プログラミングA　（山内クラス）　第１３回

――― 今日の目標 ―――

もう少し、プログラミングで遊ぼう。　ずっと先の方のサンプルを、ちょっとだけ変更して遊ぶ。

――

１．　ずっと先のサンプル　⇒　教科書 p519　アニメーションする　を試す

Lesson 8 から Lesson 16 まで、いろいろと勉強することはあるけれど、それは秋学期のプログラミングＢでやるとして .....

教科書 p519　「アニメーションをする」　に書いてあるサンプルを試してみよう。⇒　それだけなら誰でも（プログラミングＡを勉強していなくても）できるから、少しだけ追加しよう。

まずは出発点として、教科書 p519 の Sample7.java を入力して試してみよう。

知らないことがたくさん出てくる。たとえば、`public static void main` と同列に（同じレベルで）、`public Sample7()` とか `public void run()` とか `public void paint()` とか、更には `class SampleWindowListener` というわけのわからないものが出てくる。

これはいずれも Lesson 8～12で新しく学ぶ内容で、その時のお楽しみ。

少しだけ説明すると、（あまり気にしなくてよい。どうせ秋学期に勉強するのだから）

```
import java.awt.*;
import java.awt.event.*;
public class Sample7 extends Frame implements Runnable  ⇒ Javaのawtライブラリの中の
`Frame`≒ウィンドウのデータを作るぞ
{
    int num;
    public static void main(String[] args) {  ⇒  今まで全部のサンプルで、mainがあったのと同じ
        Sample7 sm = new Sample7(); ⇒  Sample7型の変数 sm を作る～配列を作った時と同じnewだ！
    }
    public Sample7() {           ⇒  Sample7型を new で作った時に同時に初期化するときの手続き
        super("テスト");                    ⇒  ウィンドウ（frame）のタイトル
        addWindowListener(new SampleWindowListener());
                                    ⇒  後で定義するSampleWindowListenerをこのframeに足す
        Thread th;
        th = new Thread(this);      ⇒  （画面を動かすために）スレッドを１つ作る
        th.start();

        setSize(250,200);     ⇒  ウィンドウ（frame）の大きさの初期値を250x200にセット
        setVisible(true);     ⇒  ウィンドウ（frame）を見えるようにセット
                                        （これをtrueにしないと表示されない）
    }
    public void run() {       ⇒  ウィンドウの中で走る手続きを書く
        try {
            for (int i=; i<10; i++) {
                num = i;
                repaint();     ⇒  ウィンドウの中身を再度描きなおす（図が変わっている）
                Thread.sleep(1000);  ⇒  1000ミリ秒（＝1秒）寝る
            }
        } catch(InterruptedException e){}
    }
    public void paint(Graphics g) {     ⇒  ウィンドウの中身の絵を描く
        String str = num + "です。";
        g.drawString(str, 100, 100); ⇒ 文字列str描画、位置x=100,y=100（右上原点でy軸下向）
    }
    class SampleWindowListener extends WindowAdapter{
```

```
        public void windowClosing(WindowEvent e) { ⇒ ウィンドウでxが押された時ここへ来るので
            System.exit(0);                          ⇒ プログラムを終了する
        }
    }
}
```

さて、動いただろうか？　新しく出てきたウィンドウの右上の×印をクリックすると、プログラムがちゃんと終了するか？

２．　改造する前に：

まずは、Sample7.javaのコピーを作ろう。　新しいファイル名に合わせて、プログラム中の４か所の `Sample7` の文字列を、新しいファイル名と同じ名前に書き直す。　～～～　今までは `public class Sample7` の所だけだったが、今回は中で合計４か所に出てくる。全部揃えて書きなおす。

改造の意味をフォローしながら改造しよう。

注）改造は、上記の「秋学期に学ぶ」に引っかからないところで行う。気を楽に持とう。

３．　文字を動かしたい

改造は、絵（ここでは文字）を描いているところを変更する。　Sample7では数字の部分が変わるだけだったが、文字を描く位置も変えてみたい。

位置はどう指定していただろうか？

`g.drawString(str, 100, 100); ⇒　文字列strを描画する、位置はx=100, y=100（右上原点でy軸下向き）`

では、この位置を変えてみよう。位置 `(150, 150)` へ位置を変えるのは簡単だろうか？

では次の改造として、時間が経つとどんどん右へシフトしていくようにしたい。

⇒　時間が経つと　≒　`for` 文でループしている所だ。　　この `i` が、時間を表している。

この改造の前に、`for` 文のまわりを少し改造しておこう。　～～　見栄えが良くなるように。

1. `for` ループの回数を、10回から100回に増やそう。
2. `Thread.slep(1000); で寝る時間を、100ミリ秒に減らそう。　この結果、動くのが速くなる。

これで、試してみよう。更に、時間を表す `i` が、文字列を `drawString` している場所で見えるようにしよう。

3. `for` 文で `(int i=0 ...)` としている `int` 宣言を、外に出そう。具体的には
    ①　５行目の `int num;` の直後に `int i = 0;` を置こう。
    ②　`for` 文の方は、`int` を消そう。　つまり、`for (i=0; i<100; i++) { ...` としよう。

これで `g.drawString( ...` のところでも `i` が見えるはずなので、`i` を使って位置を変えてみよう。

4. たとえば `x` 座標を `100+i` とすると、どうなるか？　試してみよう。
    動きが小さい？　どうしたらよいか？（`i` が１増えるごとに `x` 座標が増える量を `i` ではなくて…）

さて、横にずれていくだけでは面白くない。　縦にもずらしてみよう。　どうすればよいか？

5. 横と縦と同時にずらすようにしてみよう。　斜めに動いただろうか？

これも物足りない。　ぐるぐる回るようにできるか？

6. 中心 `(125, 100)` の周りを文字列がぐるぐる回るように、座標を変えてみよう。
    ヒント：　三角関数は Math.sin(ラジアン)、Mah.cos(ラジアン)で計算できる。
    引数に与えるのはラジアンなので、たとえば `i` の値が50の時に１周する（2πになる）

ようにするのはどうか？

半径はたとえば`50`にしてみよう。つまりx座標は`125 - 50*Math.sin(ラジアン)`となる？

まだ物足りない？　では、回りながら半径がだんだん小さくなる（とか大きくなるとか）はできるか？

7. 上記では半径の値を`50`の固定値にしたが、これをたとえば　`50 - i`　としたらどうなるか？

あとは、好きなようにいじってごらんなさい。

４．　何か他のもの（図形？）を描き足そう

次の挑戦は、文字だけでなくて他の図形も描き足せることを試してみよう。何が描けるかは、適当にネットで検索するか、

https://docs.oracle.com/javase/jp/7/api/java/awt/Graphics.html

を参照して欲しい。たくさんあるうちの`drawOOO`が、図形を描く関数で、たとえば`drawLine`は直線を描く、`drawRect`は長方形を描く、`drawOval`は楕円（円も含む）を描く。`drawImage`でイメージ（画像）を貼り付けることもできるが、画像のデータを与えるのに若干の慣れが必要なので、今回は見送り。

では、`drawString`のすぐ下に、`drawRect`で長方形を描いてみよう。（円や楕円を描いてもいい）

```
drawRect(x座標, y座標, 長方形の横幅, 長方形の高さ)
```

この`x`座標や`y`座標は画面の原点（＝画面左上の隅）からの座標で、`y`軸は下向き。

描く位置は、文字列と同じ位置だと分かりづらいので、全く別に`i`の値が増えるに従って左下から右上に、移動することにしようか。　大きさも`i`が増えるに従って大きくなるのはどうだろうか？

５．　おまけ、　色でも変えてみる？

おまけの挑戦は、色でも変えてみるか、という話。要素ごとに色を変えたいところだが、面倒なので、`g.drawString(...)`の直前で色を設定してみる。`paint()`は`repaint()`するたびに呼ばれるので、毎回色をセットすることになる。

色のセットは

```
g.setColor(Color.RED)
```

のようにするとできる。

簡単に使える（あらかじめ調合された）色は、たとえば`color.GREEN`, `color.ORANGE`, `color.BLUE`, `color.PINK`, `color.YELLOW`など。

https://docs.oracle.com/javase/jp/6/api/java/awt/Color.html

を参照のこと。　他の色の調合は、自分で調べてください。たとえば`Color(120, 70. 90)`とかすると、RGBで赤が120、緑が70、青が90の色を作ることができるハズ。それぞれの要素は0〜255の整数で、大きいほど明るく、(255, 255, 255)は白、(0, 0, 0)は黒、(255, 0, 0)は赤、など。

それで、`i`の値が進むに従って色が変わるようにする、または、たとえば`i`が10の周期で色がいろいろと変わる、などのことは簡単にできるでしょう。アイデアとしては、

```
    if (i%10<=3){
        g.setColor(Color.RED);
    } else if (i%10<6){
        g.setColor(Color.GREEN);
    } else {
        g.setColor(Color.BLUE);
    }
```

などはすぐにできる。自分で好きなように、色を変えてみてください。更に、図形を描く時に`drawRect`や`drawOval`の代わりに`fillRect`や`fillOval`を使うと、図形を塗りつぶすので、色がより目立つかも。

最後に：

授業評価アンケート（*Active Academy*）を書いてください。

＋

来週（７月１５日）の授業は、出席はとりません。

内容は、前半がファイルからデータを読んで処理する話、後半は質問・相談・その他のつもりです。