Java独習　第3版

13.12 スレッドの使用
13.13 ダブルバッファリング

2006年7月12日(水)　　南　慶典

13.12 スレッドの使用

アニメーション

アニメーションやシミュレーションなどは画面の更新が一定のタイミングで行われていく。この連続した画面の更新をスレッドを利用して行う。

しかし、paint()メソッドを直接呼び出して表示を更新することはできない。

その理由は、
▶多くの重要な作業のスケジュールはJVMによって制御されているから

- ガーベジコレクション
  →不要になったオブジェクトのメモリリソースを自動的に回収
- I/O（入出力の略）
- スレッド管理、、、など。

表示の更新もそういた作業のひとつに当たる

ウィンドウを更新するには

1．repaint()メソッドを呼び出しウィンドウの更新を要求する

2．JVMでウィンドウを更新する適切なタイミングが判断され、update()メソッドが呼び出される

3．update()メソッドの既定の実装では、アプレットウィンドウを背景色（デフォルトでは白色）で塗りつぶしてからpaint()メソッドを呼び出す

またAppletクラスを拡張しなければならないのでスレッドを作成するにはRunnableを実装する。

例　スレッドの使用

```
import java.applet.*;
import java.awt.*;
/*
  <applet code="Counter" width=250
height=100>
  </applet>
 */


public class Counter extends Applet
implements Runnable{
      int counter;
      Thread t;

      public void init(){

            // カウンタを初期化する
            counter = 0;

            // スレッドを開始する
            t = new Thread(this);
            t.start();
      }

      public void run() {
            try{
                  while(true){

                        // 再描画を要求する
                        repaint();

                  // 次のカウンタを表示する前に休止する
                        Thread.sleep(1000);

                        // カウンタを1つ進める
                        ++counter;
                  }
            }
            catch(Exception e){
            }
      }
```

```
    public void paint(Graphics g){
        //フォントを設定する
      g.setFont(new Font("Serif", Font.BOLD, 36));

        // フォントメトリックスを取得する
        FontMetrics fm = g.getFontMetrics();

        // カウンタを表示する
        String str = "" + counter;
        Dimension d = getSize();
        int x = d.width / 2 - fm.stringWidth(str) / 2;
        g.drawString(str, x, d.height / 2);

    }
}
```

update()メソッドはオーバーライドしていない

- アプレットウインドウが背景色で塗りつぶされた後でpaint()メソッドが呼び出される

このアプレットでは1秒間隔で数字が表示されていく。

例　update()をオーバーライド

0.2秒間隔でアプレットウィンドウ上に無作為に点を表示するプログラム。update()メソッドをオーバーライドして、前に表示した点を消去しないようにしている。

```
import java.applet.*;
import java.awt.*;
/*
    <applet code="Dots" width=250
height=100>
    </applet>
 */

public class Dots extends Applet
implements Runnable{
        Thread t;

        public void init(){

            // スレッドを開始する
            t = new Thread(this);
            t.start();
        }

public void run(){
    try{
        while(true){

            // 再描画を要求する
            repaint();

        / 次の点を表示する前に休止する
            Thread.sleep(200);
        }
    }
    catch(Exception e){
    }
}

public void update(Graphics g){
    paint(g);
}
```

```
    public void paint(Graphics g){

        // ウィンドウ内の無作為な場所を選択する
        Dimension d = getSize();
        int x = (int)(Math.random() * d.width);
        int y = (int)(Math.random() * d.height);

        // そこに点を描画する
        g.fillRect(x, y, 2, 2);
    }
}
```

update()メソッドをオーバーライド

- paint()メソッドを呼び出す前に、アプレットウィンドウを背景色で塗りつぶさないようにしている。

このアプレットでは、点が無作為な場所に表示され、増えていく。

13.13 ダブルバッファリング

ちらつきを防ぐ

- update()メソッドによる『ちらつき』
ウィンドウ全体を背景色で塗りつぶして消去し、その後paint()メソッドを呼び出してアプレットの出力を表示する

⇒見苦しい『ちらつき』の原因となってしまう

- **ダブルバッファリング**
  アプレットウィンドウのちらつきを防ぐ手法

1. update()メソッドをオーバーライドし、ウィンドウ全面を消去しないようにする。このメソッドではただpaint()メソッドを呼び出すようにする。

2. またpaint()メソッド内では、全ての描画操作をバッファ（オフスクリーン）に対して行う。そして最後にバッファをウィンドウにコピーする。

バックグラウンドバッファの作成

バックグラウンドのバッファ（オフスクリーン）を作成するには、
Componentクラスのcreateimage()メソッドを使う

```
Image createImage(int width, int height)
```

widthとheightは、オフスクリーンとなるバッファのサイズを指定する。
例ではアプレットウィンドウと同じサイズのバッファを作成している。

この作成したバッファ（オフスクリーン）に対して描画を行っていく。

その描画を行う際に、グラフィックコンテキストを用意する必要がある。

グラフィックコンテキストの取得

グラフィックコンテキスト
　⇒　描画環境のようなもの（色だとかフォントだとか、、）

バッファのための描画環境を用意する必要がある。このバッファのグラフィックコンテキストは、ImageのgetGraphics()メソッドを使って取得する。

```
abstract Graphics getGraphics()
```

このグラフィックコンテキストを使って描画操作を実行し、バッファをウィンドウにコピーする。

⇒　次の場面だけが映されていくのでちらつきを解消できる!!

コピーするにはGraphicsクラスのdrawImage()メソッドを使う。

ダブルバッファリングしない例

```
import java.applet.*;
import java.awt.*;
/*
    <applet code="NoDoubleBuffer"
width=300 height=100>
    </applet>
 */

public class NoDoubleBuffer extends
Applet
implements Runnable{
    int x = 0;
    Thread t;

    public void init(){

        // スレッドを開始する
        t = new Thread(this);
        t.start();
    }

    public void run(){
        try{
            while(true){

                // 再描画を要求する
                repaint();

    // ウインドウを更新する前に休止する
                Thread.sleep(100);
            }
        }
        catch(Exception e){
        }
    }
```

```
    public void paint(Graphics g){

        // 塗りつぶした円を描画する
        Dimension d = getSize();
        g.fillOval(x, d.height / 4, 50, 50);

        // x座標を増やす
        x += 5;
        if(x + 50 > d.width)
            x = 0;

    }
}
```

ダブルバッファリングする例

```
import java.applet.*;
import java.awt.*;
/*
      <applet code="DoubleBuffer"
width=300 height=100>
      </applet>
 */

public class DoubleBuffer extends Applet
implements Runnable{
      int x = 0;
      Thread t;
      Image buffer;
      Graphics bufferg;

      public void init(){

            //スレッドを開始する
            t = new Thread(this);
            t.start();
```

```
}

// バッファを作成する
Dimension d = getSize();
buffer = createImage(d.width, d.height);


public void run(){
      try{
            while(true){

                  // 再描画を要求する
                  repaint();

// ウィンドウを更新する前に休止する
                  Thread.sleep(100);
            }
      }
      catch(Exception e){
      }
}

public void update(Graphics g){
      paint(g);
}
```

```
    public void paint(Graphics g){

    // バッファのグラフィックコンテキストを取得する
        if(bufferg == null)
            bufferg = buffer.getGraphics();

        // バッファを描画する
        Dimension d = getSize();
        bufferg.setColor(Color.white);
        bufferg.fillRect(0, 0, d.width, d.height);
        bufferg.setColor(Color.black);
        bufferg.fillOval(x, d.height / 4, 50, 50);

        // ウィンドウを更新する
        g.drawImage(buffer, 0, 0, this);

        // x座標を増やす
        x += 5;
        if(x + 50 > d.width)
            x = 0;
    }
}
```

init()メソッドでアプレットと同じサイズのバッファ（オフスクリーン）を作成

paint()が最初に呼び出された時に、グラフィック画面が作成される(バッファに対して描画が行われている。)

repaint()メソッドでウィンドウを更新（バッファの画を貼り付けていく）

練習問題

問題1
デジタル時計をアプレットで作成しなさい。時計は何時何分何秒まで表示させること。

問題2
虹色カラーバーを作成し、左から右に動いていくアプレットを作成しなさい。その際ダブルバッファリングしていないものと、ダブルバッファリングしたものと2つ作成しなさい。虹色カラーバーはP396~P397の例題を参照すること。