情報処理技法
(Javaプログラミング)2

第13回
絵を描いてみよう!

人間科学科コミュニケーション専攻
白銀 純子

第13回の内容

- プログラムでのお絵描きの基礎
  - 準備編
  - お絵描き編

前回の復習問題の解答

- 下記は、JFileChooserを表示させ、選択されたファイルを開く処理である。下記の処理の記述で間違っている箇所(1箇所)はどこか、間違いの種類は、コンパイルエラー・例外・処理の内容違いのどれか、そしてどのように修正すれば良いかを答えなさい。

```
JFileChooser fileSelection = new JFileChooser();
int approve = fileSelection.showOpenDialog(null);
if (approve == 1) {
  File readFile = fileSelection.getSelectedFile();
  try {
    FileReader fr = new File
    BufferedReader br = ne

    while (br.ready()) {
      System.out.println(br
    }
  }
  catch(IOException ioe) {
  }
}
```

JFileChooserで「開く」ボタンを押したときの戻り値は「0」なので、この部分は「0」に修正すれば良い。間違いの種類は、「処理の内容違い」である。

準備編

お絵描きをするには?

- プログラムでの「紙」にあたるもの: JFrameまたはJPanel
- JFrameやJPanelの上に、座標を指定して、絵を描いていく
  - 楕円
  - 四角形
  - 線
  - etc.
- 絵を描くには、JPanelの方がオススメ

「紙」の準備(1)

- 「ただの」JPanelの上に絵を描くと、ウィンドウを操作すると、絵が消えてしまう
  - ウィンドウの最小/最大化、別のウィンドウを重ねる, etc.
- JPanelは画面に表示された瞬間に、JPanelの「paintComponent」メソッドで、絵が描かれる
  - 何も設定をしていなければ、JPanelをグレーに塗りつぶすだけ

※これらは、JFrameも同様

「紙」の準備(2)

- ウィンドウの操作がされるたびに「paintComponent」が呼び出される
  - グレーに塗りつぶされる = 描いた絵が消える
- 描いた絵が消されないようにするには?
  - 「paintComponent」を定義しなおす

    =ウィンドウの操作がされると、定義しなおされた「paintComponent」が呼び出される

    =ウィンドウの操作がされると、もう一度同じ絵が描かれる

    =絵が消えない
  - JPanelを継承して「paintComponent」をオーバーライド

JPanelを継承して…?(1)

- ファイルを1つ作成する

JPanelを継承したクラス

```
import  java.awt.*;
import  java.io.*;
import  java.lang.*;
import  javax.swing.*;

public  class  クラス名  extends  JPanel {
   public  void  paintComponent(Graphics  g) {
      
   }
}
```

JPanelの継承

JPanel上に絵を描くメソッド

JPanelに描く絵の内容

JPanelを継承して…?(2)

JPanelを継承したクラス

```
import  java.awt.*;
import  java.io.*;
import  java.lang.*;
import  javax.swing.*;

public  class  クラス名  extends  JPanel {
   public  void  paintComponent( Graphics  g ) {
```

- 「g」は、「Graphics」というクラスのオブジェクトで、JPanel専用のペンの役割をする
- 絵を描くときは、「ペン」に対して、どのような絵を描くか命令をする

プログラム本体は?

- 書き方は通常のGUIプログラムと同じ
- 絵を描くために使うJPanelのみ、自分で作成したクラスを利用すること

```
public 本体のクラス名 extends JFrame {
  JFrame frame;
  JButton but1, but2;
  描画用パネルクラス名 canv;
  public 本体のクラス名 {
    ......
    canv = new 描画用パネルクラス名();
    ......
  }
  public static void main(String[] args) {
    new 本体のクラス名();
  }
}
```

お絵描き編

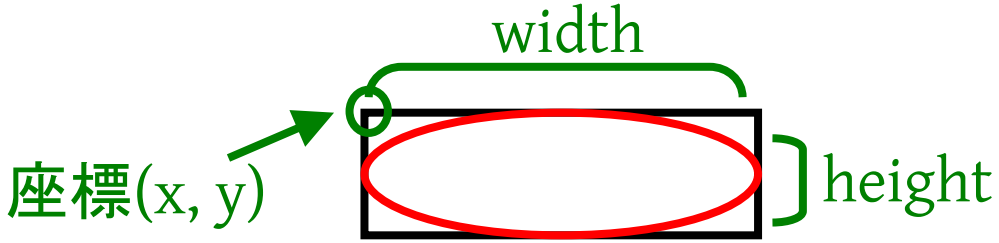
お絵描きのためのメソッド

- 絵を描くときは、JPanel専用の「ペン」に対する命令を記述
  →「ペン」の役割をする「Graphics」クラスに、絵を描くためのメソッドが用意されている
  - 「draw」で始まるメソッド: 四角形や円などを塗りつぶさずに描く
  - 「fill」で始まるメソッド: 四角形や円などを塗りつぶして描く
- 絵を描くときは、各部品の左上隅の点の座標と縦横の大きさを指定
- X座標は右方向、Y座標は下方向

メソッド例(1)

- drawRect(int x, int y, int width, int height)
  - 四角形を描く(fillRectも同様)
  - x, y: 左上の点のx座標とy座標
  - width, height: 四角形の幅と高さ
- drawOval(int x, int y, int width, int height)
  - 楕円を描く(fillOvalも同様)
  - x, y: 左上の点のx座標とy座標
  - width, height: 楕円の幅と高さ

ただし、楕円を四角形で囲んだときの、四角形の座標と幅・高さ



メソッド例(2)

- drawString(String str, int x, int y)
  - 文字列の描画
  - str: 描画する文字列
  - x, y: 文字列の左上の点のx座標とy座標
- draw3DRect(int x, int y, int width, int height, boolean raised)
  - 3Dで強調表示される四角形の描画(fill3DRectも同様)
  - x, y: 左上の点のx座標とy座標
  - width, height: 四角形の幅と高さ
  - raised:「true」で浮き出たように見え、「false」で彫り込まれたような感じ

メソッド例(3)

- drawLine(int x1, int y1, int x2, int y2)
  - 線の描画
  - x1, y1: 線の開始点のx座標とy座標
  - x2, y2: 線の終了点のx座標とy座標
- drawPolyline(int[] xPoints, int[] yPoints, int nPoints)
  - 折れ線の描画
  - xPoints, yPoints: 折れ線の開始点, 折れる点, 終了点を配列に格納したもの
  - nPoints: 配列の中からいくつの点を使って折れ線を描くか

メソッド例(4)

- drawPolygon(int[] xPoints, int[] yPoints, int nPoints)
  - 多角形の描画(fillPolygonも同様)
  - xPoints, yPoints: 多角形の折れ線の開始点, 折れる点, 終了点を配列に格納したもの
  - nPoints: 配列の中からいくつの点を使って多角形を描くか

メソッド例(5)

- drawArc(int x, int y, int width, int height, int startAngle, int arcAncle)
  - 楕円弧の描画(fillArcも同様)
  - x, y: 描画される孤の左上の点のx座標とy座標
  - width, height: 描画される孤の幅と高さ
  - startAngle: 開始角度
  - arcAngle: 開始角度に対する弧の展開角度の大きさ

メソッド例(6)

- drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)
  - 角の丸い四角形の描画(fillRoundRectも同様)
  - x, y: 描画される四角形のx座標とy座標
  - width, height: 描画される四角形の幅と高さ
  - arcWidth: 4 隅の弧の水平方向の直径
  - arcHeight: 4 隅の弧の垂直方向の直径

メソッド使用例

```
public void CanvasPanel extends JPanel {
  public void paintComponent(Graphics g) {
    g.fillRect(5, 5, 390, 290); /* 楕円の描画 */

    g.fillOval(5, 5, 390, 290); /* 楕円の描画 */

    int[] x1 = {200, 5, 390};  /* 三角形の頂点のx座標の配列 */
    int[] y1 = {5, 150, 150};  /* 三角形の頂点のy座標の配列 */
    g.fillPolygon(x1, y1, 3); /* 三角形の描画 */

    int[] x2 = {200, 5, 390}; /* 三角形の頂点のx座標の配列 */
    int[] y2 = {290, 150, 150}; /* 三角形の頂点のy座標の配列 */
    g.fillPolygon(x2, y2, 3);  /* 三角形の描画 */
  }
}
```

色をつける

- 「setColor」というメソッドで、ペンの色を変更
- ペンの色は1度変えると、その後もずっと同じ色
  - ある部品を赤で描き、その次の部品を元の色で描きたい場合には、赤で部品を描いた後に、「setColor」でペンの色を黒に戻すことが必要
- 「setColor」の引数は「Color」というクラスのオブジェクト

色を使う

- 色を扱うクラス: Color
- 色の使い方その1: Javaで指定されている色を使う
  - Color.*色の名前*
- 色の使い方その2: RGBカラーを使う
  - 「new Color(int r, int g, int b)」で色を作成する
    - r: 赤成分(0～255)
    - g: 緑成分(0～255)
    - b: 青成分(0～255)

Javaでの色の名前

- Color.black
- Color.blue
- Color.cyan
- Color.darkGray
- Color.gray
- Color.green
- Color.lightGray
- Color.magenta
- Color.orange
- Color.pink
- Color.red
- Color.white
- Color.yellow

RGBカラー

- コンピュータの色: 赤(Red), 緑(Green), 青(Blue)の光から全ての色を表現する

RGBカラー

  - それぞれ256段階の濃淡があり、それを混ぜ合わせて色を表現する
  - 混ぜ合わせる赤, 緑, 青の濃さの段階を数値で表す
  - 255が最も濃く、0が最も淡い

例えば...

赤: 255, 緑: 0, 青: 0 → 

赤: 204, 緑: 153, 青: 255→ 

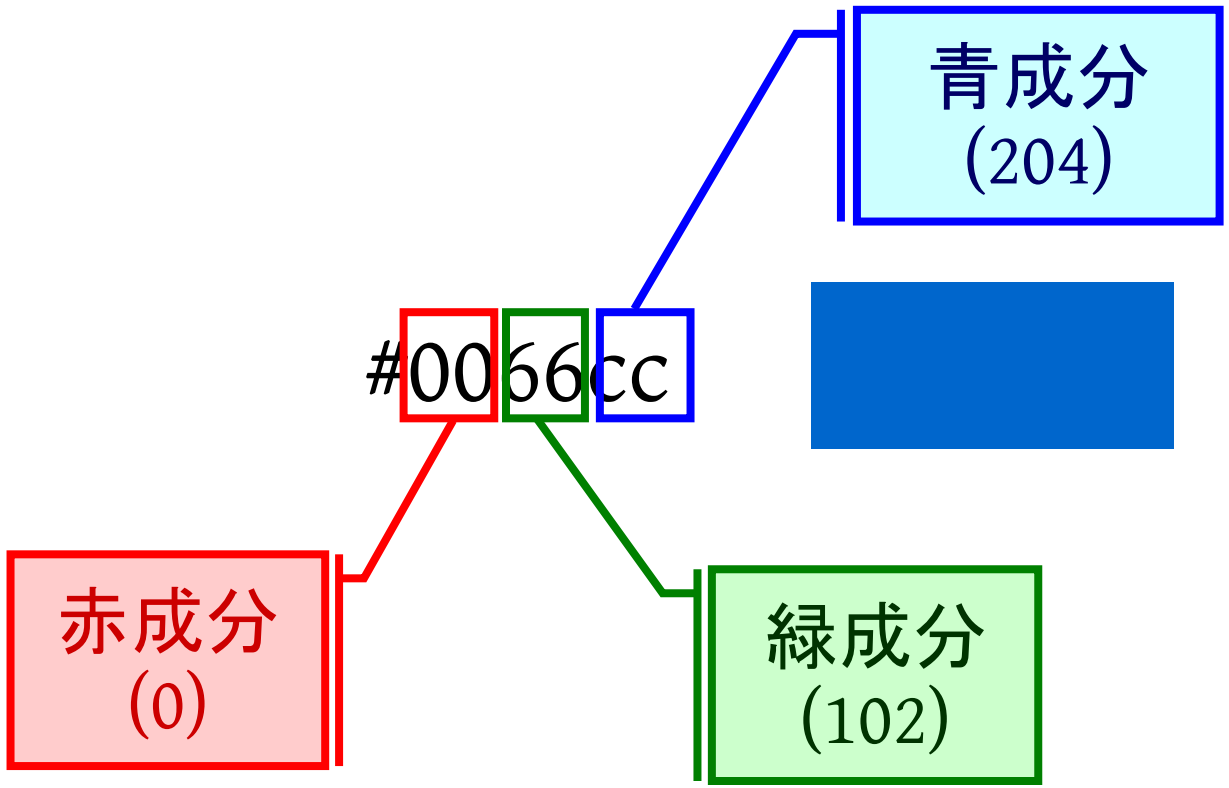
コンピュータでは:
赤, 緑, 青の256段階の濃淡の数値を16進数で表す

16進数(1)

- 数を16個の文字で表す方法
  - 普段は10進数(数を10個の文字で表す方法)
  - 10進数の「16」を、16進数では「10」と表記
- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, fの16個の文字で表現
  - 「a」が10進数の10,「b」が10進数の11,「c」が10進数の12,「d」が10進数の13,「e」が10進数の14,「f」が10進数の15

16進数での色の表現

- 「#」を最初につけ、R(赤)、G(緑)、B(青)の順に16進数で2桁ずつで表現

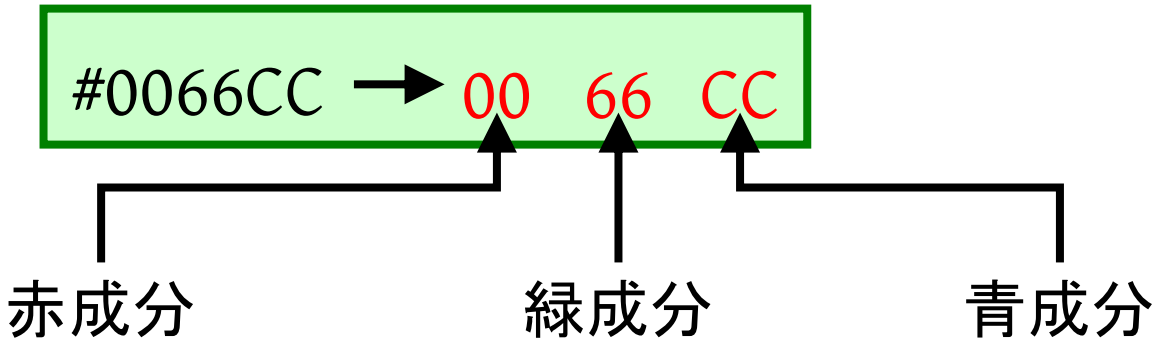


RGBカラーを使うには?

- 「色見本」などのキーワードで、色のサンプルリストを探す
- 色のサンプルリストから、好きな色を選択する
  - 色のサンプルリストは、「#」つきの色の名前が示されていることが多い
- 「#」つきの色の名前を、10進数の赤・緑・青の数値に直して、Colorクラスのコンストラクタに設定する

16進数の色の名前を10進数に直す

1. 「#」の後の6桁の16進数を2桁ずつに分解する
   - 左側から赤・緑・青の数値になる



16進数の色の名前を10進数に直す

2. アルファベットを10進数に直す

- A → 10
- B → 11
- C → 12
- D → 13
- E → 14
- F → 15

※アルファベットの大文字・小文字は関係なし

00 → 0 0
66 → 6 6
CC → 12 12

16進数を10進数に変換

3. 2. の数の1桁目の上に「16^0」、2桁目の上に「16^1」と書く

16^1	16^0
0	0

16^1	16^0
6	6

16^1	16^0
12	12

16進数を10進数に変換

4. 各桁の上の「16^n」と、それぞれの桁の数をかけあわせる

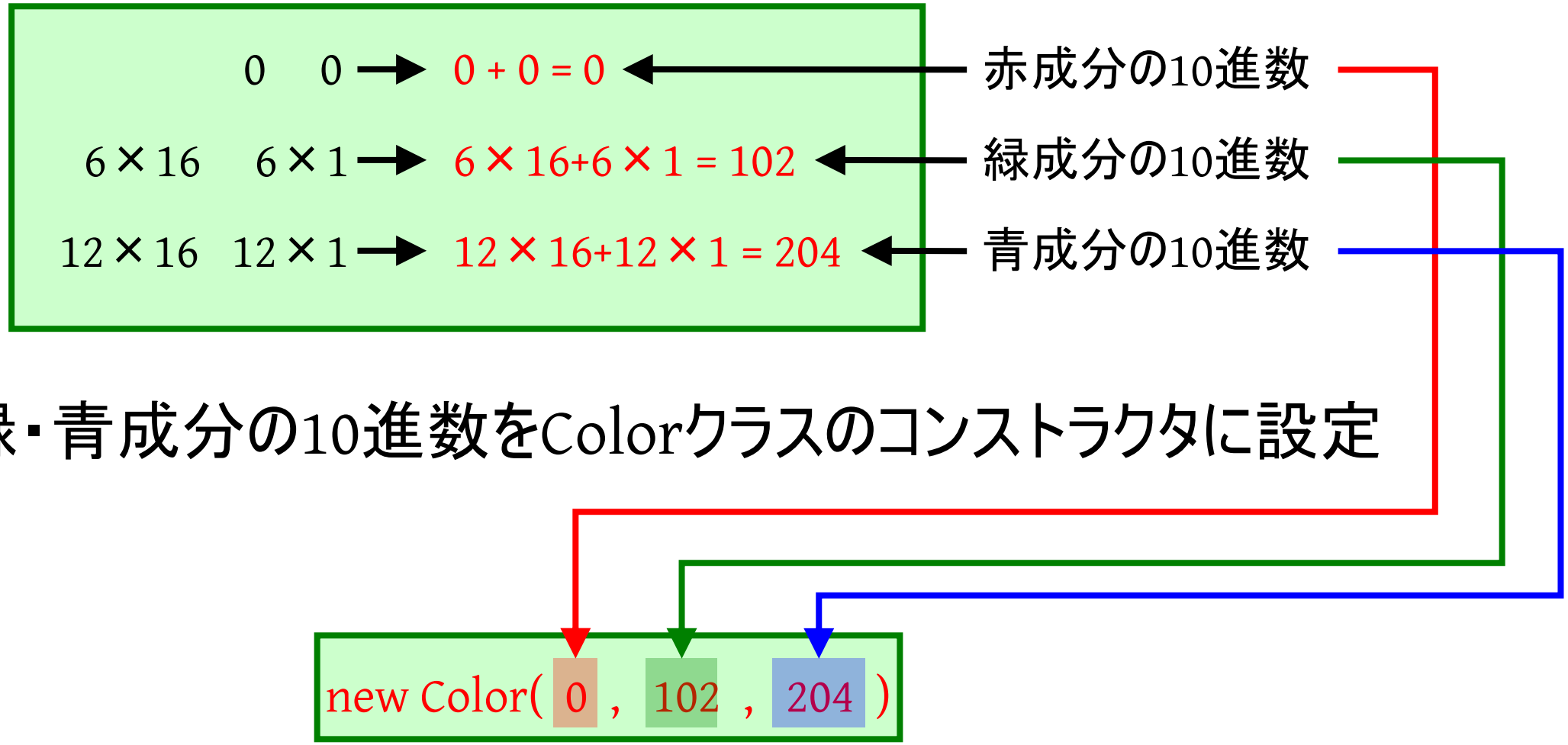
16^1 16^0
× × → 0 0
0 0

16^1 16^0
× × → 6×16 6×1
6 6

16^1 16^0
× × → 12×16 12×1
12 12

16進数を10進数に変換

5. 4. の各桁の数を足し合わせる



6. 赤・緑・青成分の10進数をColorクラスのコンストラクタに設定

色を使う(例)

キャンバスの「paintComponent」メソッド

```
public void paintComponent(Graphics g) {
   g.setColor(new Color(0, 0, 255)); /* 青色の四角形を描く */
   g.fillRect(5, 5, 390, 290);

   g.setColor(new Color(0, 255, 0));  /* 緑色の楕円を描く */
   g.fillOval(5, 5, 390, 290);

   int[] x1 = {200, 5, 390};
   int[] y1 = {5, 150, 150};
   g.setColor(Color.yellow); /* 黄色の三角形を描く */
   g.fillPolygon(x1, y1, 3);

   int[] x2 = {200, 5, 390};
   int[] y2 = {290, 150, 150};
   g.setColor(Color.cyan); /* シアン色の三角形を描く */
   g.fillPolygon(x2, y2, 3);
}
```

例～写して実行してみよう!～(1)

メインのプログラム

```
public class DrawPicture extends JFrame{
  CanvasPanel canv;

  public DrawPicture() {
    canv = new CanvasPanel();

    getContentPane().add(canv);
    setTitle("お絵かき");
    setSize(410, 340);
    setDefaultCloseOperation(Jframe.EXIT_ON_CLOSE);
    setVisible(true);
  }
  public static void main(String[] args) {
    new DrawPicture();
  }
}
```

例～写して実行してみよう!～(2)

描画領域のクラス

```
public class CanvasPanel extends JPanel {
  public void paintComponent(Graphics g) {
    g.setColor(new Color(0, 0, 255));
    g.fillRect(5, 5, 390, 290);

    g.setColor(new Color(0, 255, 0));
    g.fillOval(5, 5, 390, 290);

    int[] x1 = {200, 5, 390};
    int[] y1 = {5, 150, 150};
    g.setColor(Color.yellow);
    g.fillPolygon(x1, y1, 3);

    int[] x2 = {200, 5, 390};
    int[] y2 = {290, 150, 150};
    g.setColor(Color.cyan);
    g.fillPolygon(x2, y2, 3);
  }
}
```

補講と期末試験のお知らせ

- 補講: 1月20日(水) 5限 9301教室
  - 課題の質問受け付け
  - 出席課題はなし
- 期末試験: 1月26日(火) 3限 9301教室
  - 内容: 後期の講義内容すべて
    - 用語の意味の選択・説明
    - 概念に関する説明
    - 実技