GUI 2

プログラミングでお絵かきする

2018/5/15(火)　プログラミングIV 第六回

福井大学　工学研究科　情報・メディア工学専攻

長谷川達人

演習問題の解説

- HP上にアップロードした.
- 点数に不服がある人は個別に申し立ててください.
- 今から解説します.

- 学籍番号を入力すると，成績を表示する仕組みを開発してみましたので使ってみてください.

質問への回答

- 「ボタンを押している間ずっと」とか「ボタンを離したとき」等を検出するにはどうすればよいか.
  - 今日学習するMouseListenerを使う.
- 2つの別のパッケージから同じ関数名をインポートしたときにどっちのパッケージの関数が優先されるのか.
  - 衝突するのでimportに失敗する．どちらかを完全修飾子で記述する必要がある．fu.pro4.enemy.Monsterみたいな.
- ActionListenerはメソッドしか入っていないということ？
  - そういうこと.
  - だってインタフェース（抽象メソッドのみを持つ）だもん.

質問への回答

- BoxLayoutのY_AXISとは何か？他にもあるなら教えて.
  - Y軸（縦方向）という意味．AXISは直訳すると軸．Xもある.
- Eclipseの予測変換に頼りきりでスペル正しく打てないのですがこの状態で大丈夫ですか.
  - 開発現場では予測変換をガンガン使うので問題ない.
  - テストではスペルは基本的には問わない予定である.
  - 社会に出た後のために，よく使うメソッドくらいは暗記したい.
- Layoutでわけたコンテナのサイズは変えれますか？
  - コンテナは変えられないがコンポーネントは変えられる.

```
panel.setPreferredSize(Dimension d);
```

質問への回答

- GUI使いにくくない？
  - 開発はしにくいけどユーザは使いやすいはず.
  - 現在はプログラムとデザインを切り分けることが主流なので，前回のようにプログラム内で配置等々を行うことは減っている.
- GUIのレイアウトには他にもあるのか.
  - 他にもある．興味のある人はAPIリファレンスへ.
  - 授業では時間の兼ね合いがあり必要最小限を教えている．発展的な内容を学習したい意欲的な学生は自学自習に励んでほしい.
- GUIの応用例はどのようなものがあるか.
  - 君たちが使ったことあるアプリはほぼ全てGUI.

質問への回答

- ウィンドウからウィンドウを呼び出す、とか、二つ目のウィンドウで何かしたら一つ目のウィンドウが影響を受ける、みたいなプログラムって作れますか？
  - 前者は前回の課題３の内容である．後者はフレームと閉じたというイベントを検出するListenerがあるので，イベント検出時に元フレームに対してコールバックすると実装できる．
- GUIをどうやって応用するか
  - 今週やるMouseListenerや授業でやらないKeyListenerでキータイプを検出できれば，ゲーム等も作れる．アイデア次第．

質問への回答

- コンテナは一つのフレームに対し、１つだけなのか？
  - その通り．なのでJPanelをうまく使おう．
- コンポーネントを自分で開発することは可能か．
  - コンポーネントも継承できるので，コンポーネントのコンストラクタに細々とした設定を内包したりできる．
  - ボタンに画像を表示したり，デザインやサイズも変えられる．
- パネルは無限に作れるのか．パネルの使い方を知りたい．
  - 無限に作れる（メモリある限り）．
  - パネルは数種類のLayoutを入れ子にして使うことが多い．

質問への回答

- ボタンの位置をもっと自由に動かすことはできませんか。
  - 実は座標で指定することができる．以下参考 https://www.javadrive.jp/tutorial/nulllayout/index1.html
- メソッドにパブリックをつけなかったらどうなるのか（修飾子なしのメソッドとは）
  - 修飾子なし＝同じパッケージからのみアクセスできる.
- 現在、主流のライブラリはなんですか
  - ライブラリは種類が多すぎるので何とも言えない.
  - 企業それぞれで使うフレームワークが違ったりするので，まずは基礎を知り，就職先の独自文化は就職後に学ぶとよい.

質問への回答

- 採点は部分点などはなし？
  - 採点基準を複数設けて自動採点している．（例えば課題１の中でもmethodA()ができているか，Bができているか等，これはいわゆる部分点ではないだろうか．）
- 自由課題はどのような採点方式をとるのか
  - 全員で主観的に採点＋長谷川の個人採点　にしようかなぁ．
  - 採点基準は面白さ，技術的難しさ，アピール力で見ると思う．
- 自由課題でハトヤマスタンプを使うには．
  - その場でUSBで渡すかも．
  - なお使ったところで点数は伸びないつもりでいてほしい！

本講義の概要

前半		後半	
第1回	基本文法の復習	第9回	標準ライブラリ
第2回	クラス～カプセル化の復習	第10回	ファイル入出力
第3回	抽象クラス，インタフェース	第11回	デバッグ，インポート，高速化
第4回	ポリモーフィズム	第12回	オブジェクト指向
第5回	GUI 1	第13回	自由開発演習 1
第6回	**GUI 2**	第14回	自由開発演習 2
第7回	スレッド，例外処理	第15回	自由開発演習発表会
第8回	ジェネリクス，コレクション	第16回	期末試験

何を開発するか，少しずつ考えておくこと

本日の目標

概要

プログラミングでお絵かきする.

目標

意図した図形をプログラムで描画できる.
マウスイベントを使いこなせる.

本日の提出課題

講義パート

課題を意識しながら
講義を聞くと良い.

課題 1

本日の授業を聞いて,
よくわかったと思う内容を2点簡潔に述べよ.

課題 2

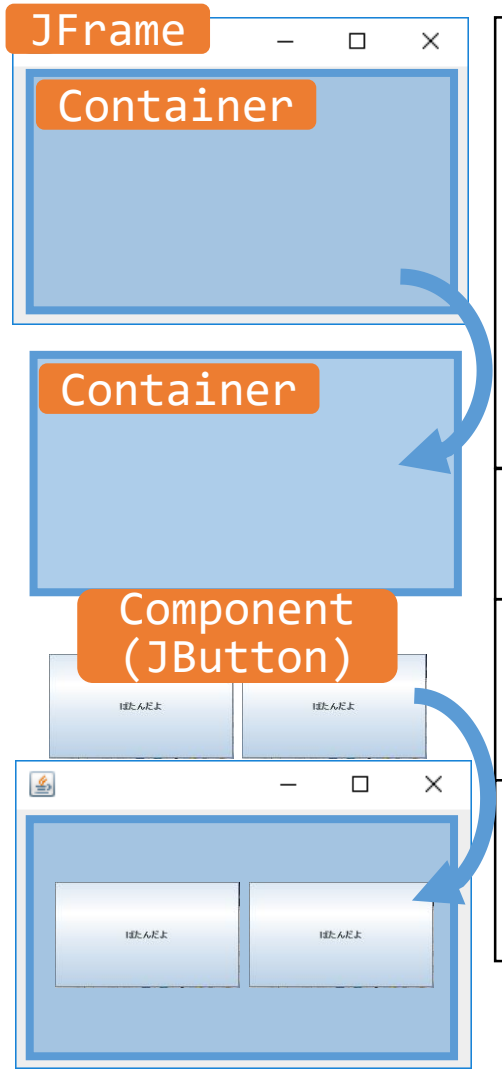
本日の授業を聞いて,
質問事項または**気になった点**を1点以上簡潔に述べよ.

課題 3

感想（あれば）

プログラミングでお絵かきする

前回の復習と今日学ぶこと




```java
// JFrame型のインスタンスを生成する
JFrame frame = new JFrame();
// このフレームの×ボタンを押したときの動作を設定する
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
// このフレームのサイズ（Width[px], Height[px]）を設定する
frame.setSize(300, 200);
// このフレームの表示をtrueにする
frame.setVisible(true);
```

```java
// コンテナの取得
Container c = frame.getContentPane();
```

```java
// ボタンコンポーネントの生成
JButton button1 = new JButton("ボタン1");
JButton button2 = new JButton("ボタン2");
```

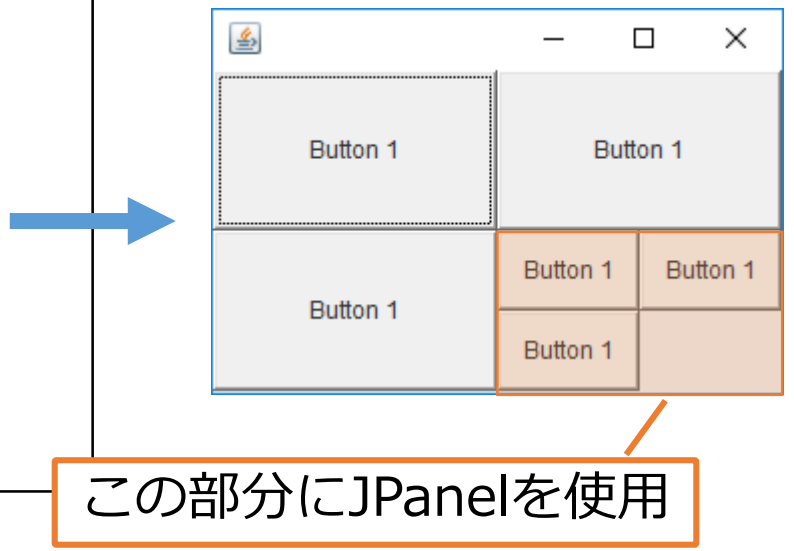
```java
// ボタンをコンテナに追加
c.add(button1);
c.add(button2);
```

プログラミングでお絵かきする

前回の復習と今日学ぶこと

コンポーネントを乗せることができる（コンテナのような）コンポーネントを**パネル(JPanel)**と呼ぶ.

```
frame.getContentPane().setLayout(new GridLayout(2,2));
frame.getContentPane().add(new Button("Button 1"));
frame.getContentPane().add(new Button("Button 1"));
frame.getContentPane().add(new Button("Button 1"));
JPanel panel = new JPanel();
panel.setLayout(new GridLayout(2,2));
panel.add(new Button("Button 1"));
panel.add(new Button("Button 1"));
panel.add(new Button("Button 1"));
frame.getContentPane().add(panel);
```



本日は**このJPanel**を改造してお絵かきを実現する.

プログラミングでお絵かきする

前準備

Main.javaのmain()メソッドでJFrameインスタンスを作成し，JPanelを継承したMyPanelインスタンスを追加して表示する．このMyPanelが今日の本題となる．

```
public static void main(String[] args){
  SwingUtilities.invokeLater(new Runnable() {
    public void run() {
      JFrame frame = new JFrame();
      // フレームの設定関連
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(400, 300);
      MyPanel panel = new MyPanel();
      frame.getContentPane().add(panel);
      frame.setVisible(true);
    }
  });
}
```

main()で書く時のおまじない

JPanelを継承したMyPanelをこれから開発していく

For Step Up!!

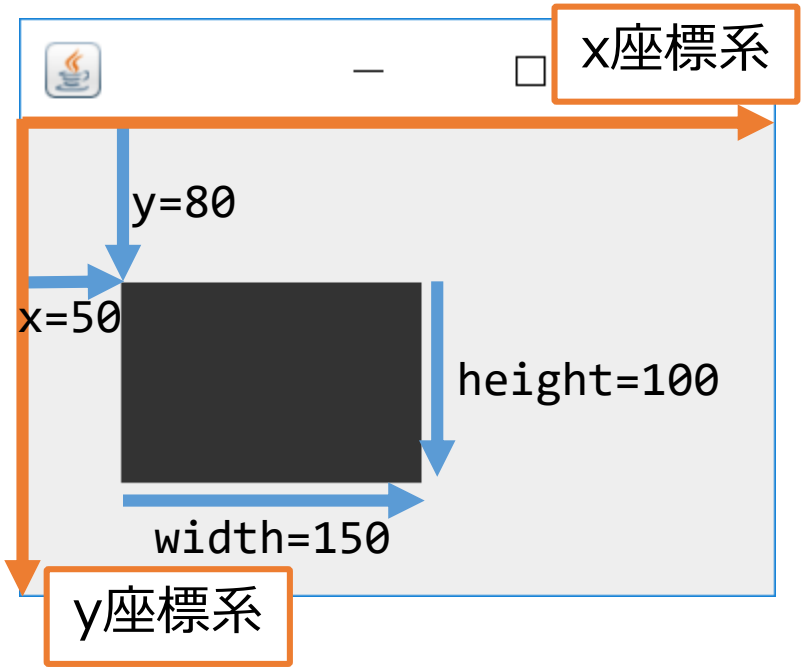
先人が開発したContainerインスタンスがなぜ我々が開発したMyPanel型を引数として受け入れることができるのか？

プログラミングでお絵かきする

図形をプログラミングする

JPanelを継承したMyPanel

```
public class MyPanel extends JPanel{
  @Override
  public void paintComponent(Graphics g){
    // 座標(50, 80)に150*100の四角形を描画
    // fillRect(x, y, width, height);
    g.fillRect(50, 80, 150, 100);
  }
}
```



paintComponent(Graphics g)は**JPanel**作成時に呼び出されるメソッドである.
→これをオーバーライドすることで，　**JPanel**に独自のお絵かきを実装することができる！

プログラミングでお絵かきする

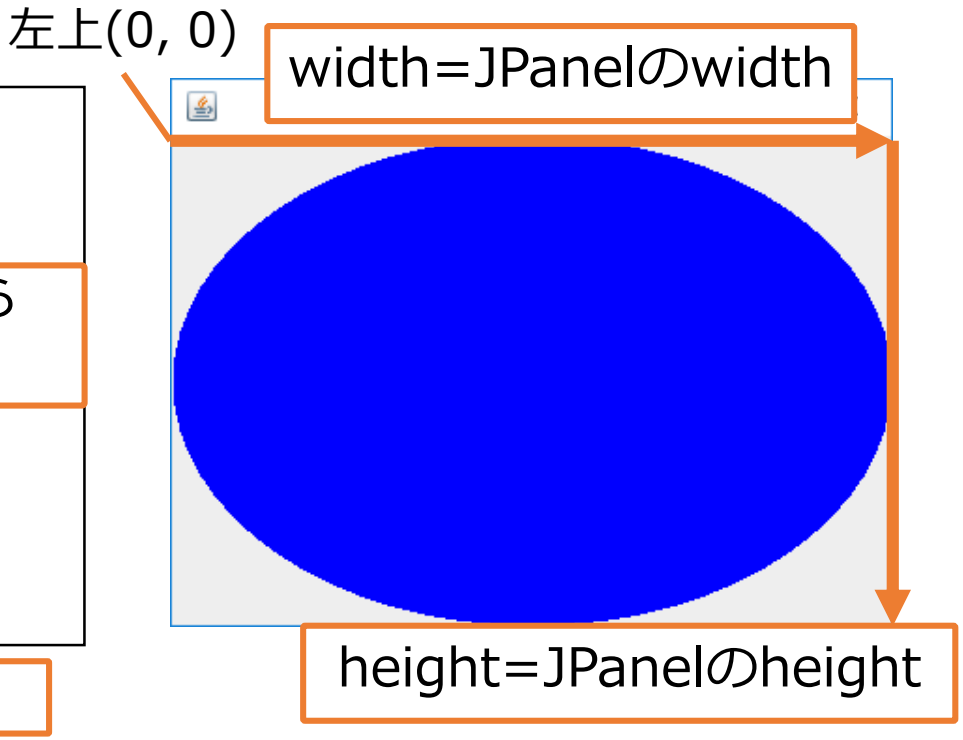
図形をプログラミングする

JPanelのサイズ（描画領域のサイズ）は**this.getSize()**を用いてDimension型で取得する事ができる.

```
@Override
public void paintComponent(Graphics g){
  // Dimension型で自身(JPanel)のサイズを取得する
  Dimension d = this.getSize();
  // 以降の描画色を青色に変更する
  g.setColor(Color.BLUE);
  // 左上(0, 0)でJPanel自身と同じサイズの長方形に
  // 内接する塗りつぶし楕円を描画する
  g.fillOval(0, 0, (int)d.getWidth(), (int)d.getHeight());
}
```

1．色を選んでから
2．描く

描画領域のサイズが分かると色々と応用が効く



プログラミングでお絵かきする

図形をプログラミングする

メソッド	説明
`void fillRect(int x, int y, int w, int h)`	座標(x, y)を左上として，サイズ(width*height)の塗りつし四角を描画する(drawRect()もある)
`void drawLine(int x1, int y1, int x2, int y2)`	座標(x1, y1)から(x2, y2)まで直線を描画する
`void fillArc(int x, int y, int w, int h, int degree1, int degree2)`	座標(x, y)を左上として，サイズ(width*height)の四角に内接する塗りつぶし円の一部を描画する(描画範囲はdegree1～degree2の角度に収まる範囲)(drawArc()もある)
`void fillOval(int x, int y, int w, int h)`	fillArc()の角度指定なし(drawOval()もある)
`void drawString(String str, int x, int y)`	座標(x, y)に文字列strを描画
`void fillPolygon(int x[], int y[], int n)`	n個の頂点を持つ多角形を描画．頂点の座標はx[]とy[]で指定．(drawPolygon()もある)
`void setColor(Color c)`	以降の描画色を変更する

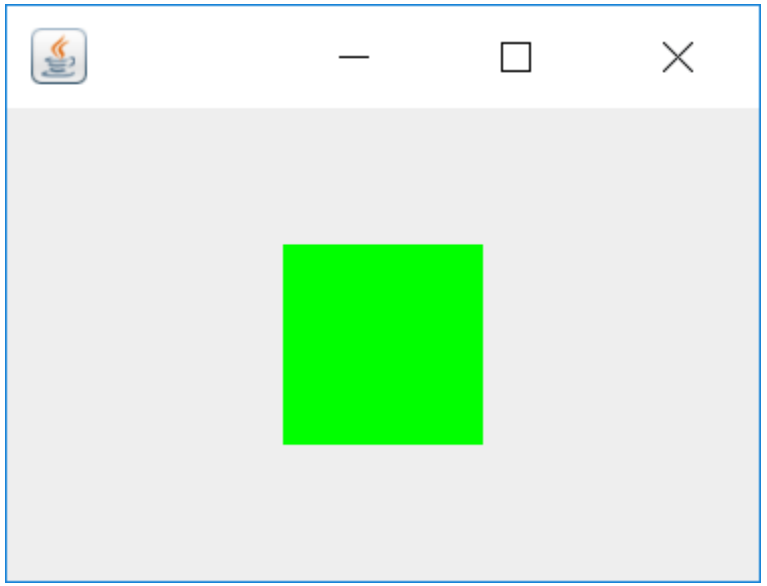
For Step Up!!

Graphicsにはまだ他にもメソッドがあるのでAPIリファレンスを参照しよう．
Graphics gをGraphics2Dにキャストして使用すると更に多くのメソッドがあるのでこれも調べてみよう．

プログラミングでお絵かきする

練習問題１：はじめてのプログラミングお絵描き

下図のように，フレーム**中央**に緑色の100*100の四角を描画せよ.



プログラミングでお絵かきする

練習問題２：プログラミングお絵描きの醍醐味

以下のプログラムの動作を確認せよ.

```
public class MyPanel extends JPanel{
  @Override
  public void paintComponent(Graphics g){
    for(int i=0;i<360;i+=30){
      int x = 200 + (int)(100*Math.cos(Math.toRadians(i)));
      int y = 120 + (int)(100*Math.sin(Math.toRadians(i)));
      g.fillOval(x - 10, y - 10, 20, 20);
      g.drawLine(200, 120, x, y);
    }
  }
}
```

プログラミングでお絵かきする
マウスイベント

JPanelはマウスイベントを検出できる.

```
public class MyPanel extends JPanel implements MouseListener, MouseMotionListener{
  public MyPanel(){
    this.addMouseListener(this);  this.addMouseMotionListener(this);
  }
  @Override  // マウスクリック時 (MouseListener)
  public void mouseClicked(MouseEvent e) {}
  @Override  // マウス押下時 (MouseListener)
  public void mousePressed(MouseEvent e) {}
  @Override  // マウス押下後解放時 (MouseListener)
  public void mouseReleased(MouseEvent e) {}
  @Override  // マウスがパネルに入った時 (MouseListener)
  public void mouseEntered(MouseEvent e) {}
  @Override  // マウスがパネルから出た時 (MouseListener)
  public void mouseExited(MouseEvent e) {}
  @Override  // マウスをドラッグした時 (MouseMotionListener)
  public void mouseDragged(MouseEvent e) {}
  @Override  // マウスを動かした時 (MouseMotionListener)
  public void mouseMoved(MouseEvent e) {}
}
```

この二つを実装する

この二つをリスナ登録する

イベント検出時の処理を実装する

プログラミングでお絵かきする

マウスイベント（あえてリスナを別クラスにした場合）

```
public class MyPanel extends JPanel{
  public MyPanel(){
    MyMouseListener listener = new MyMouseListener();
    this.addMouseListener(listener);
  }
}
```

MouseListenerを実装したインスタンスを登録すると，マウスイベントのコールバックを受け取れる．

```
public class MyMouseListener implements MouseListener{
  @Override
  public void mouseClicked(MouseEvent e) {}
  @Override
  public void mousePressed(MouseEvent e) {}
  @Override
  public void mouseReleased(MouseEvent e) {}
  @Override
  public void mouseEntered(MouseEvent e) {}
  @Override
  public void mouseExited(MouseEvent e) {}
}
```

JPanelのaddMouseListener()ではMyMouseListener型ではなく，MouseListener型で引数を受け取る．（ポリモーフィズム）
これにより，左の5メソッドは必ず使えることが保証されるので，イベント検出時にコールバック呼び出しが可能となる

プログラミングでお絵かきする

マウスイベント（別クラスにせずに，まとめて書いた場合）

```
public class MyPanel extends JPanel implements MouseListener{
  public MyPanel(){
    MouseListener listener = new MyMouseListener() -> this;
    this.addMouseListener(listener);
  }


  @Override
  public void mouseClicked(MouseEvent e) {}
  @Override
  public void mousePressed(MouseEvent e) {}
  @Override
  public void mouseReleased(MouseEvent e) {}
  @Override
  public void mouseEntered(MouseEvent e) {}
  @Override
  public void mouseExited(MouseEvent e) {}
}
```

MouseListenerを実装したインスタンスを登録すると，マウスイベントのコールバックを受け取れる．

JPanelのaddMouseListener()では
~~MyMouseListener型ではなく，~~
MouseListener型で引数を受け取る．
（ポリモーフィズム）
これにより，左の5メソッドは必ず使えることが保証されるので，イベント検出時にコールバック呼び出しが可能となる

プログラミングでお絵かきする

練習問題３：マウスイベント

マウスイベント(MouseListenerとMouseMotionListener)を実装し，動作を確認せよ．動作確認には以下のソースコードを，イベント検出時に実行する処理として記述して，各イベントがどのタイミングで発生するのかを体験せよ．

```
Graphics g = this.getGraphics();
g.fillOval(e.getX()-5, e.getY()-5, 10, 10);
```

パネルからGraphicsをいつでも受け取れる．

MouseEventの引数のe

マウスの座標X, Y

プログラミングでお絵かきする

ダブルバッファリング

本日の演習課題３では「画像がマウスカーソルについてくる」という機能を開発する．実際の動作を見てみよう．

プログラミングでお絵かきする
ダブルバッファリング

JavaのGUIでは「動き」を実装したい時，

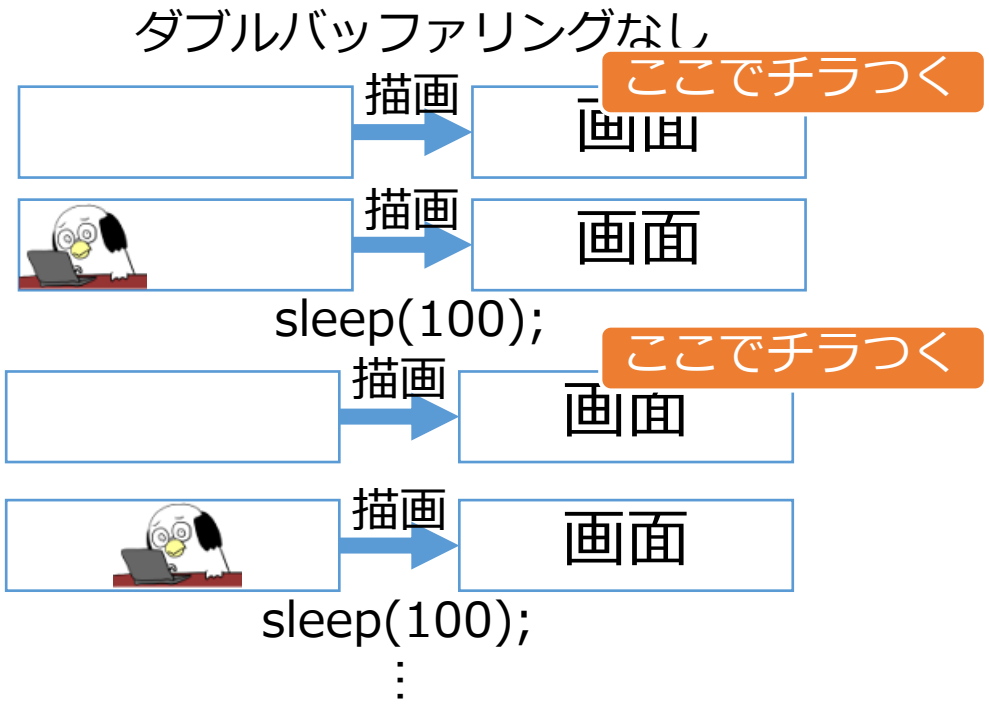
1. 背景色で全体を塗りつぶし
2. 動いた後の描画

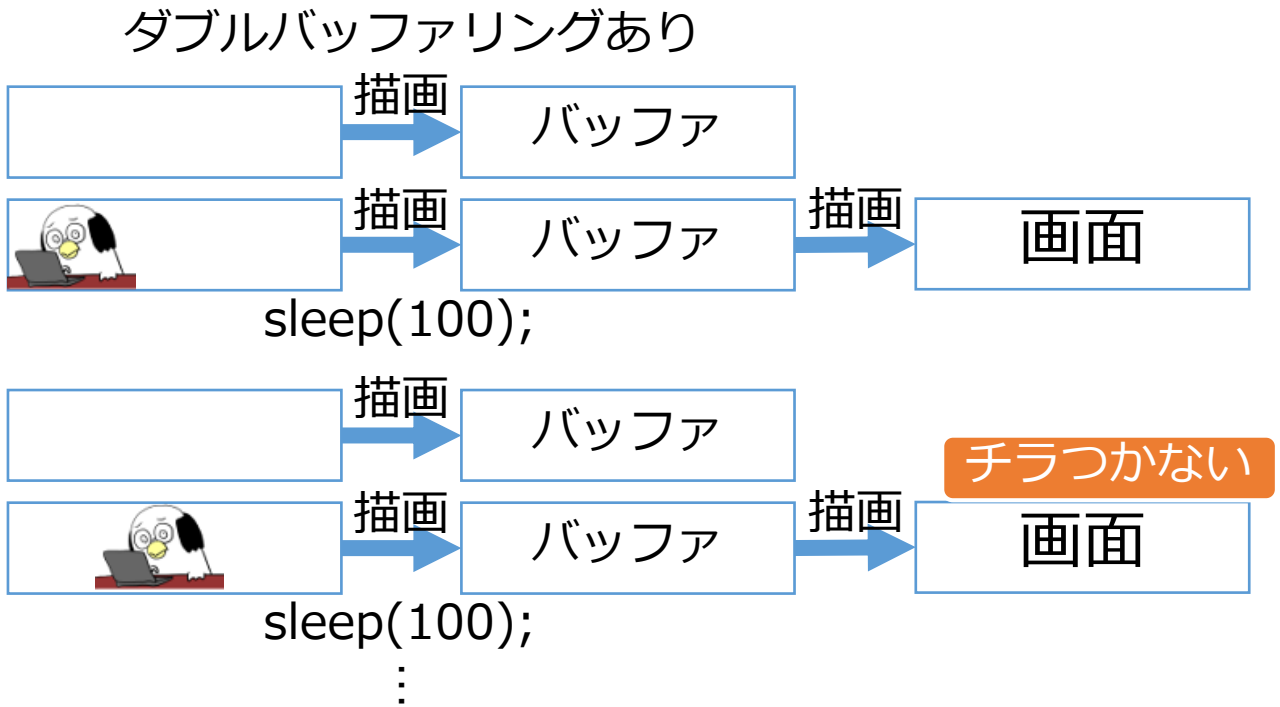
を高速に切り替えることでアニメーションを実現する．

すると「１のみが描画されている状態」を利用者が視認できるタイミングがあり，「動き」を表現したい対象がチラついて見えることがある．チラつきを改善する技術として，**ダブルバッファリング**という方法がある．

プログラミングでお絵かきする
ダブルバッファリング

ダブルバッファリングは，前述の１-２を直接画面に描画せず，一旦バッファに描画し，バッファに描画が完了した段階で，バッファを画面に表示させる手法である.





プログラミングでお絵かきする
ダブルバッファリング

ダブルバッファリングをプログラムで実装するには,

1. バッファイメージを作成する
```
Image buffer = this.createImage(400, 300);
```
2. バッファイメージのGraphicsを取得する
```
Graphics gbuf = buffer.getGraphics();
```
3. バッファイメージに描画処理を記述する
```
gbuf.fillOval(0, 0, 10, 10);
```
4. 画面にバッファイメージを描画する
```
g.drawImage(buffer, 0, 0, this);
```

となる．が．．．

プログラミングでお絵かきする
ダブルバッファリング

実はSwingはダブルバッファリングを**標準実装している**.
（Swingの前身であるAWTではダブルバッファリングを自分で実装する必要があった）

Q)ではなぜ先程の例ではチラつきが起こったのだろうか？

A)mouseMoved()内で描画処理を行っていたから.

→**paintComponent()内での描画処理のみ**ダブルバッファリングに標準対応している.
→そもそも本来の使い方として，描画処理はpaintComponent()以外に記述されるべきではない.

プログラミングでお絵かきする
ダブルバッファリング

Swingでダブルバッファリングさせるには以下の通り.

1. mouseMoved()では座標取得＋**repaint()**
2. paintComponent()内で描画処理を実装

```
private int lastX = -1, lastY = -1;
@Override  // 座標の取得＋repaint()
public void mouseMoved(MouseEvent e) {
  this.lastX = e.getX();  this.lastY = e.getY();
  this.repaint();
}
@Override
public void paintComponent(Graphics g){
  Dimension d = this.getSize();
  g.setColor(Color.WHITE);
  g.fillRect(0, 0, (int)d.getWidth(), (int)d.getHeight());
  g.setColor(Color.BLACK);
  g.fillOval(this.lastX-5, this.lastY-5, 10, 10);
}
```

this.repaint()はJPanelの再描画を促すメソッドである.

this.repaint()を明示しないと，座標が変更されても画面上何も変化しない.

プログラミングでお絵かきする
まとめ

- JPanelのpaintComponentをOverrideすることで，様々な図形をプログラムで描画することができる.
- MouseListenerとMouseMotionListenerでマウスの動きを検出することができる.
- ダブルバッファリングという手法がある.
  （今後実装する機会はないかもしれないが，このような技法があるということは知っておくべきである.）

次週予告

※次週以降も計算機室

前半

マルチスレッド，例外処理といった，
Javaを扱うなら知っておいてほしい標準機能を学ぶ.

後半

講義内容に関するプログラミング演習課題に取り組む.

本日の提出課題

講義パート

課題 1

本日の授業を聞いて，
よくわかったと思う内容を
2点簡潔に述べよ.

課題 2

本日の授業を聞いて，
質問事項または**気になった点**
を1点以上簡潔に述べよ.

課題 3

感想（あれば）

課題 4

なし

演習

- 昼休み，いつものWebページに演習問題をPDFで演習問題をアップロードする．各自実施してプロII同様のWebページから提出すること.
- 質問は３人体制で受け付けるので遠慮なく申し出る．質問の際は，どこまでわかっていて何がわからないのかを申し出ること.
- （ないとは思うが）コピペは発覚次第両成敗する.
  - ✓ {コピペ，カンニング} ∈不正行為
- つまらないミスも今回は問答無用で×とするので，最終チェックを怠らないこと．（昨年は目視で甘めに採点していたが，自動採点を開発している意味がないので．．．）