プログラミング言語IIIB(Java) テーマ20

Timerクラス

Timerクラスを用いると, 予め設定した時間間隔で自動的にアクションイベントが繰り返し発生し, actionPerformedメソッドが呼び出される. この機能を利用して, 一定時間毎に少しずつ異なる画像や図形を, 少しずつ異なる位置に描画する事で, アニメーション動作を実現できる.

Timerクラスを利用する方法は以下の通りである. 今回は, タイマーをフレーム側に設置し, タイマーで定期的に呼び出されるactionPerformedメソッドから, 描画パネルに対してrepaintをかける方式で行う.

1. フレーム側のフィールドに, Timerクラスのオブジェクト変数を追加する.

```
// -------- MyFrame側フィールド --------
private Timer timer;
```

2. フレーム側のコンストラクタで, Timerクラスのインスタンスを生成する. 引数は呼び出し間隔(単位ミリ秒)と, 呼び出されるactionPerformedメソッドを持つインスタンスである(今回はthis). その後, タイマーをスタートさせる. ※actionPerformedメソッドを使うので, implements ActionListenerも必要.

```
略 implements ActionListener

// -------- MyFrame側コンストラクタ --------
this.timer = new Timer(500, this); // 0.5秒ごとにthisのactionPerformedメソッドを呼び出す
this.timer.start();                // タイマースタート
```

3. フレーム側のactionPerformedメソッドに, 一定時間毎に行いたい処理を書く. (今回は描画パネルのrepaint)

```
// -------- MyFrame側メソッド --------
public void actionPerformed(ActionEvent e)
{
    System.out.println("in actionPerformed");
    this.mp.repaint(); // 描画パネルに対して再描画を指示する → paintComponentが呼び出される
    Toolkit.getDefaultToolkit().sync(); // OSによるフレームレート低下への対策
}
```

4. 描画パネル側のpaintComponentで, グラフィックスを描く.

```
// -------- MyPanel側メソッド --------
public void paintComponent(Graphics g)
{
    System.out.println("in paintComponent");
    super.paintComponent(g); // 背景クリア

    // ここで, 少しずつ異なる画像や図形を, 少しずつ異なる位置に描画するとアニメーションになる.
}
```

一つのクラスに全ての機能を詰め込むのではなく, クラスを2つに分けて, 役割を分担させている点に注目せよ.

Timerクラスにはこの他にも様々なメソッドがある. 自分のアイデアに応じて適宜調査して利用してみて欲しい.

画像ファイルの読込みと描画

パネルに描く画像は以下の手順で用意する. どこ(場所)に, 何を記述するのか, 良く注意すること.

1. フレーム側のコンストラクタで画像ファイルを読み込む. 画像ファイルはプログラムと同じディレクトリに置いておく.

```
// -------- MyFrame側コンストラクタ --------
Toolkit tk = Toolkit.getDefaultToolkit(); // 画像ファイルを読み込む機能等が入った道具箱
Image chara = tk.getImage("chara.png"); // chara.pngを読み込み, 変数charaに代入する.
```

注意!! 画像ファイルが無く, 読込みに失敗してもエラーは出ないので, 画像ファイルは確実に用意しておく. ファイル名やディレクトリを良く確認のこと.

2. フレーム側のコンストラクタで, 描画パネルのインスタンスを生成する時に, 手順1で用意した画像インスタンスを引数として渡す.

```
// -------- MyFrame側コンストラクタ --------
this.mp = new MyPanel(chara); // 引数charaに注目
                      ~~~~~ここ
```

3. 描画パネル側のコンストラクタで, 引数で受け取った画像をフィールドに保存しておく.

```
// -------- MyPanel側フィールド --------
private Image chara;
```

```
// -------- MyPanel側コンストラクタ --------
public MyPanel(Image chara) // 引数で受け取り
{
    this.chara = chara;     // フィールドに保存する．セッターと同じ考え方．
}
```

※コンストラクタの引数で渡すのではなく, セッターで渡す方式にしても良い(同じこと).

4. 描画パネル側のpaintComponentメソッドで, 画像を描画する.

```
// -------- MyPanel側メソッド --------
public void paintComponent(Graphics g)
{
    super.paintComponent(g);
    g.drawImage(this.chara, X座標, Y座標, this);
    // X座標, Y座標は自分で数値を設定する, 最後の引数はthisにする.
}
```

画像の配列

アニメーションを行うためには, 少しずつ異なる複数の画像が必要となる. それら複数の画像をひとまとめに扱うには, 配列が便利である.

以下に, 3枚の画像から成る配列の生成方法の例を示す. 前節の例では単一の画像データ(chara)をフレーム側から描画パネル側へ渡していたが, この場合は画像の配列(chara_array)を渡している.

```
// -------- MyFrame側コンストラクタ --------
Image[] chara_array = new Image[3]; // Imageクラスの配列
for (int i = 0; i < 3; i++) {
    String filename = "chara" + i + ".png"; // これで chara0.png, chara1.png, chara2.png の三つになる
    System.out.println("filename = " + filename); // 確認のためファイル名を表示
    chara_array[i] = tk.getImage(filename);
}
this.mp = new MyPanel(chara_array); // 配列を渡す．受け取る方も当然，修正が必要.
```

※大切な事なのでもう一度: 受け取る方も修正が必要. どの様に修正するかは, 自分で考えてみましょう.

※配列とfor文が苦手な人は, 前期の内容を再確認して十分に復習しておく.

アニメーション

描画パネル側のフィールドに, 描画する画像番号を表す変数を追加し, その番号の画像を描画する. その後, 画像番号をインクリメント(+1)する.

また, キャラクターを描く座標(x, y)もフィールドとし, (x, y)の位置に画像を描く. (x, y)を次々と変化させることで,「動き」を表すことができる. 例えばx座標を増加させると右に, 減少させると左に, 放物線状に変化させるとジャンプ動作, 極座標(円座標)を使うと回転動作することになる.

```
// -------- MyPanel側フィールド --------
private int chara_i, chara_x, chara_y;

※注意: フィールドの名前を単純な x, y にしないこと．(GUI部品のスーパークラスにあるため.)

// -------- MyPanel側コンストラクタ(初期値の設定) --------
this.chara_i = 0;
this.chara_x = 200;
this.chara_y = 100;

// -------- MyPanel側メソッド --------
public void paintComponent(Graphics g)
{
    super.paintComponent(g);
    System.out.println("chara_i = " + this.chara_i);
    System.out.println("chara_x = " + this.chara_x);
    System.out.println("chara_y = " + this.chara_y);
    g.drawImage(this.chara_array[this.chara_i], this.chara_x, this.chara_y, this);

    // 動きのコードの例
    this.chara_i++; // 画像の番号を進める
    if (this.chara_i >= this.chara_array.length) { this.chara_i = 0; } // 最後まで進んだら初期値に戻す
    this.chara_x -= 2; // 2ピクセル左へ移動させる
    if (this.chara_x < 0) { this.chara_x = 200; } // 左端まできたら初期値に戻す
```

課題

以下の課題のレポートは, レポートファイルreport20.txtを作成してアップロードにより提出すること.

1. Timerクラスを用いて, 1秒ごとにpaintComponentメソッドが呼び出されるプログラムを作成せよ. テーマ18の課題1のプログラムに, 上記のTimerクラスの説明のサンプルコードを組み込んで作成すると良い. paintComponentメソッドの中にSystem.out.printlnを入れて何か文字列を表示し, そこが繰り返し実行されていることを確認すること. (ソースをレポート)
2. 画像ファイルを一つ読み込み, 描画パネル内の適当な固定位置に, その画像を描画するプログラムを作成せよ. (ソースをレポート)

- 画像ファイルは各自の好みで用意して良いが, ハラスメントや誹謗中傷となる画像は避けること. GIMP等による自作を推奨するが, フリーの各種素材を利用しても良い.
- [重要] 画像の大きさは64x64ピクセル以下程度の小さいものを用いること.
- ※大切な事なのでもう一度: 画像は小さいものを用いること.
- getImageメソッドによる画像ファイルの読込みはバックグラウンドでの並列処理となるため, 読込み処理が完了せずとも次の行に進んでしまう. このため, 画像ファイルが巨大だったりネットワークに遅延があると, まだ画像の読込みが終了していない状態で描画してしまい, 画像が表示されないという不具合が発生する. この問題を回避するためにはMediaTrackerクラスという, 画像ファイルの読込み完了を待つ(wait)機能を用いる必要がある. 以下のサンプルを参考にして自分のプログラムに組み込んでみて欲しい.

```
// -------- MyFrame側コンストラクタ --------
Toolkit tk = Toolkit.getDefaultToolkit();
MediaTracker tracker = new MediaTracker(this); // これを追加する

// 画像ファイルを読み込む
Image chara = tk.getImage("chara.png");
tracker.addImage(chara, 0); // これを追加する

// 読み込みが完了するまで待つ
try {
    tracker.waitForAll();
} catch (Exception ex) {
    System.exit(1);
}
```

3. 画像ファイルを複数読み込み, 適当な固定位置に, それぞれ位置をずらして全て描画するプログラムを作成せよ. 画像の一覧を静止画表示するということである. 複数の画像は配列を用いてまとめて扱うこと. (ソースをレポート)
   - 画像ファイルは少なくとも3枚あれば良い.
   - まだアニメーション動作にする必要はない. →それは課題4にて
4. 課題3のプログラムを元に, Timerクラスを用いて画像のアニメーション表示を行うプログラムを作成せよ. (ソース, アニメーションの絵的な説明および, 座標や画像番号の変化などコード的な説明をレポート)
   - 画像ファイルは少なくとも3枚あれば良い. むしろ動き(座標)の変化や, 状態(画像番号)の変化に力を注いで欲しい.
   - ※大切な事なのでもう一度: 座標の変化や状態の変化を重点的に.
   - 画像を描画する位置を時々刻々と変化させるのであるから, 画像の大きさは64x64程度に小さくすること. 大きな画像を内容だけ変化させて固定位置に描画しては間違いである.
   - ※大切な事なのでもう一度: 画像は小さいものを用いること.
   - 上級者向け: アニメーションで動かしたいキャラクター画像の背景部分を透過させ, 大きな背景画像と重ね合わせするには, キャラクターの画像ファイルを透過PNGに変換しておくと良い.

ヘルプとアドバイス

- 今回用いるTimerクラスはjavax.swing.Timerである. 間違えてjava.util.Timerを使うとエラーや挙動不審の原因となるので注意せよ.
- タイマーで呼び出されるactionPerformedは, 短い時間間隔で次々と繰り返し呼び出されるものであるから, actionPerformedやpaintComponentの中であまりに時間のかかる重い処理を行うと, 処理落ち(コマ落ち)が発生してしまう.

レポート

- 内容: 課題中に指示されている通り. 必要な項目を全て記載しているか, 十分に確認すること.