第１章．Java による CG 作成方法②

【学習のねらい】

① 先週に続いて Java 言語（Eclipse 環境における）を用いて CG（コンピュータグラフィックス）を作成する方法の基礎を学習する。今回は、（作成した）CG が自動的に再描画される様にするための処理（のプログラミング）を学習する。

今回の学習で Java による CG 作成方法を終了し、次週以降は CG 作成のアルゴリズムの学習に入ります。今回の学習内容が身に付いていなければ、次週以降の CG 作成ができません。しっかりと学習して下さい。

１－３　再描画処理（描画処理の改良）

「プログラミング」テキストの 8-7 節（p.227～235（p.225～233：2012 年度版））を良く読み、【応用課題 8-7-A】のプログラムを作成して下さい。ただし、作成の前に、以下の＜作成上の注意＞を良く読んで下さい。

＜作成上の注意＞

① p.228（p.226：2012 年度版）の【準備】から行わなければ作成できません。

② もし、何らかのプロジェクトが開かれていたら、［ファイル］→「すべて閉じる」によって、必ず全てのプロジェクトを閉じてからアプリケーションの新規作成を行って下さい。今後も、混乱を避けるため新規にプログラムを作成する際にはプロジェクトを閉じるようにして下さい。

③ テキストの p.231（p.229：2012 年度版）で MyPanel コンポーネントをフレーム上に貼り付ける際には、これまで通りフレームの Layout を「Absolute Layout」にしておいて下さい。

④ p.234～235（p.232～233：2012 年度版）で用いている **Image** オブジェクトについては、テキストの 8-6 節で説明しています。p.215～218（p.213～216：2012 年度版）の【練習課題】を読んで下さい。

プログラムを作成し、再描画処理が行われることを確認したら、次の【基礎課題 1-4】以降に進んで下さい。

【基礎課題 1-4】

p.234（p.232：2012 年度版）で説明している MyPanel クラスを次のように（２ヶ所）書き換えました。下線部が変更部分です。このプログラムが【応用課題 8-7-A】と同様に動作するためには、空欄部分をどのように書き換えればよいですか？適切な変数名（オブジェクト名）あるいは命令文を記入して下さい。

```
public class MyPanel extends JPanel {
  private Image Img1=null; //Image オブジェクトの宣言
  public void paintComponent(Graphics g){
    super.paintComponent(g);
    if( ①  !=null) {
     g.drawImage( ②  ,0,0,this);
    }
  }
  public void setImage(Image img2) {
     ③  ;
  }
}
```

変更部分

＜課題提出の仕方について＞

先週のプリントの p.2 で説明した要領で課題を提出して下さい。本日提出してもらうのは【基礎課題 1-3】～【基礎課題 1-6】の 4 題です。

【基礎課題 1-5】

テキスト p.235（p.233：2012 年度版）に与えている［描画］ボタンクリック時のプログラムについて、以下の①から③の設問に答えて下さい。

```
private void jButton1ActionPerformed(ActionEvent evt) {
  Graphics g=myPanel1.getGraphics();
  g.setColor(Color.black);
  g.fillOval(10,10,200,100);
  g.dispose();
  int w=myPanel1.getWidth(); //パネル幅の取得
  int h=myPanel1.getHeight(); //パネル高さの取得
  Image img=this.createImage(w,h);①
  Graphics g2=img.getGraphics(); //Graphics オブジェクトの取得
  g2.setColor(Color.white);
  g2.fillRect(0,0,w,h);
  g2.setColor(Color.black);
  g2.fillOval(10,10,200,100);
  myPanel1.setImage(img);③
  g2.dispose();
}
```

②

※　プログラム中の①～③の番号がそれぞれの設問に対応します。

① `createImage(w,h)` メソッドが行う処理内容として最も適切なものを下の選択肢から選び、その記号を記入して下さい。

A．パネル上の座標（w,h）の位置に Image オブジェクトの画像を表示する。
B．幅 w、高さ h の四角形の Image オブジェクトを生成する。
C．幅 w、高さ h の四角形の Image オブジェクトの画像をパネル上に表示する。

② 点線枠で囲まれた部分をプログラムから削除した場合、再描画時の実行結果はどのようになりますか？最もあてはまるものを選択肢から選びその記号を記入して下さい。

A．以前と全く同様に再描画される。
B．楕円が再描画されなくなる。
C．楕円は再描画されるが、パネルの白い枠は描画されなくなる。

③ `setImage(img)` メソッドの処理内容に関する説明として最もよくあてはまるものを次の選択肢から選び、その記号を記入して下さい。

A．Image オブジェクト img を MyPanel クラス内のオブジェクトに引き渡す。
B．Image オブジェクト img を MyPanel クラス内のオブジェクトに表示させる。
C．Image オブジェクト img に楕円を描画する。

１－４　再描画処理の応用

今後は、パネル上に様々な CG を作成して行きます。そのためには、1-3 節で作成したプログラムの MyPanel クラスは改変する必要はなく、ただ［描画］ボタンクリック時のプログラムを（作成する CG の内容に応じて）変更すれば良いだけです。そのことを具体的に確認しましょう。

前節で作成したプログラム（Ouyou8_7_A）中の［描画］ボタンクリック時のプログラムを開いて下さい。

```
private void jButton1ActionPerformed(ActionEvent evt) {
  Graphics g=myPanel1.getGraphics();
  g.setColor(Color.black);
  g.fillOval(10,10,200,100);
  g.dispose();
  int w=myPanel1.getWidth(); //パネル幅の取得
  int h=myPanel1.getHeight(); //パネル高さの取得
  Image img=this.createImage(w,h); //Image オブジェクトの生成
  Graphics g2=img.getGraphics(); //Graphics オブジェクトの取得
  g2.setColor(Color.white);
  g2.fillRect(0,0,w,h);
  g2.setColor(Color.black);
  g2.fillOval(10,10,200,100);
  myPanel1.setImage(img); //画面を Image オブジェクトとして設定
  g2.dispose();
}
```

これは（黒色の）楕円を描くプログラムでしたが、例えば赤色の四角形を描くプログラムに変更するには、上の（2 ヶ所の）点線枠部分を修正すれば良いのです。直接修正しても良いのですが、よく見ると分かるように、点線枠内の処理は、対象とする Graphics オブジェクトが違う（g と g2）のみで、あとは全く同じです。このような場合は、描画を行うメソッドを定義すれば記述は一度で済み、また描画内容が複雑になった時にも拡張が容易になります。

そこで、上のプログラムの後に以下の様に、CG の描画を行うメソッド DrawGraphics() を定義してみましょう。

```
private void jButton1ActionPerformed(ActionEvent evt) {
   ・・・
}

void DrawGraphics(Graphics g) {
  g.setColor(Color.red);
  g.fillRect(10,10,200,100);
}
```

新たに定義した部分

これを用いれば、［描画］ボタンクリック時のプログラムは次のように、前ページの（2 ヶ所の）点線枠部分を DrawGraphics()メソッドで置き換えることで実現できます。

```
private void jButton1ActionPerformed(ActionEvent evt) {
  Graphics g=myPanel1.getGraphics();
  DrawGraphics(g);
  g.dispose();
   ・・・
  g2.fillRect(0,0,w,h);
  DrawGraphics(g2);
  myPanel1.setImage(img); //画面を Image オブジェクトとして設定
  g2.dispose();
}
```

プログラムを修正したら、各自動作を確認して下さい。

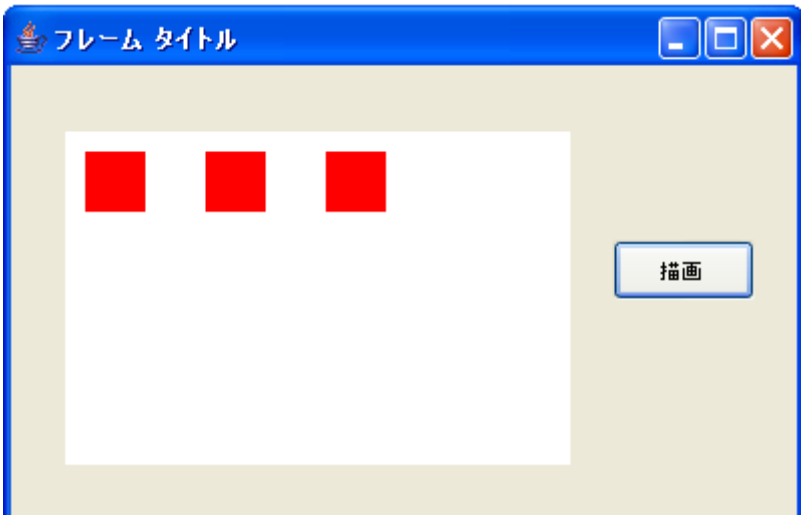
このようにプログラムを整理しておくと、新たに描画内容を書き換えるときには、メソッド **DrawGraphics()の内容（のみ）を修正するだけで良い**事が分かります。したがって以降の学習では、作成する CG に応じてこの DrawGraphics()内のプログラムを書き換えて行く事になります。

【基礎課題 1-6】　既存のプログラムを利用してプログラムを作る方法

今後、課題毎に新たにプログラムを作成して行きますが、上で述べた様に一度（基になる）プログラムを作成すれば、NewJFrame クラスの DrawGraphics(g)メソッド以外の部分は書き換える必要がありません。ですから、その度に同じ内容を記述するのは面倒で、できれば、修正する部分のみを新たに記述するだけで済ませたいものです。それを実現する、最も手っ取り早い方法はプロジェクト全体をコピーしてそれを用いる方法です。

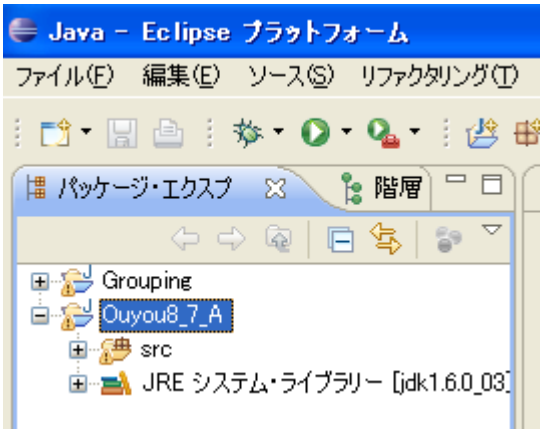
以下、プログラム「Ouyou8_7_A」を修正して、次の様に（３つの）四角形を表示させる

プログラムを作る場合を例に取り上げて、その方法を紹介しましょう。手順は以下の通りです。

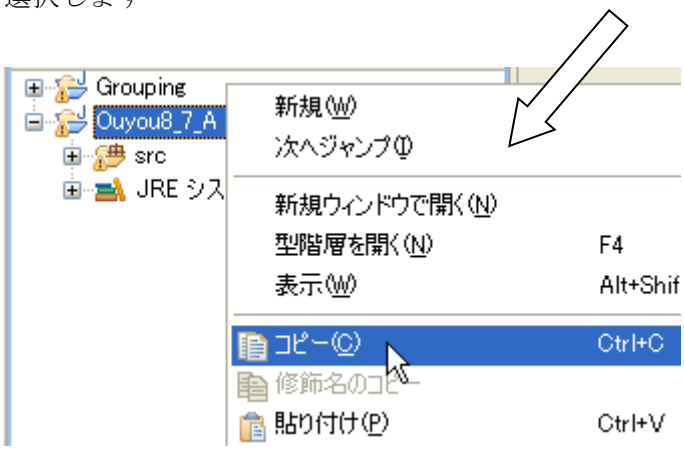


① 今開いているプロジェクトがあったらいったんそれらを閉じます。

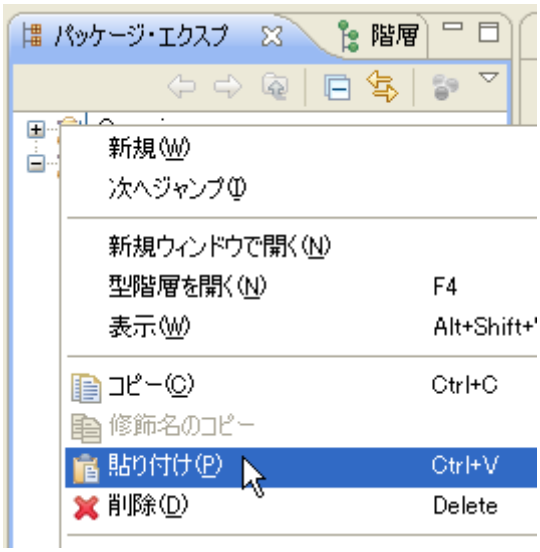
② 次に、利用する（基になる）プロジェクト「Ouyou8_7_A」（別の名前で作った人はそれに読み替えて下さい）をパッケージエクスプローラから選択して、右ボタンクリックします。



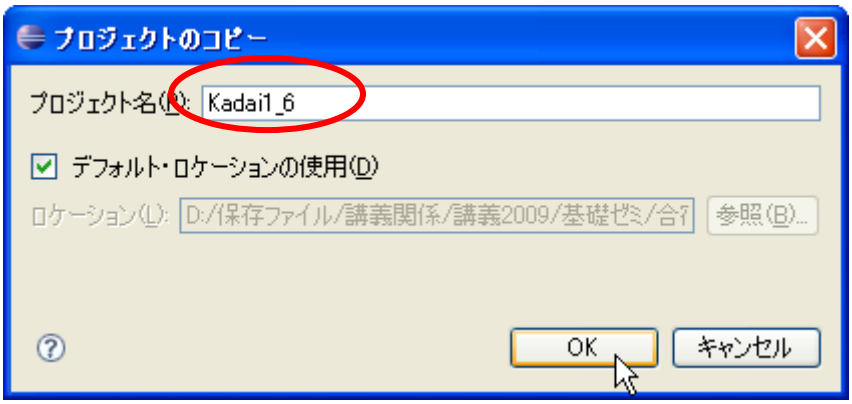
そして、現れたメニューから「コピー」を選択します



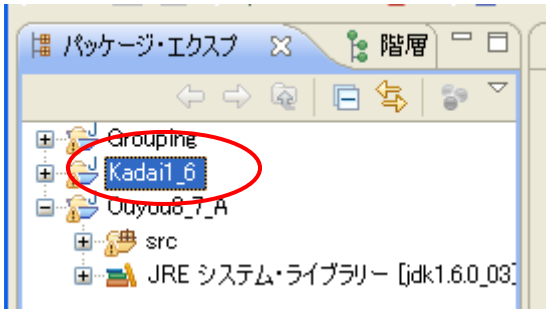
③　続いてパッケージエクスプローラ内で右ボタンクリックし、「貼り付け」を選択します。



すると、次の画面が現れるので、（コピー先の）プロジェクト名を指定します。ここでは、「Kadai1_6」としましょう。



④　これで、プロジェクト「Ouyou8_7_A」が「Kadai1_6」という名前でコピーされ、次のように、パッケージエクスプローラに表示されます。これは、以前のプロジェクト（Ouyou8_7_A）とは独立の（新しい）プロジェクトなので、自由に変更修正ができます。



⑤　新たに作成したプロジェクト「Kadai1_6」の NewJFrame.java にある DrawGraphics(g)メソッドを次のように書き換えます（今の場合）。

```
void DrawGraphics(Graphics g) {
  g.setColor(Color.red);
  g.fillRect(10,10,30,30);
  g.fillRect(70,10,30,30);
  g.fillRect(130,10,30,30);
}
```

作成したら実行してみてください。うまく表示されたら、

プログラム作成方法を確認しました。

というメッセージを送って下さい。それを以てこの課題の提出とみなします。

次週以降もこの要領でプログラムを作って行って下さい。作成時間を節約し、CG 作成部分に集中できるはずです。