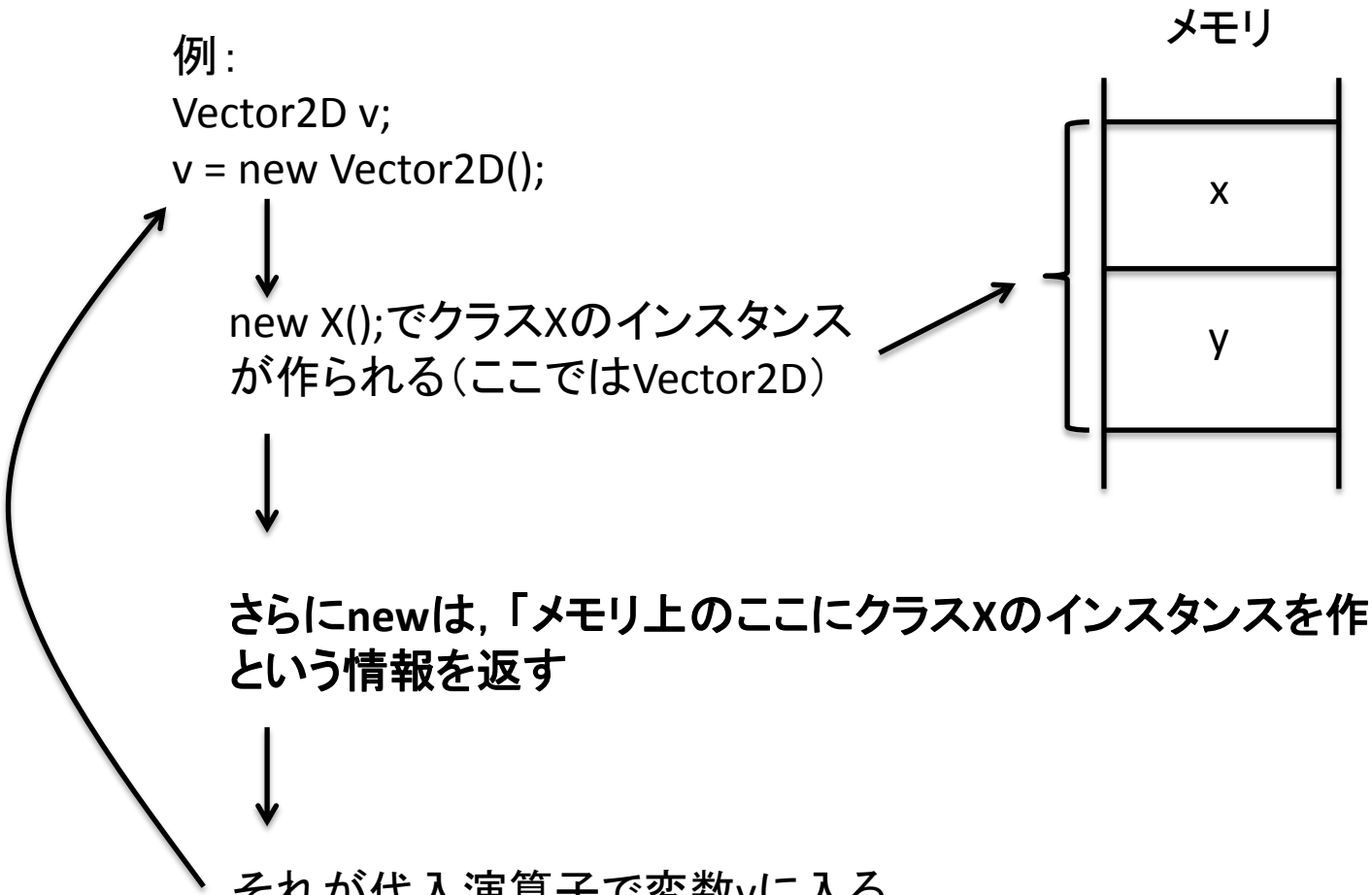
（⇒これを**インスタンス**と呼ぶ）

例：
Vector2D v;
v = new Vector2D();



newの役割

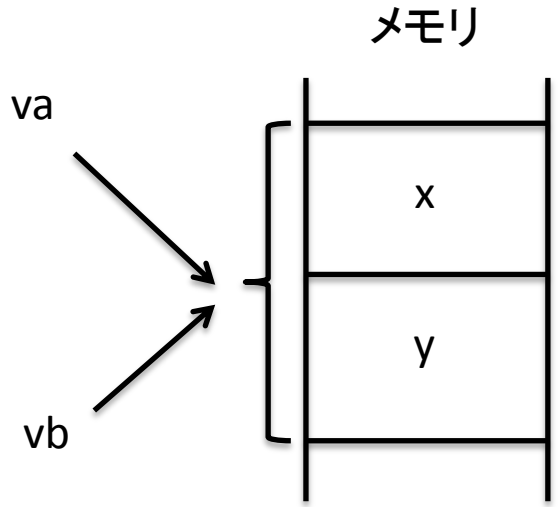
例：
Vector2D v;
v = new Vector2D();



newの役割

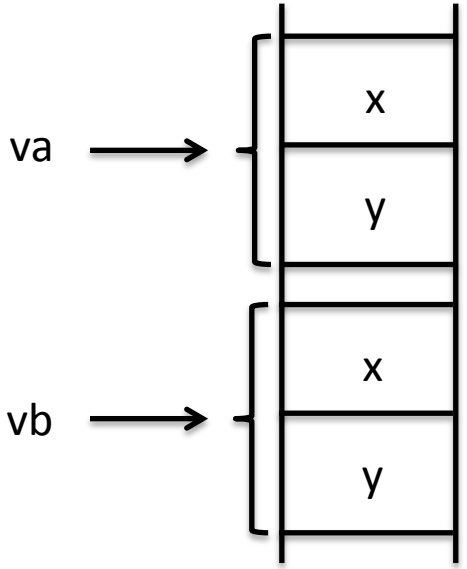
よって，右の例では，
vaとvbは同じもの，
ということになる
（結局使える
ベクトルは1つ）

例：
Vector2D va, vb;
va = new Vector2D();
vb = va;



この例では，２つのVector2D
インスタンスが作られる．
ベクトルを2つ使えるようになる
ということ

例：
Vector2D va, vb;
va = new Vector2D();
vb = new Vector2D();



属性（フィールド）の使用

```
static public void main(String[] args)
{
        Vector2D v0, v1;
        v0 = new Vector2D();
        v1 = new Vector2D();
        v0のx = Double.parseDouble(args[0]);
        v0のy = Double.parseDouble(args[1]);
        v1のx = Double.parseDouble(args[2]);
        v1のy = Double.parseDouble(args[3]);

        double addx = v0のx + v1のx;
        double addy = v0のy + v1のy;
        double subx = v0のx - v1のx;
        double suby = v0のy - v1のy;

        System.out.println("Add:(" + addx + ", " + addy + ")");
        System.out.println("Sub:(" + subx + ", " + suby + ")");
}
```

属性（フィールド）の使用

```
static public void main(String[] args)
{
        Vector2D v0, v1;
        v0 = new Vector2D();
        v1 = new Vector2D();
        v0.x = Double.parseDouble(args[0]);
        v0.y = Double.parseDouble(args[1]);
        v1.x = Double.parseDouble(args[2]);
        v1.y = Double.parseDouble(args[3]);

        double addx = v0.x + v1.x;
        double addy = v0.y + v1.y;
        double subx = v0.x - v1.x;
        double suby = v0.y - v1.y;

        System.out.println("Add:(" + addx + ", " + addy + ")");
        System.out.println("Sub:(" + subx + ", " + suby + ")");
}
```

演習

- 9月26日の課題，VectorsAddSubを改良していく
- Vector2Dクラスを利用するように改良せよ

メソッドの作成，利用

```
static public void main(String[] args)
{
        Vector2D v0, v1;
        v0 = new Vector2D();
        v1 = new Vector2D();
        v0.x = Double.parseDouble(args[0]);
        v0.y = Double.parseDouble(args[1]);
        v1.x = Double.parseDouble(args[2]);
        v1.y = Double.parseDouble(args[3]);

        double addx = v0.x + v1.x;
        double addy = v0.y + v1.y;
        double subx = v0.x - v1.x;
        double suby = v0.y - v1.y;

        System.out.println("Add:(" + addx + ", " + addy + ")");
        System.out.println("Sub:(" + subx + ", " + suby + ")");
}
```

x，y各属性にいちいち
値を入れるのも煩雑
1行でできないか？

メソッドの作成，利用

```
static public void main(String[] args)
{
        Vector2D v0, v1;
        v0 = new Vector2D();
        v1 = new Vector2D();
        v0のxyはDouble.parseDouble(args[0])とDouble.parseDouble(args[1]);
        v1のxyはDouble.parseDouble(args[2])とDouble.parseDouble(args[3]);

        double addx = v0.x + v1.x;
        double addy = v0.y + v1.y;
        double subx = v0.x - v1.x;
        double suby = v0.y - v1.y;

        System.out.println("Add:(" + addx + ", " + addy + ")");
        System.out.println("Sub:(" + subx + ", " + suby + ")");
}
```

メソッドの作成，利用

そこで，Vector2Dクラスに，x，yに値を設定するメソッドを作る

```
class Vector2D
{
    double x;
    double y;

    void SetValue(double newx, double newy)
    {
        x = newx;
        y = newy;
    }
}
```

メソッドの作成，利用

```
static public void main(String[] args)
{
        Vector2D v0, v1;
        v0 = new Vector2D();
        v1 = new Vector2D();
        v0のxyはDouble.parseDouble(args[0])とDouble.parseDouble(args[1]);
        v1のxyはDouble.parseDouble(args[2])とDouble.parseDouble(args[3]);

        double addx = v0.x + v1.x;
        double addy = v0.y + v1.y;
        double subx = v0.x - v1.x;
        double suby = v0.y - v1.y;

        System.out.println("Add:(" + addx + ", " + addy + ")");
        System.out.println("Sub:(" + subx + ", " + suby + ")");
}
```

メソッドの作成，利用

```
static public void main(String[] args)
{
        Vector2D v0, v1;
        v0 = new Vector2D();
        v1 = new Vector2D();
        v0.SetValue(Double.parseDouble(args[0]),Double.parseDouble(args[1]));
        v1.SetValue(Double.parseDouble(args[2]),Double.parseDouble(args[3]));

        double addx = v0.x + v1.x;
        double addy = v0.y + v1.y;
        double subx = v0.x - v1.x;
        double suby = v0.y - v1.y;

        System.out.println("Add:(" + addx + ", " + addy + ")");
        System.out.println("Sub:(" + subx + ", " + suby + ")");
}
```

メソッドの作成，利用

属性（フィールド）に対して，直接値を代入したり，読み出すことはあまり好ましくない

そこで，Vector2Dクラスに，x，yの値を得るメソッドを作る

```
class Vector2D
{
   double x;
   double y;

   double GetX()
   {
      return x;
   }

   double GetY()
   {
      return y;
   }

   void SetValue(double newx, double newy)
   {
      x = newx;
      y = newy;
   }
}
```

メソッドの作成，利用

```
static public void main(String[] args)
{
        Vector2D v0, v1;
        v0 = new Vector2D();
        v1 = new Vector2D();
        v0.SetValue(Double.parseDouble(args[0]),Double.parseDouble(args[1]));
        v1.SetValue(Double.parseDouble(args[2]),Double.parseDouble(args[3]));

        double addx = v0.x + v1.x;
        double addy = v0.y + v1.y;
        double subx = v0.x - v1.x;
        double suby = v0.y - v1.y;

        System.out.println("Add:(" + addx + ", " + addy + ")");
        System.out.println("Sub:(" + subx + ", " + suby + ")");
}
```

メソッドの作成，利用

```
static public void main(String[] args)
{
        Vector2D v0, v1;
        v0 = new Vector2D();
        v1 = new Vector2D();
        v0.SetValue(Double.parseDouble(args[0]),Double.parseDouble(args[1]));
        v1.SetValue(Double.parseDouble(args[2]),Double.parseDouble(args[3]));

        double addx = v0.GetX() + v1.GetX();
        double addy = v0.GetY() + v1.GetY();
        double subx = v0.GetX() - v1.GetX();
        double suby = v0.GetY() - v1.GetY();

        System.out.println("Add:(" + addx + ", " + addy + ")");
        System.out.println("Sub:(" + subx + ", " + suby + ")");
}
```

演習

- 9月26日の課題，VectorsAddSubを改良していく
- Vector2DクラスにSetValueメソッド，GetXメソッド,GetYメソッドを作成し，利用するように改良せよ

演習

- 9月26日の課題，VectorsAddSubを改良していく
- 足し算，引き算の結果もVector2D型（クラス）の変数に記憶するように改良せよ

演習

- 9月26日の課題，VectorsAddSubを改良していく
- Vector2Dクラスの属性x，yを
  （x, y）
  という形で表示するVector2D.Printメソッドを作り，利用せよ

- 例：
  Vector2D v;
  V = new Vector2D();
  v.SetValue(0.0, 1.0);
  v.Print();

  実行結果
  (0.0, 1.0)

演習（Advanced）

- 9月26日の課題，VectorsAddSubを改良していく
- Vector2Dクラスの属性x，yを
  （x, y）
  という形の文字列で返すVector2D.ToStringメソッドを作り，利用せよ

- 例：
  Vector2D v;
  V = new Vector2D();
  v.SetValue(0.0, 1.0);
  System.out.println(v.ToString());

  実行結果
  (0.0, 1.0)