Javaプログラミング

第8回　色の設定とグラフの描画

担当：劉　雪峰

復習と今回の内容

- 前回の内容
  - Javaのフレームの雛形
  - Javaで絵を描く

- 今回の内容
  - Javaで色を設定する
  - 数学のグラフを描く

Javaでの色の設定

Javaで色を設定する方法を学びます.

Graphics型をgとし，以下のように色を設定します。

```
g.setColor(色);
```

ただし、色はColorクラスで設定されているものです。

Color.red，Color.blue，Color.green,
Color.white, Color.pink など

Javaでの色の設定(実験)

```
public void paintComponent(Graphics g) {
    g.setColor(Color.red);
    g.drawLine(100,100,200,200);
    g.setColor(Color.blue);
    g.drawLine(100,200,200,300);
    g.drawLine(100,300,200,400);
}
```

Javaでの色の自由な設定

- もっと自由に色を設定するために，三原色を組み合わせて色を生成します.
- 三原色とはRGBということです。
  - Red
  - Green
  - Blue

  これらの強度を0~255までの整数で与える.

Javaでの色の自由な設定

- Graphics型をgとすると

```
g.setColor(new Color(255, 0, 0));  //赤色
g.fillRect(100,100,100,100);

g.setColor(new Color(0,0,0));    //黒色
g.fillRect(200,100,100,100);

g.setColor(new Color(128,128,128));   //グレー
g.fillRect(300,100,100,100);
```

演習１

５個の円盤を描画しなさい。

1. 円盤の中心座標を乱数で生成すること。(500×500の範囲に越えないように)
2. 円盤の半径を50~100の間の乱数とすること。
3. 円盤を塗りつぶす時、色は乱数で生成すること。

ヒント

1. Math.random()は区間[0,1)に属する乱数を生成する。
2. 整数Minと Maxの間の整数乱数の生成:

```
Min + (int)(Math.random() * ((Max - Min) + 1))
```

3. 色のR,G,B成分の範囲は[0,255]であるので、以下の式で色を乱数で生成できる。

```
Color rand_color = new Color( (int)(Math.random()*256),  (int)(Math.random()*256),
                              (int)(Math.random()*256) );
```

ヒント

```
public void paintComponent(Graphics g) {
  int i,  rand_radius, rand_x, rand_y;
  Color rand_color;
  for (i=0;i<5;i++) {
      rand_color = ???;
      rand_radius = ???;
      rand_x = ???;   rand_y = ???;
      g.setColor(rand_color);
      g.fillOval (??, ??, ??, ??);
  }
}
```

テーマ2：グラフを描こう

- $y=x^2$ のグラフを描いてもらいます.
- xの範囲は[-1,1]とします。

折れ線でグラフを描画

- 関数を描くときに,曲線を描くようなメソッドははじめから用意はされていません。
- 細かい折れ線を書いていき,擬似的に曲線を描かせます。

```
public void paintComponent(Graphics g) {
    //雛形Hina08_2.javaを使う
    //ここに作業を行う
}
```

まずは軸を描きましょう。

```
public void paintComponent(Graphics g) {
    g.setColor(new Color(255, 0, 0));

    //軸の線分を書く
    g.drawLine(0,250,500,250);
    g.drawLine(250,0,250,500);
    //軸の文字を書く
    g.drawString("x",480,260);
    g.drawString("y",260,10);
}
```

グラフを描くために

分割数を考え，刻み幅を指定：s

$f(x)=x^2$とすると

- $(x,f(x))$ から $(x+s,f(x+s))$ までの線
- $(x+s,f(x+s))$ から $(x+2s,f(x+2s))$ までの線
- $(x+2s,f(x+2s))$ から $(x+3s,f(x+3s))$ までの線

を繰り返す．

基本アルゴリズム

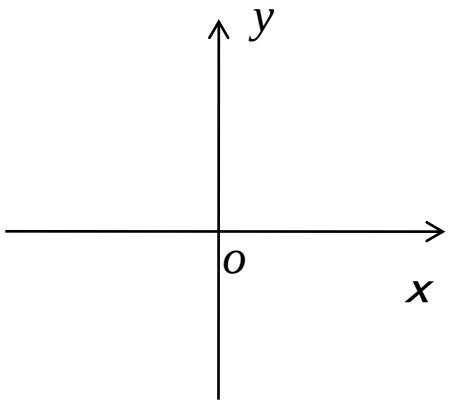
流れを考えましょう！(イメージ)

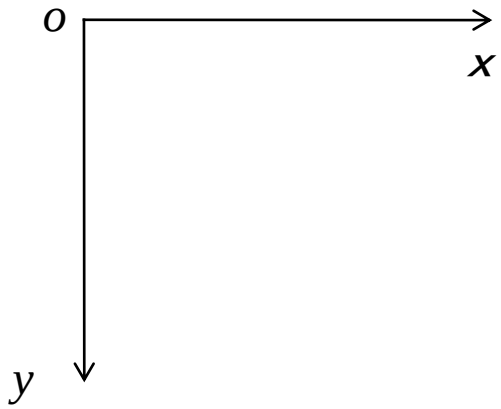
```
double x;
double step = 2.0/20.0;
for (x=-1;x<1;x=x + step) {
  x1=x;
  y1=x1*x1;
  x2=(x+ step);
  y2=x2*x2;
  ??? //(x1,y1)から(x2,y2)に線を引く;
}
```

これからコンピュータの座標に変換します.

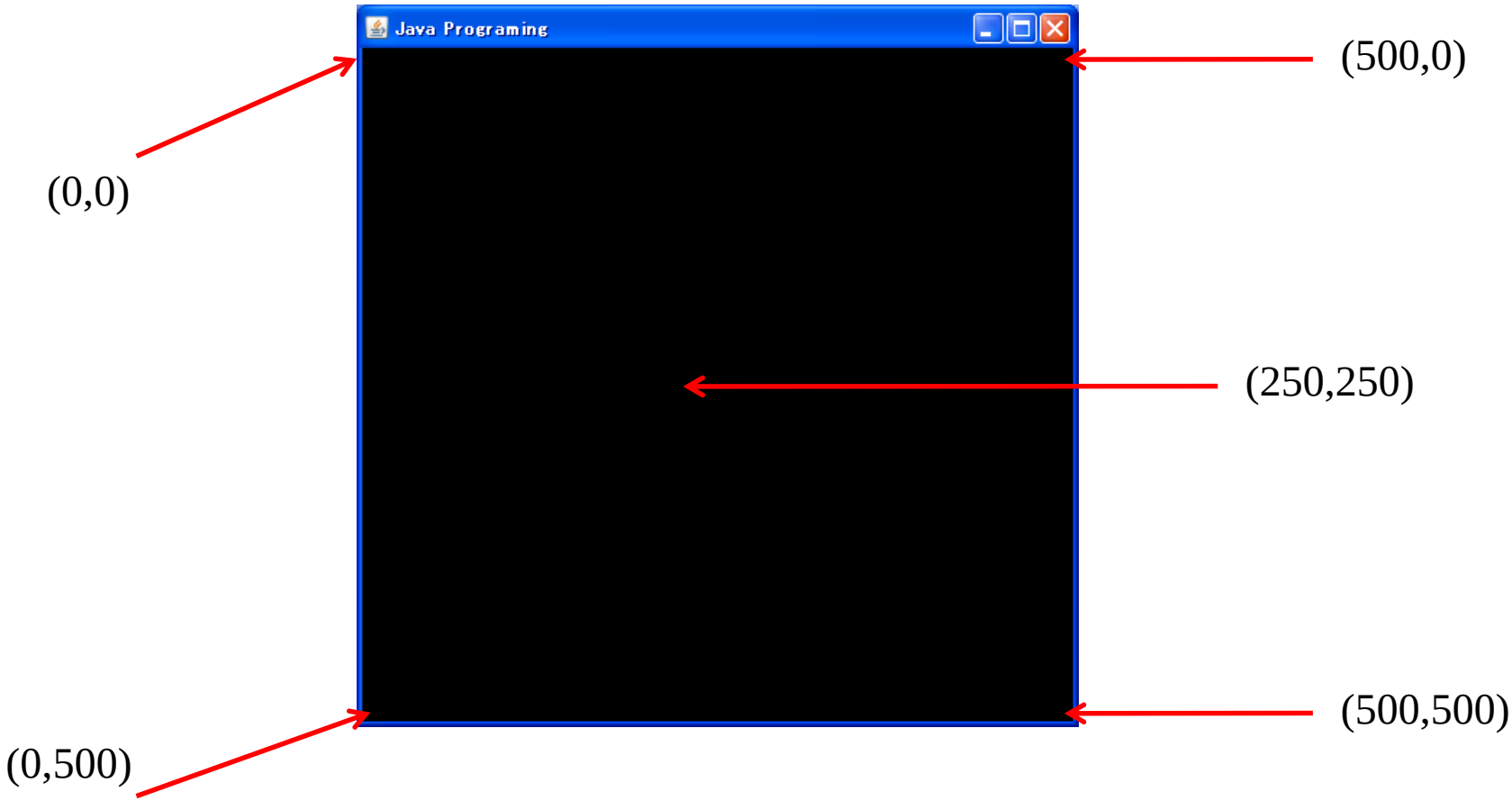
コンピュータの座標（復習）

- コンピュータの座標は数学の座標とは異なる
  - 正の整数のみで定義される.
  - ドット（ピクセル）単位であり，連続ではない.
  - y軸の向きが逆
  - 原点が左上





コンピュータの座標(復習)



座標の変換

- 数学上の座標(-1,1)×(-1,1)とコンピュータの画面500×500との対応関係を考えましょう.

- 数学上の座標
  - (0,0)は画面の(250,250)に対応(注意).
  - (1,1)は画面の(500,0)に対応
  - (-1,-1)は画面の(0,500)に対応

座標の変換(一般化)

- 数学上の座標(-1,1)×(-1,1)とコンピュータの画面500×500との対応関係を考えましょう.

- 数学上の座標(x,y)は画面の(x',y')に対応
  - x'=??
  - y'=??

(考えてください)

関数のグラフの描画（座標変換を含む）

```
public void paintComponent(Graphics g) {

      ??? //軸の描画
      g.setColor(Color.black);
      double x, step=2/20.0;
      int px1,py1,px2,py2;
      for (x=-1; x<1; x=x+step) {
            x1=x;    y1=x*x;
            x2=(x+step);     y2=(x+step)*(x+step);
            //座標の変換
            px1=(int)(250*x1+250);   py1=(int)(250-250*y1);
            px2=(int)(250*x2+250);   py2=(int)(250-250*y2);
            g.drawLine(px1,py1,px2,py2);
      }
}
```

考察

このプログラムにはいたらないことが多々ある。

- 一般性の無さ(画面のサイズが変わる場合を考えなさい)
- 計算の無駄（これは演習３で改善しなさい）

座標の計算を一般化する

もし,もっと大きな画面にグラフを描かせたいとする（つまり画面のサイズを変更する）。

すると全ての座標を変えなければならない

（これは面倒である。プログラムではなるべく固定の数値を使用しないのが一般である）

画面のサイズを習得する

Dimension クラスを利用する.

```
public void paintComponent(Graphics g) {
        Dimension d;   // 宣言
        d=getSize();    // 画面のサイズを得る
        // d.widthには横の長さが格納されている
        // d.heightには縦の長さが格納されている
}
```

参考

https://docs.oracle.com/javase/jp/1.4/api/java/awt/Dimension.html

軸を引く(一般)

```
g.setColor(new Color(255, 0, 0));
//軸を引いて
g.drawLine(0, d.height/2, d.width, d.height/2);
g.drawLine(d.width/2, 0, d.width/2, d.height);
//文字を書く
g.drawString("x", d.width-20, d.height/2+10);
g.drawString("y", d.width/2+20, 10);
```

以下の座標変換はどのように修正するか？

```
px1=(int)(250*x1+250);   py1=(int)(250-250*y1);
px2=(int)(250*x2+250);   py2=(int)(250-250*y2);
```

演習2

グラフを描かせる部分についてもDimension クラスの変数dを用いて
一般的にしなさい.

描画の画面サイズが変わると、グラフの変化を確認しなさい。

演習３

関数$y=\sin(x)$と$y=\cos(x)$ 両方のグラフを同時に描きなさい.

x の範囲を$-3 \leqq x \leqq 3$とする。

注意：

1. 二つのグラフは異なる色で描くこと。
2. 画面のサイズに応じてグラフは変化できること。
3. （オプション）今日のスライドの中(第１８頁)では，効率が悪い部分があると書きましたが，効率が悪い部分を指摘をして，演習３のコードでその部分を書き換えてください.

本日はここまで

- 本日の内容
  - 色の変更
  - グラフを描く
- 次回の内容
  - ラベルとテキストフィールドの配置と利用
  - ボタンとそのイベント処理