オブジェクト指向プログラミング(2)

第1回

横山　孝典
E-mail: tyoko@tcu.ac.jp

本授業の概要

［科目概要］

オブジェクト指向プログラミングの基本概念とプログラミングの基本を学ぶ。

→　講義（4時限）

Javaによるオブジェクト指向プログラムの設計と実装を学習する。

→　演習（5時限）

［到達目標］

オブジェクト指向の概念を理解するとともに、実際にプログラミングを行うことで、オブジェクト指向の考え方と、Java言語の構文とその意味を理解し、プログラムを記述できるようになる。

参考書

- Javaによるオブジェクト指向プログラミング
  - Mat Weisfeld 著、萩本順三監訳、多摩ソフトウェア訳
    「JavaとUMLで学ぶオブジェクト指向の考え方」
    翔泳社
- Java言語
  - K.アーノルド、D.ホームズ、J.ゴスリン著、柴田芳樹訳
    「プログラミング言語Java」
    ピアソン・エデュケーション

参考）UML

- M.ファウラー、K.スコット著、
  羽生田栄一監修、多摩ソフトウェア訳
  「UMLモデリングのエッセンス」
  翔泳社

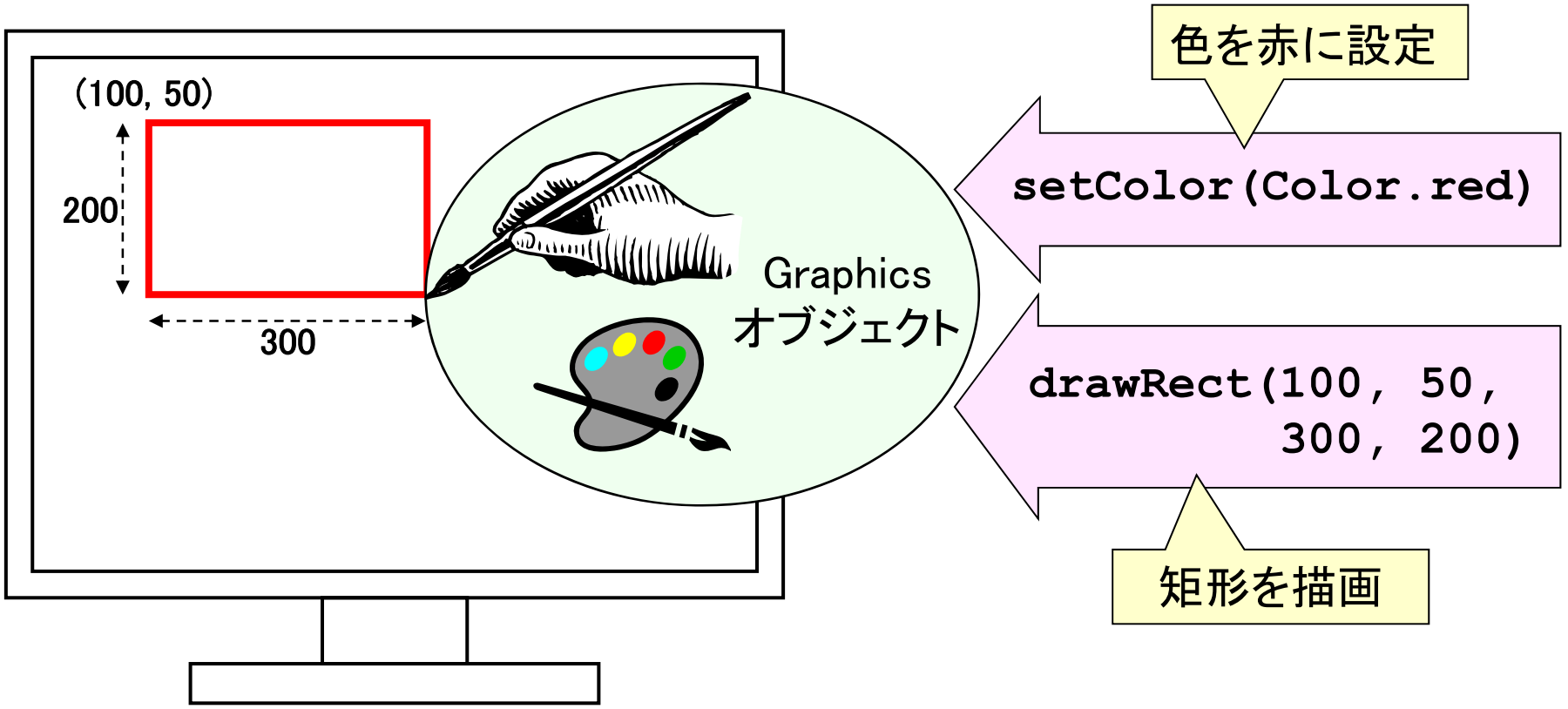
授業計画

1. GUIプログラミング --- 図形描画、APIの使用
2. クラス構成の設計
   --- デザインパターン、フレームワーク、イベント駆動プログラミング
3. インタフェース --- インタフェースの継承、多重継承
4. Javaプログラミング技法 --- パッケージ、型の互換性等
5. Java言語処理系
   --- 仮想マシン、ゴミ集め、修飾子、スコープ、例外処理等
6. オブジェクト指向設計 --- クラス設計の考え方
7. まとめ、試験 --- まとめと学生自身による達成度評価

Javaにおける図形描画（1）

図形描画のためのクラス“Graphics”

Graphicsオブジェクト（グラフィックスコンテキスト）による図形描画方法



Javaにおける図形描画（2）

クラス“Graphics”の代表的なメソッド

Graphics	

setColor(Color c): void　　// 引数はクラス“Color”のインスタンス
　　// 描画の色をcに設定（cは“Color.色”、色の例：black, blue,
　　// cyan, gray, green, magenta, orange, pink, red, white, yellow）
	描画の色を設定するメソッド（描画用メソッドを呼び出す前に設定しておく）

drawLine(int x1, int y1, int x2, int y2): void
　　// 始点(x1, y1)、終点(x2,y2)の線分を描画
drawRect(int x, int y, int width, int height): void
　　// 座標(x, y)、幅width、高さheightの矩形を描画
drawOval(int x, int y, int width, int height): void
　　// 座標(x, y)、幅width、高さheightの楕円を描画
fillRect(int x, int y, int width, int height): void
　　// 座標(x, y)、幅width、高さheightの塗りつぶし矩形を描画
fillOval(int x, int y, int width, int height): void
　　// 座標(x, y)、幅width、高さheightの塗りつぶし楕円を描画
	描画用のメソッド（図形によって異なるメソッドを呼び出す）

Javaにおける図形描画(3)

Graphicsオブジェクトを使って図形を描画するためには、ソースファイルの先頭で、以下を宣言する。

```
import java.awt.Graphics;    // クラスGraphicsを利用する
import java.awt.Color;       // クラスColorを利用する
```

Graphicsオブジェクトは、Javaのウィンドウ(フレーム(Frame))から取得するが、難しい(本授業の範囲を超える)ので、引数で与えることとする。

Graphics のメソッドの引数

setColor(Color c): void

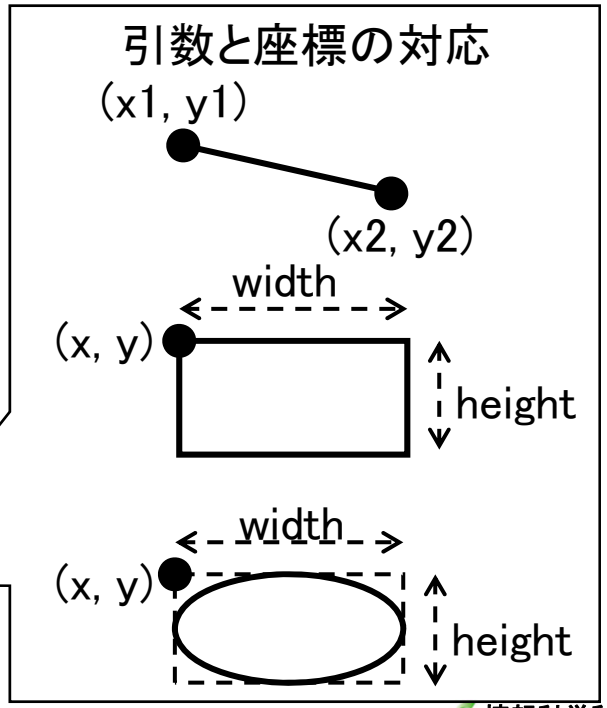
引数とする色の例

Color.black, Color.blue, Color.cyan, Color.gray, Color.green,
Color.magenta, Color.red, , Color.yellow

drawLine(int x1, int y1, int x2, int y2): void

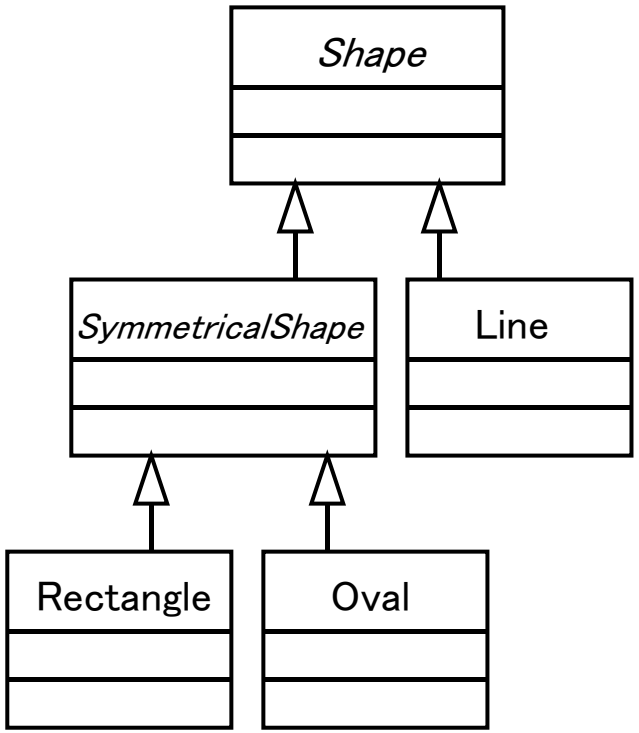
drawRect(int x, int y, int width, int height): void

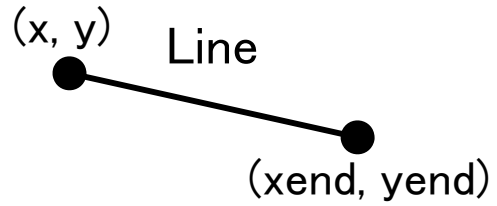
drawOval(int x, int y, int width, int height): void

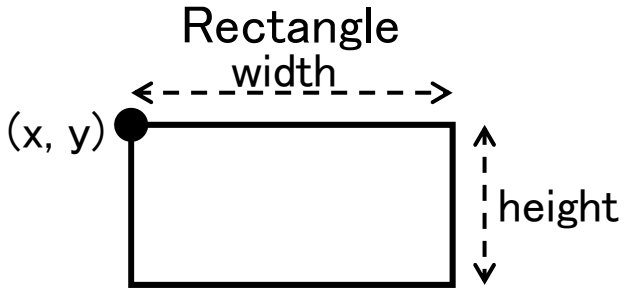


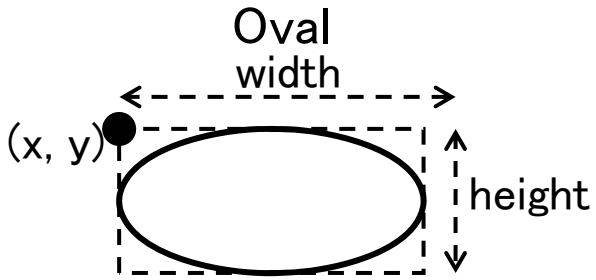
演習問題(1)

画面に表示可能な図形のクラスを作成する。
クラス構成は下記。









演習問題(2)

クラスShape(図形)、Line(直線(線分))のクラスは完成したものを与える。

SymmetricalShape(左右対称図形)、Rectangle(矩形)、Oval(楕円)のクラスは、テンプレート(未完成のクラス)を与えるので、必要なコードを追記して完成させる。

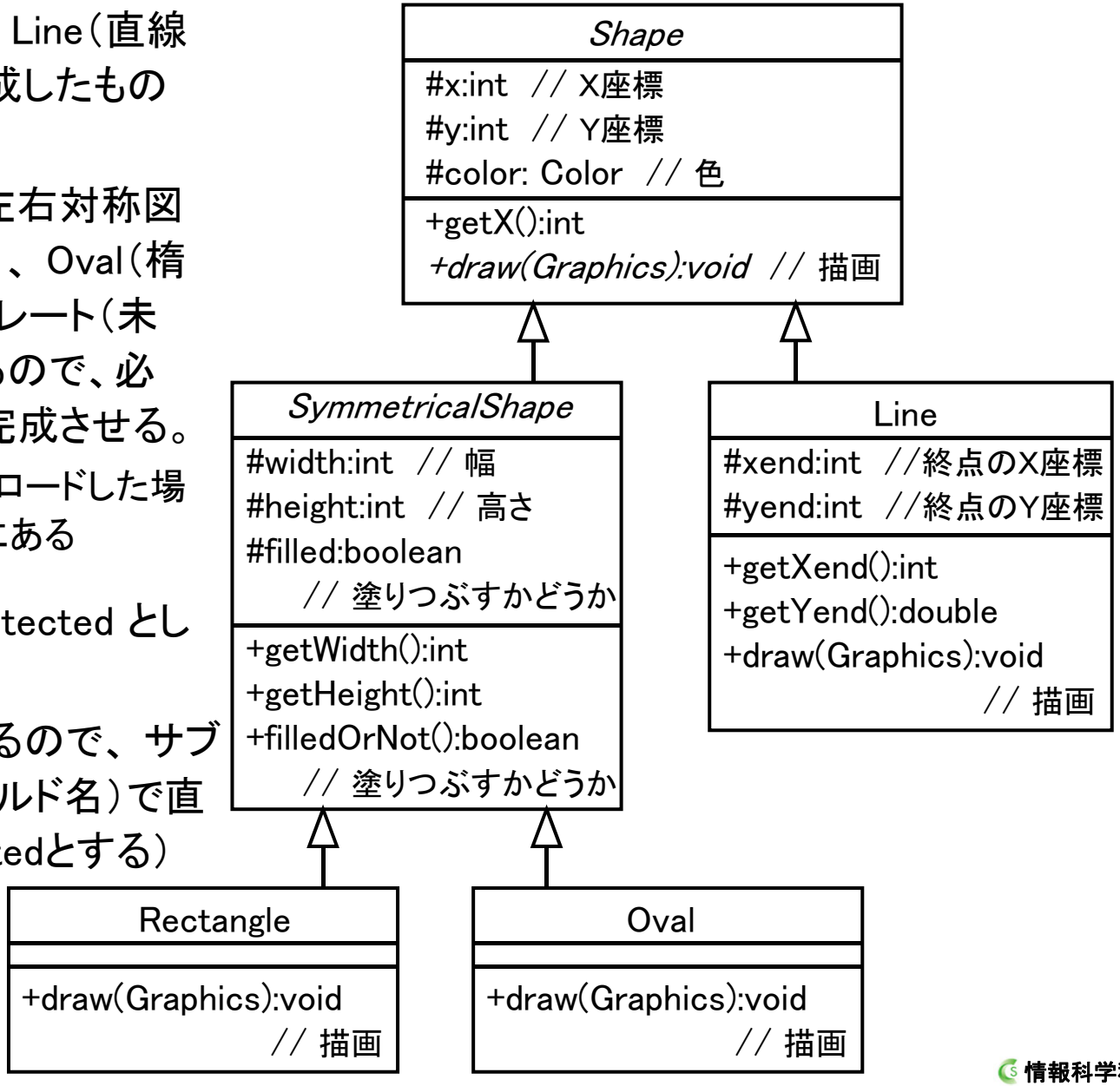
注)添付ファイルをダウンロードした場合、ディレクトリtemplateにある

属性(フィールド)は protected としている。

(今回から継承を多用するので、サブクラスから属性名(フィールド名)で直接アクセス可能なprotectedとする)

注)図では、コンストラクタは省略している

詳細クラス図



演習問題(3)

Graphicsオブジェクトを使って図形を描画するためには、
図形のクラスに以下の記述が必要

(1) ソースファイルの先頭で下記を宣言する

```
import java.awt.Graphics;     // クラスGraphicsを利用する
import java.awt.Color;        // クラスColorを利用する
```

(2) 描画用のメソッド“draw()”を作成する

```
// 図形描画用のメソッドの形式
public void draw(Graphics gra) {   // 引数はGraphicsオブジェクト
  gra.setColor( Color.色 );        // 描画する色を設定する
  gra.描画用メソッド( 引数の並び );   // 図形を描画する
}
```

drawLine, drawRect, drawOval, fillRect, fillOval など

演習問題(4)


```
// Shape.java

import java.awt.Graphics;  // クラスGraphicsを利用する
import java.awt.Color;     // クラスColorを利用する

// 図形のクラス
public abstract class Shape {
  int x = 0;                        // X座標
  int y = 0;                        // Y座標
  Color color = Color.black;        // 図形の色(初期値は黒)

  // コンストラクタ
  // 引数:X座標、Y座標、色
  public Shape(int xx, int yy, Color c) {
    x = xx;
    y = yy;
    color = c;
  }

  public int getX() {
    return x;
  }

  public int getY() {
    return y;
  }

  // 図形を描画するメソッド
  public abstract void draw(Graphics gra);

}
```

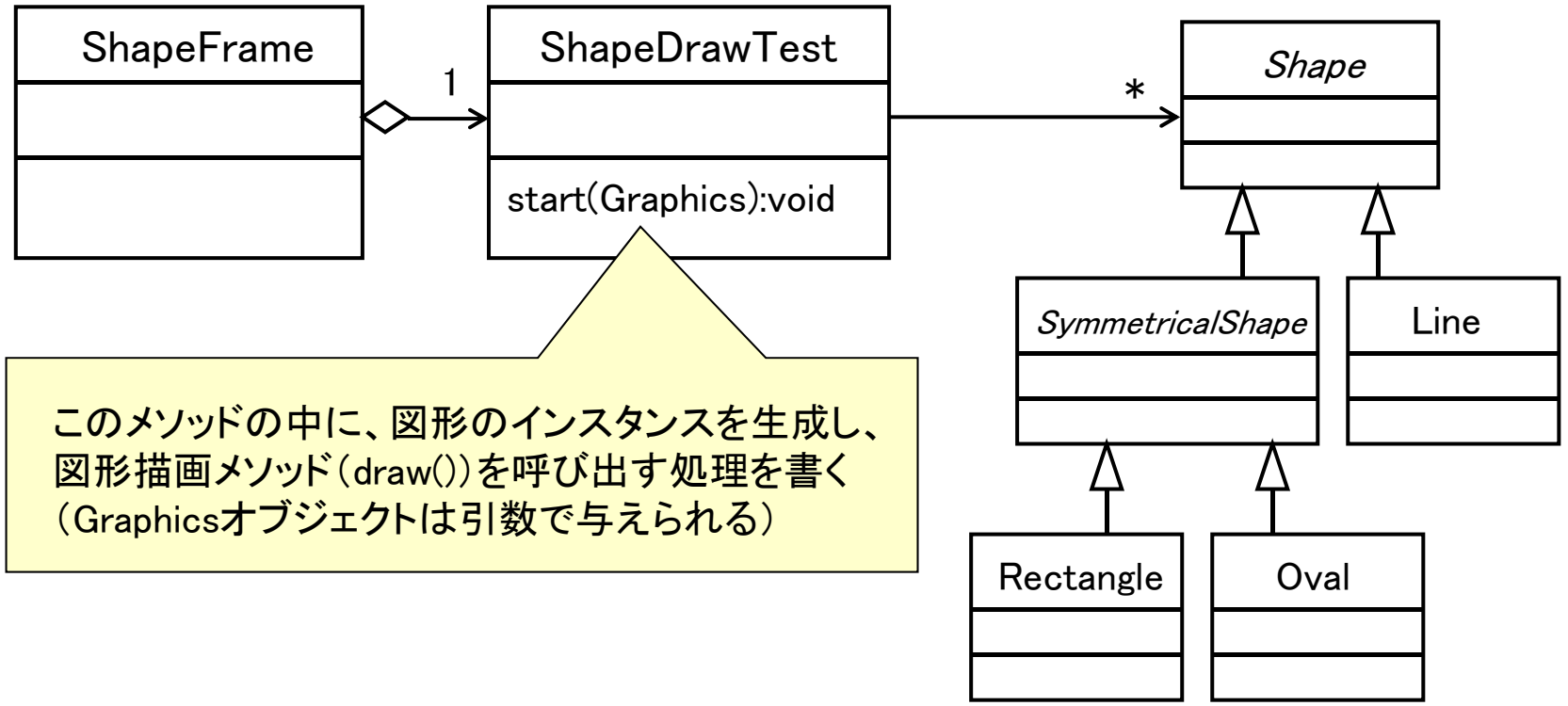
演習問題(5)


```
// Line.java

import java.awt.Graphics;  // クラスGraphicsを利用する
import java.awt.Color;     // クラスColorを利用する

// 直線(線分)を表すクラス
public class Line extends Shape {
  int xend = 0;   // 終点のX座標
  int yend = 0;   // 終点のY座標

  // コンストラクタ
  // 引数:始点のX座標、始点のY座標、終点のX座標、終点のY座標、色
  public Line(int x1, int y1, int x2, int y2, Color color) {
    super(x1, y1, color);
    xend = x2;
    yend = y2;
  }

  public int getXend() {
    return xend;
  }
  public int getYend() {
    return yend;
  }

  // 直線を描画するメソッド
  public void draw(Graphics gra) {
    gra.setColor(color);               // 描画色を設定
    gra.drawLine(x, y, xend, yend);   // 直線(線分)を描画
  }

}
```

演習問題(6)

さらに、図形を表示するためのテストプログラムのクラス ShapeDrawTest を作成する（“ShapeDrawTest.java”のテンプレートを与えるので、いろいろな図形のインスタンスを生成し、描画するコードを追加する）。

ウィンドウ表示のクラスは難しいので、ShapeFrame というクラスを与える。



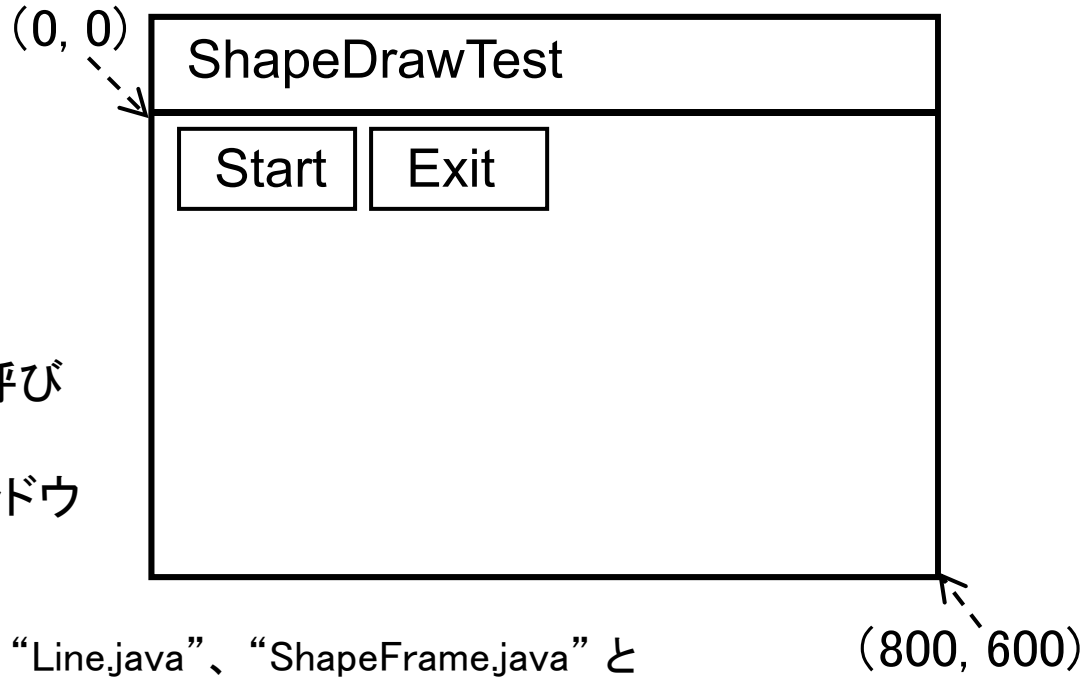
演習問題(7)

```
// ShapeDrawTest.java

import java.awt.Graphics;   // クラスGraphicsを利用する
import java.awt.Color;      // クラスColorを利用する

public class ShapeDrawTest {            // 図形描画テスト用プログラム

  public void start(Graphics gra) {   // ボタン"Start"を押すとこのメソッドを実行

    Line l = new Line(100, 100, 160, 130, Color.red);
                  // 線分(始点(100,100),終点(160,130),色は赤)のインスタンスの生成
    l.draw(gra);    // 線分描画

    // ここに矩形(塗りつぶしなし、塗りつぶしあり)や
    // 楕円(塗りつぶしなし、塗りつぶしあり)など、
    // いろいろな図形のインスタンスを生成して、
    // それを描画するコードを書く

  }
}
```

演習問題（8）

プログラムの起動法は下記。

```
>java ShapeFrame
```



右のようなウィンドウが現れる。
ボタン“Start”をクリックすると、ShapeDrawTest のメソッドstart() が呼び出される。
ボタン“Exit”をクリックすると、ウィンドウが消えて終了。

まずは、ダウンロードした“Shape.java”、“Line.java”、“ShapeFrame.java” と“ShapeDrawTest.java”のテンプレート（添付ファイルをダウンロードした場合、ディレクトリtemplateにある）をコンパイルして実行してみよう。赤い線分が一本表示される。

“SymmetricalShape.java”、“Rectangle.java”、“Oval.java”ができたら、“ShapeDrawTest.java”の例（添付ファイルをダウンロードした場合、ディレクトリtemplateにある）を実行してみよう。何か図が表示される。

“SymmetricalShape.java”、“Rectangle.java”、“Oval.java”、“ShapeDrawTest.java” を以下により圧縮し、ひとつのファイルとして提出してください。

```
zip src.zip SymmetricalShape.java Rectangle.java Oval.java ShapeDrawTest.java
```