情報実験I
プログラムの仕様とテスト

第3回
5月10日

本日の予定

- テストの実施
- 仕様変更要求の説明
- Applet、AWT入門
  - 出力の変更　長方形の描画

テストの実施

- 2人で1組になる。
- 自分のテストケースで、他方のプログラムをテストする。

- 報告
  - 自分のテストケースの数
  - テスト実施者の学籍番号
  - テスト実施者の氏名
  - テストしてもらったすべてのテストケース数
  - 上記の内、自分のプログラムが合格した数
  - 人のプログラムを操作した感想を記せ

GUI入門(1)

長方形エディタのGUI化（出力）

- 標準出力にボードと長方形を表示する

⇒ 四角形の画面への描画・画面の書き換え

- アプレットのサイズからボードのサイズを決める
  - ボードからはみ出しているかの判定は大丈夫？
- 長方形データを描画する
  - 描画位置（x座標、y座標、幅、高さ）
  - 色
  - 塗りつぶし
- 描画のタイミング
  - いつ画面を書き換えるか
  - コマンドを選択するループ

```
>java –cp class EIEV2.RectangleEditor
1:create
2:move
3:expand
4:shrink
5:delete
6:deleteAll
7:intersect
8:displayBoard
9:exit
1
create
width = ?
100
height = ?
100
x = ?
20
y = ?
30
c = ?
1:red
2:blue
3:yellow
4:gray
1
1:[w = 100,h = 100,x = 20,y = 30,color = red]
1:create
2:move
3:expand
4:shrink
5:delete
6:deleteAll
7:intersect
8:displayBoard
9:exit
```

ボードサイズ　固定　600×500で起動
コマンド一覧が表示される

createを選択
必要な入力を要求される

長方形を生成し、ボードに登録する
ボードにある長方形データが表示される

Version 2

Version 3

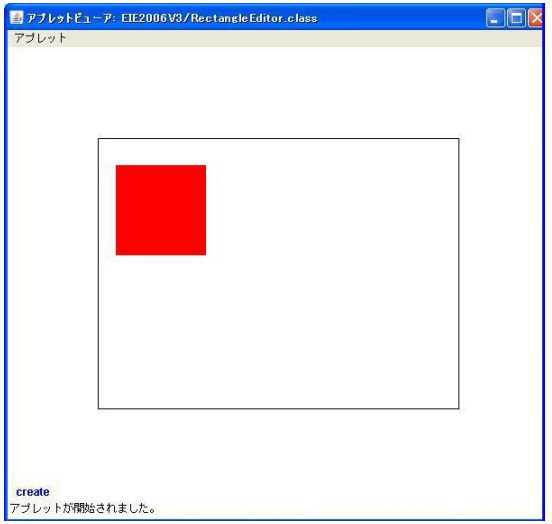
```
>appletviewer class¥Test.html
1:create
2:move
3:expand
4:shrink
5:delete
6:deleteAll
7:intersect
1
create
width = ?
100
height = ?
100
x = ?
20
y = ?
30
c = ?
1:red
2:blue
3:yellow
4:gray
1
1:[w = 100,h = 100,x = 20.0,y = 30.0,color = red]
1
1:create
2:move
3:expand
4:shrink
5:delete
6:deleteAll
7:intersect
```

起動

Test.html

```
<HTML>
<HEAD>
<TITLE>
Rectangle Editor
</TITLE>
</HEAD>
<BODY>
<P>Rectangle Editor Applet:Version 3.
<BR>
<APPLET CODE = “EIEV3/RectangleEditor.class” WIDTH=600 HEIGHT=500>
</APPLET>
</P>
</BODY>
</HTML>
```

ボードにある長方形データが表示される



Applet

- ネットワークを通じてWebブラウザにダウンロードされ、ブラウザのウィンドウに埋め込まれて実行されるJavaプログラム
- 安全のための制約（できないこと）
  - ユーザのハードディスクの内容の読み書き
  - 自分が呼び出されたWebサーバ以外のコンピュータへの接続
  - 他のアプリケーションソフトの起動
- 効果
  - HTMLで記述された静的なWebページでは実現できない動的な表現が可能になる

Javaアプリケーションの基本構造

```
import …
class  クラス名{
            フィールドの宣言
            public static void main(String[] args){



            }
}
```

main()の最初から順番に処理される
main()の終了まで

```
class  クラス名{
            フィールドの宣言
            メソッドの宣言
}
```

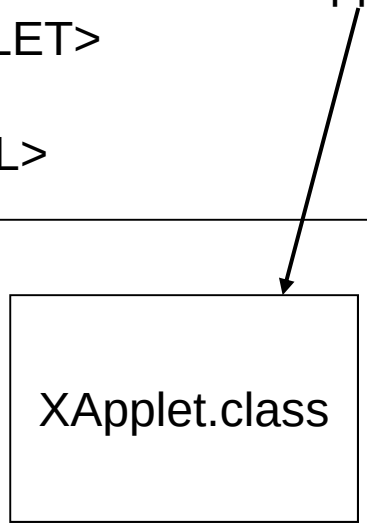
```
class  クラス名{
            フィールドの宣言
            メソッドの宣言
}
```

アプレットの基本構造

```
<HTML>

<APPLET CODE = XApplet.class …>
</APPLET>

</HTML>
```




```
import    java.applet.Applet;
class  XApplet extends Applet{
            フィールドの宣言
            public void init(){
            }
            public void start(){
            }
            public void stop(){
            }
            public void paint(){
            }
}
```

```
class  クラス名{
            フィールドの宣言
            メソッドの宣言
}
```

定義していないメソッドは
スーパークラスのメソッドが使われる

アプレットの実行手順

- コンパイルする

⇒　A.class　等のクラスファイルが生成される

- HTMLファイルを定義する

```
<HTML>
<HEAD>
<TITLE>
Rectangle Editor</TITLE>
</HEAD>
<BODY>
<P>Rectangle Editor Applet:Version 3.
<BR>
<APPLET CODE = A.class WIDTH=800 HEIGHT=150>
</APPLET>
</P>
</BODY>
</HTML>
```

※AはAppletクラスを継承しているクラス
パッケージの定義があれば
APPLET CODE =
にはパスが通るようにクラスファイル名
を指定すること

- ブラウザで表示する

アプレットの実行

- Java対応のブラウザから見る
- アプレットビューアを使う
  - appletviewer *appletName.html*

Version 3の場合は、キーボード（標準入力）からの入力を行うので、ブラウザから起動するのではなく、アップレットビューアを使って、コマンドラインから起動すること！

Version 3

```
>appletviewer class¥Test.html
1:create
2:move
3:expand
4:shrink
5:delete
6:deleteAll
7:intersect
1
create
width = ?
100
height = ?
100
x = ?
20
y = ?
30
c = ?
1:red
2:blue
3:yellow
4:gray
1
1:[w = 100,h = 100,x = 20.0,y = 30.0,color = red]
1
1:create
2:move
3:expand
4:shrink
5:delete
6:deleteAll
7:intersect
```

起動

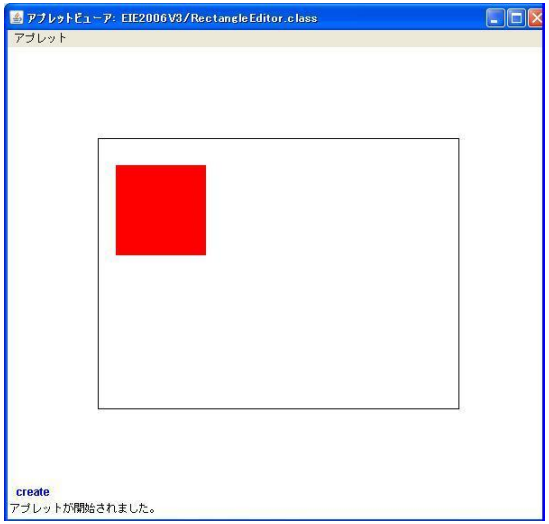
Test.html

```
<HTML>
<HEAD>
<TITLE>
Rectangle Editor
</TITLE>
</HEAD>
<BODY>
<P>Rectangle Editor Applet:Version 3.
<BR>
<APPLET CODE = “EIEV3/RectangleEditor.class” WIDTH=600 HEIGHT=500>
</APPLET>
</P>
</BODY>
</HTML>
```

RectangleEditorクラスを
Appletクラスを拡張して定義する

ボードにある長方形データが表示される

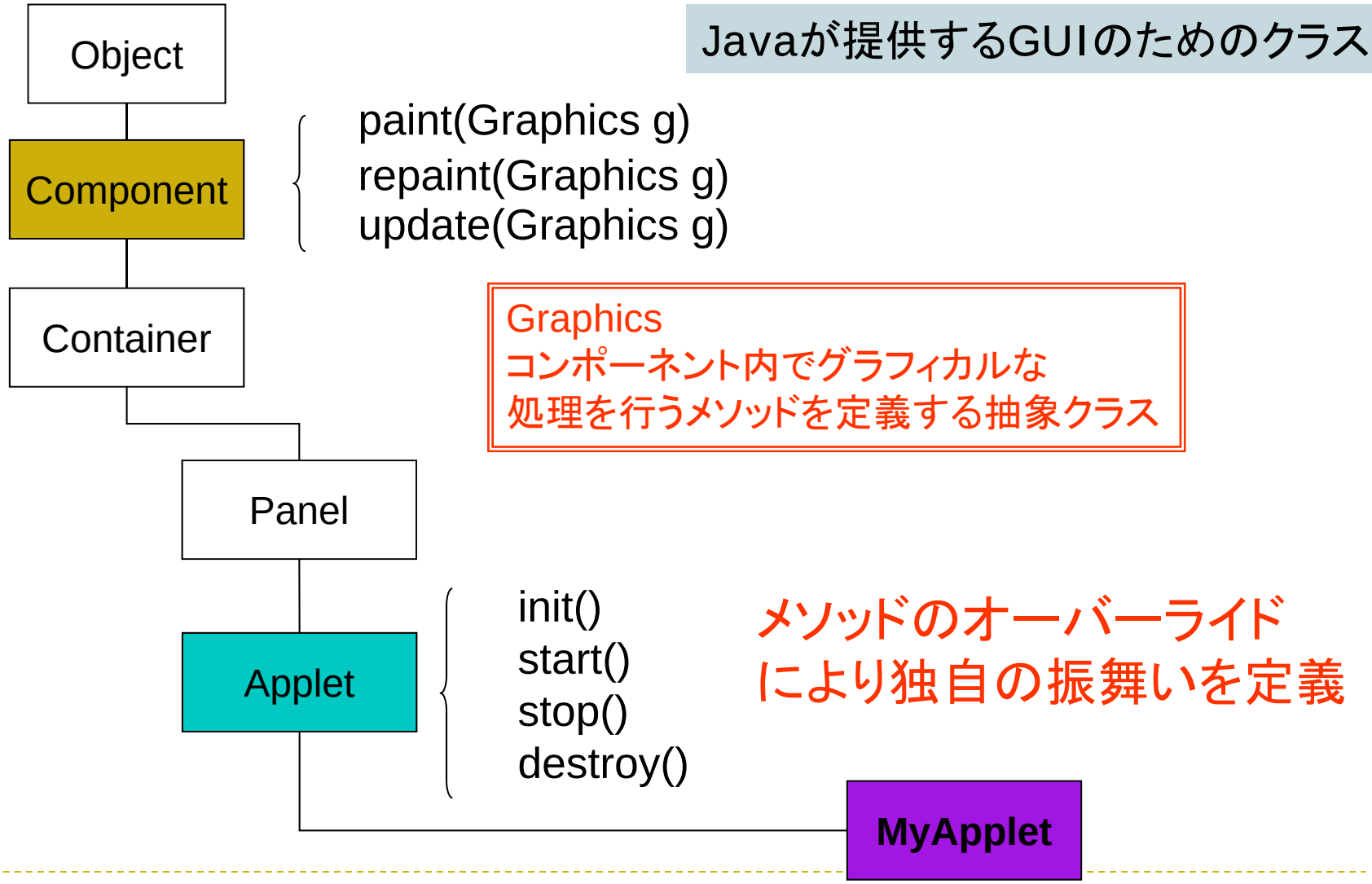
このデータを
Applet上に描画する



何を変更すればよいのか？

- 起動の仕組みを変える
  - Appletクラスを継承してクラスを定義する
- ボード上の長方形データを描画する
  - アプレット上に長方形データに基づき、長方形を描画する
    - 座標を計算して、適切な位置に表示することが必要

アプレットの継承の関係とメソッド

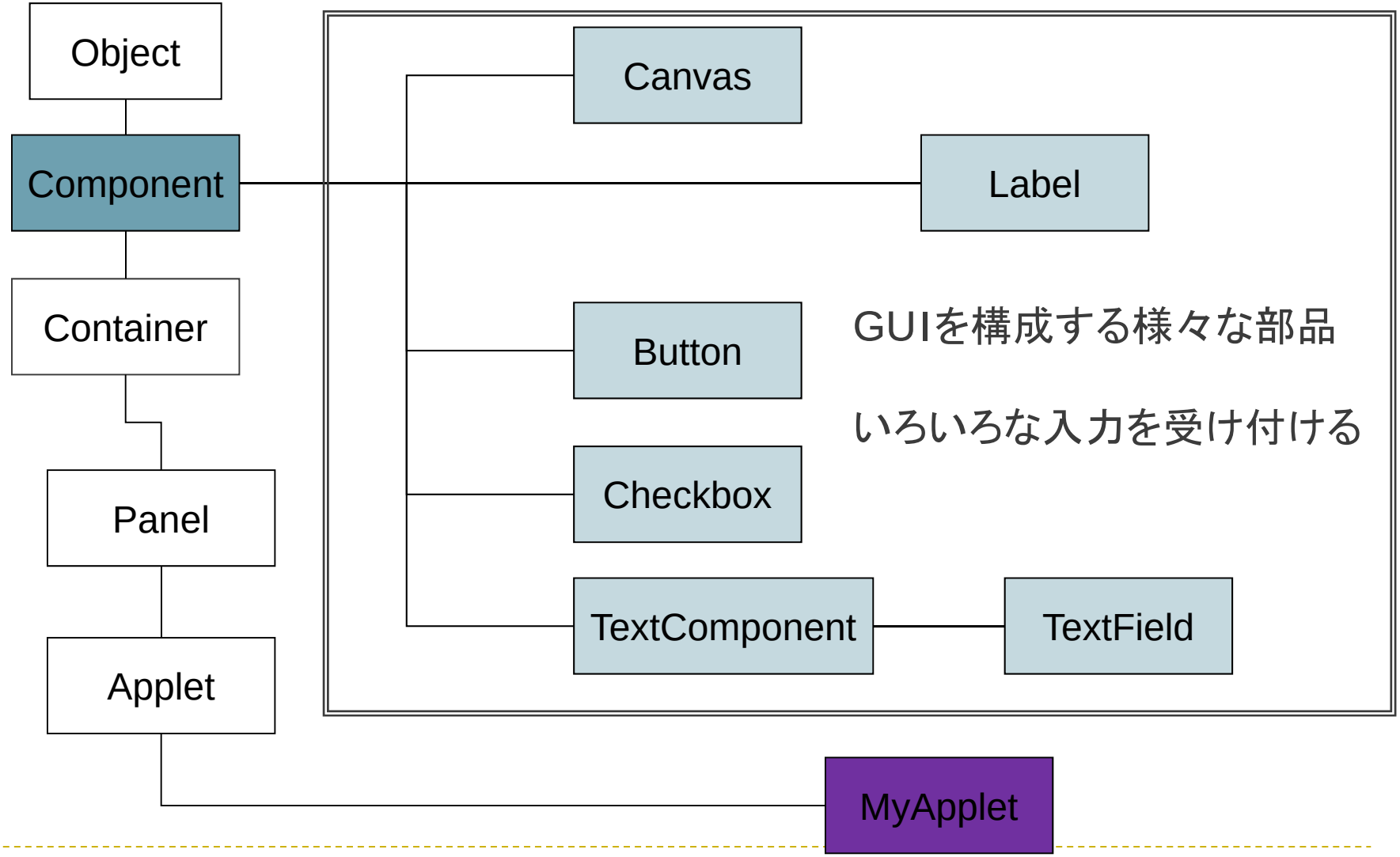


Javaが提供するGUIのためのクラス

Graphics
コンポーネント内でグラフィカルな
処理を行うメソッドを定義する抽象クラス

メソッドのオーバーライド
により独自の振舞いを定義

Applet上で利用できる様々な部品（Component）



Applet　クラス

- アプレットの動作を定義するメソッドの基本セットを提供
- アプレットの実行を制御するメソッド
  - ユーザが呼び出すのではない
  - ブラウザが自動的に呼び出す
  - init()
  - start()
  - stop()
  - destroy()

アプレットの実行を制御するメソッド

- 初期化
  public void init(){
  }
  - アプレットがロードされるときに最初に呼び出される
- 開始
  public void start(){
  }
  - アプレットが初期化された後、続いて呼び出される
  - アプレットが一旦停止した後で再開する場合にも呼び出される
- 終了
  public void stop(){
  }
  - アプレットを含むドキュメントの表示が終了するときに呼び出される

  public void destroy(){
  }
  - stop()メソッドの後に呼び出され、リソースを開放する

アプレットの表示更新メソッド

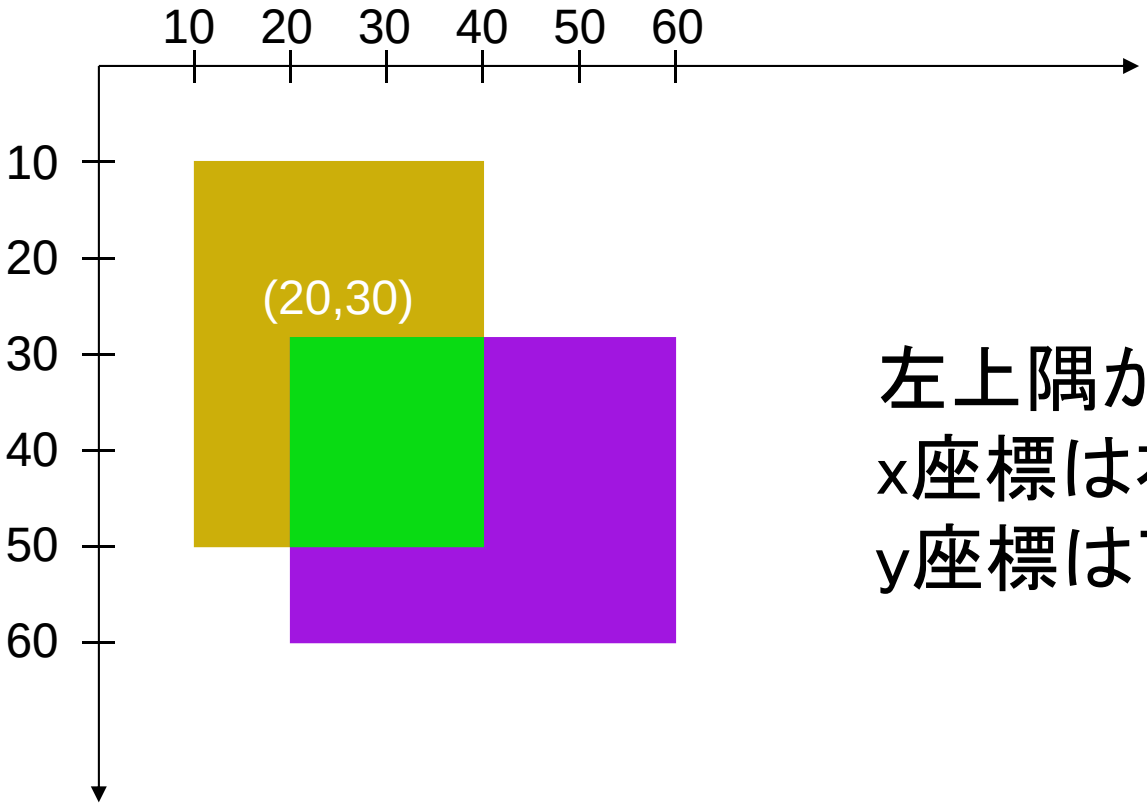
- java.awt.Componentクラスのメソッド
  - paint()
  - repaint()
  - update()
- オーバーライドして使う
- 描画

```
public void paint(){
}
```

  - アプレットの開始時に1度呼び出される
  - repaint() ⇒ update() ⇒ paint()
  - repaint()
    - update() メソッドを呼び出す
  - update()
    - デフォルトは背景色で画面をクリアした後paint()を呼び出す

描画する

Graphicsの座標系　単位はピクセル



左上隅が(0,0)
x座標は右方向に増加
y座標は下方向に増加

Graphicsクラス　java.awt パッケージ

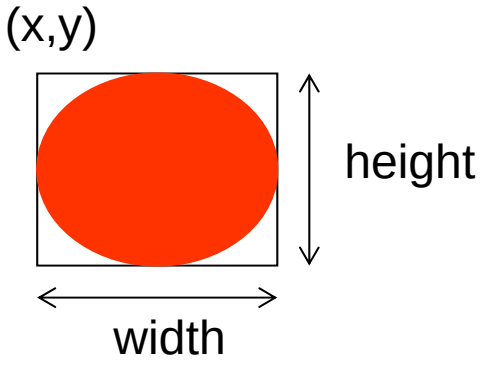
メソッド

- テキストを描く
  - drawString(String str, int x, int y)
- 矩形を描く
  - drawRect(int x, int y, int width, int height)
  - drawRoundRect(int x, int y, int width, int height)
- 線を描く
  - drawLine(int x1, int y1, int x2, int y2)
- 楕円を描く
  - drawOval(int x, int y, int width, int height)
- 塗りつぶす
  - fillRect (int x, int y, int width, int height)
  - fillOval (int x, int y, int width, int height)

(x,y) String

ベースライン

座標、幅、高さの単位はピクセル



長方形を描く

- 長方形は　幅、高さ、左上隅の座標、色の属性を持っている
- 赤色で塗りつぶした矩形を描くには？
  - Graphicsクラスのメソッド
  - fillRect (int x, int y, int width, int height)を使用

⇒　このメソッドは

  - 幅、高さ、左上隅の座標を指定してGraphicsオブジェクトに矩形を塗りつぶすことを命令しているが、色の指定は？

Fontクラス・Colorクラス　java.awtパッケージ

- Graphicsオブジェクトに描画するとき、カレントの色やフォントが設定されている
- 描画対象の色やテキストのフォントを指定してから描画する
- Fontオブジェクトの生成
  - Font font = new Font("TimesRoman",Font.BOLD,36);
- Colorオブジェクトの生成
  - 標準の色　Color.white Color.black Color.red etc.
  - Color color = new Color(140,140,140);
- フォントを設定する
  - setFont(Font font)
- 色を設定する
  - setColor(Color color)

```
package GUISample;
import java.applet.Applet;
import java.awt.Graphics;
import java.awt.Color;

public class drawSample extends Applet {
        public void init() {
                setBackground(Color.white);
        }
        public void paint(Graphics g) {
                g.setColor(Color.blue);
                for(int i = 1; i < 10; i++){
                        g.drawLine(50,  50*i, 450, 50*i);
                }
                g.setColor(Color.green);
                for(int i = 1; i < 9; i++){
                        if (i%2==0){
                                g.drawLine(100,  50*(i+1), 400, 50*i);
                        } else{
                                g.drawLine(100,  50*i, 400, 50*(i+1));
                        }
                }
                g.setColor(Color.red);
                g.fillOval(50+(30/2), (50-30), 30, 30);
                g.fillOval(50+(30/2), (50-30)+8*50, 30, 30);
        }
}
```

使用するクラスをインポートする

背景色を設定する

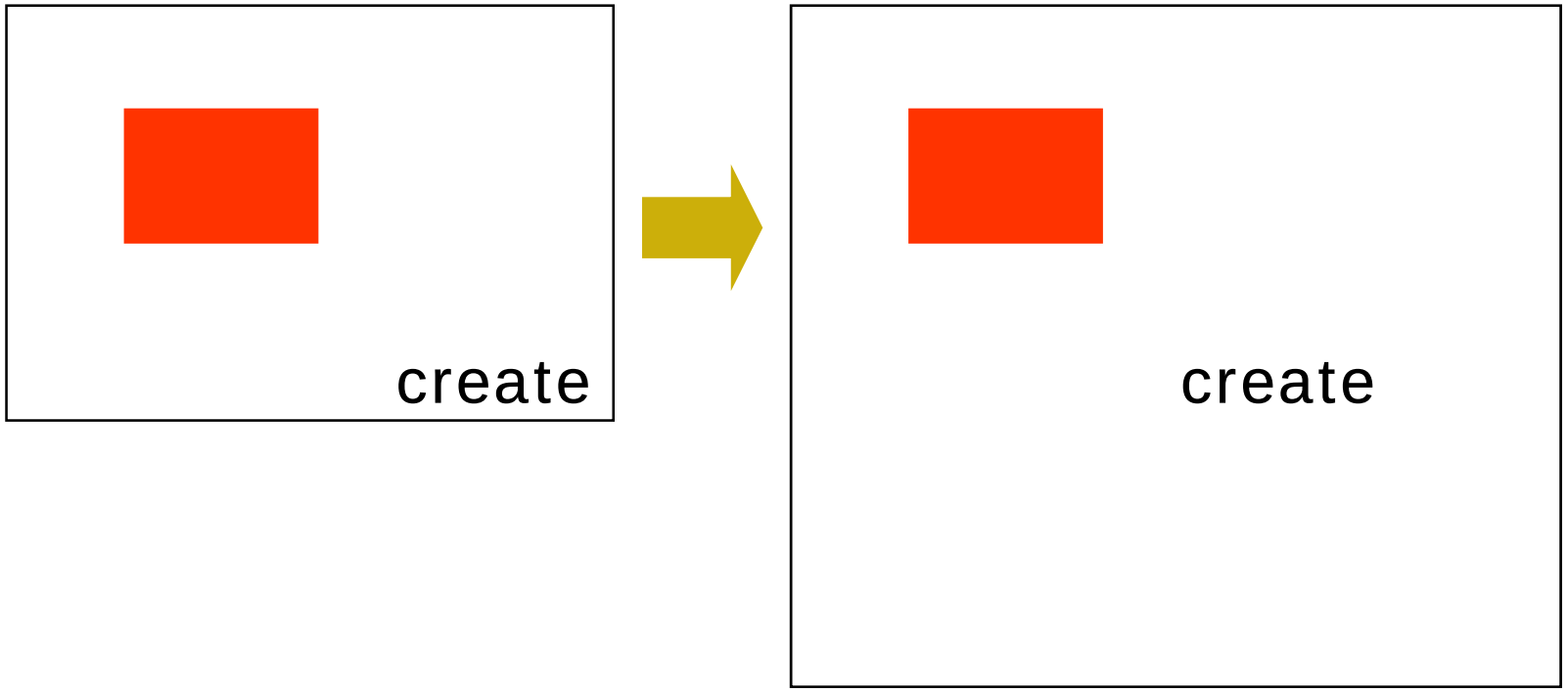
使用する色を設定する

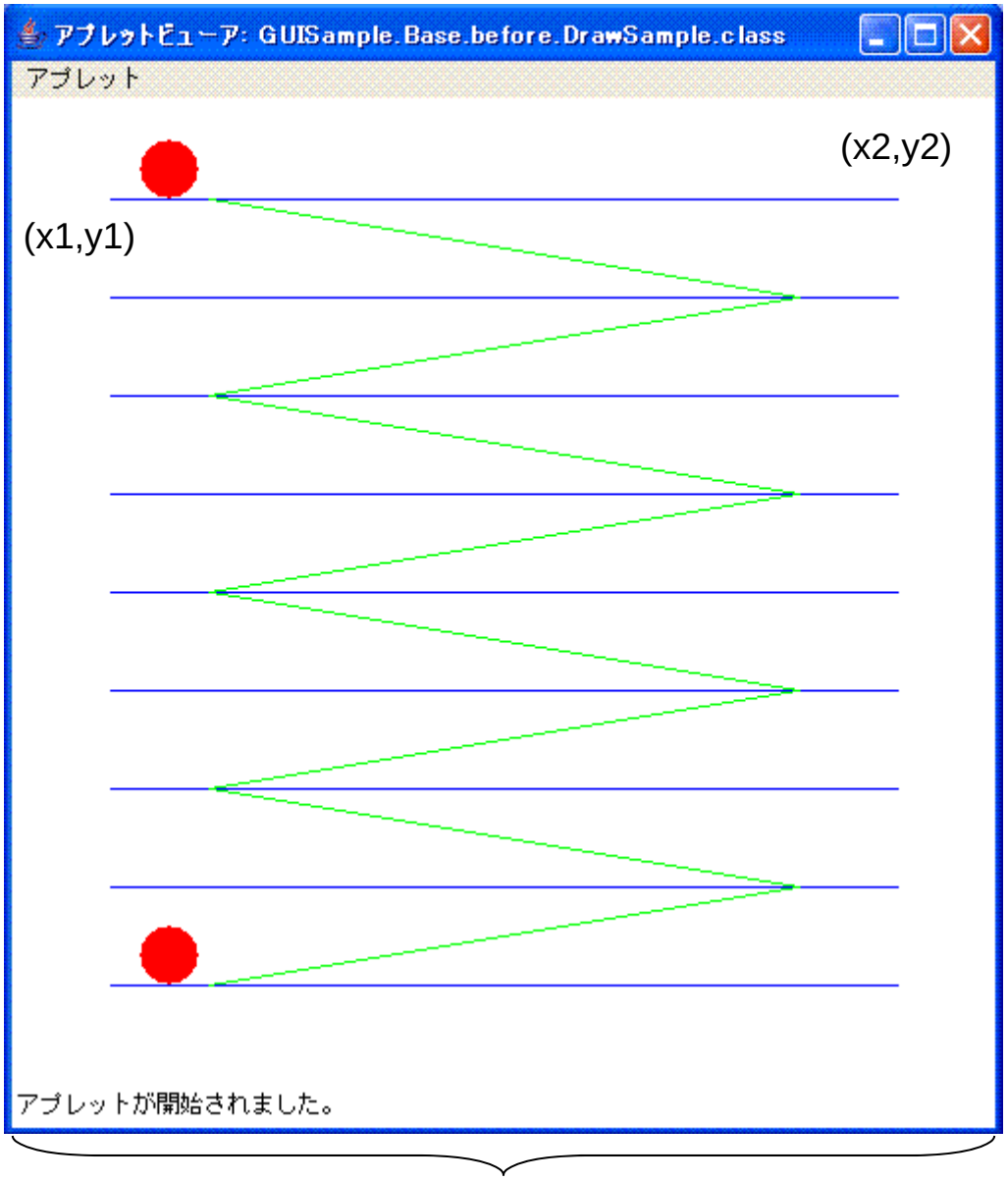
線分を描く

円を塗りつぶす

アプレットのサイズ

- ボードの大きさはアプレットのサイズで決まるとする
- 描画の位置がアプレットのサイズに依存しないようにするには？





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

高さ

幅

アプレットの大きさに合わせて
横には一定のスペースを設け
一定の間隔の線分を引く

円は斜め線の開始位置から見て
丁度よさそうなところに置く

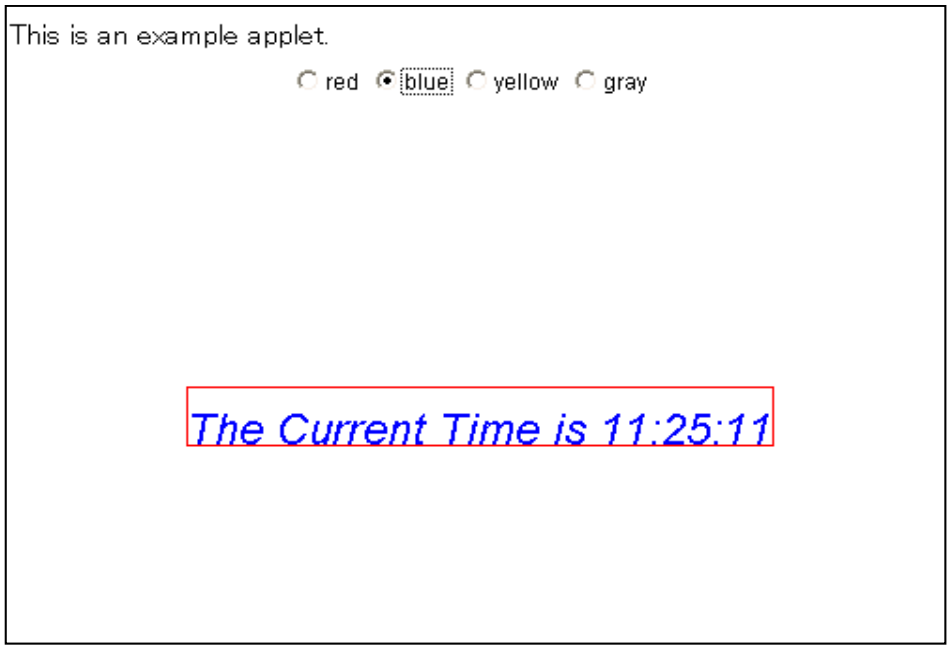
アプレットの大きさを変えても、
このように描けるようにするには？

入力と出力

- 画面からの入力によって出力を変化させる
  - 入力：ラジオボタン（チェックボックス）
  - 入力結果を反映して出力する

  ＝入力によりデータを変化させ、表示を変更する

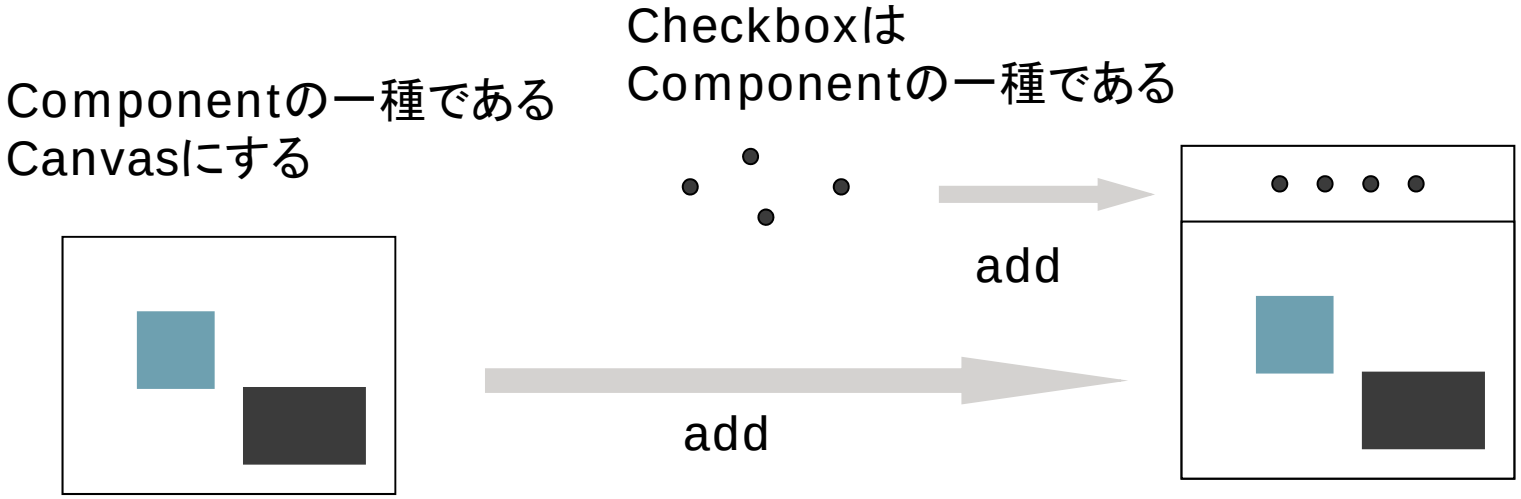
- 文字の色をラジオボタンを使用して指定する。



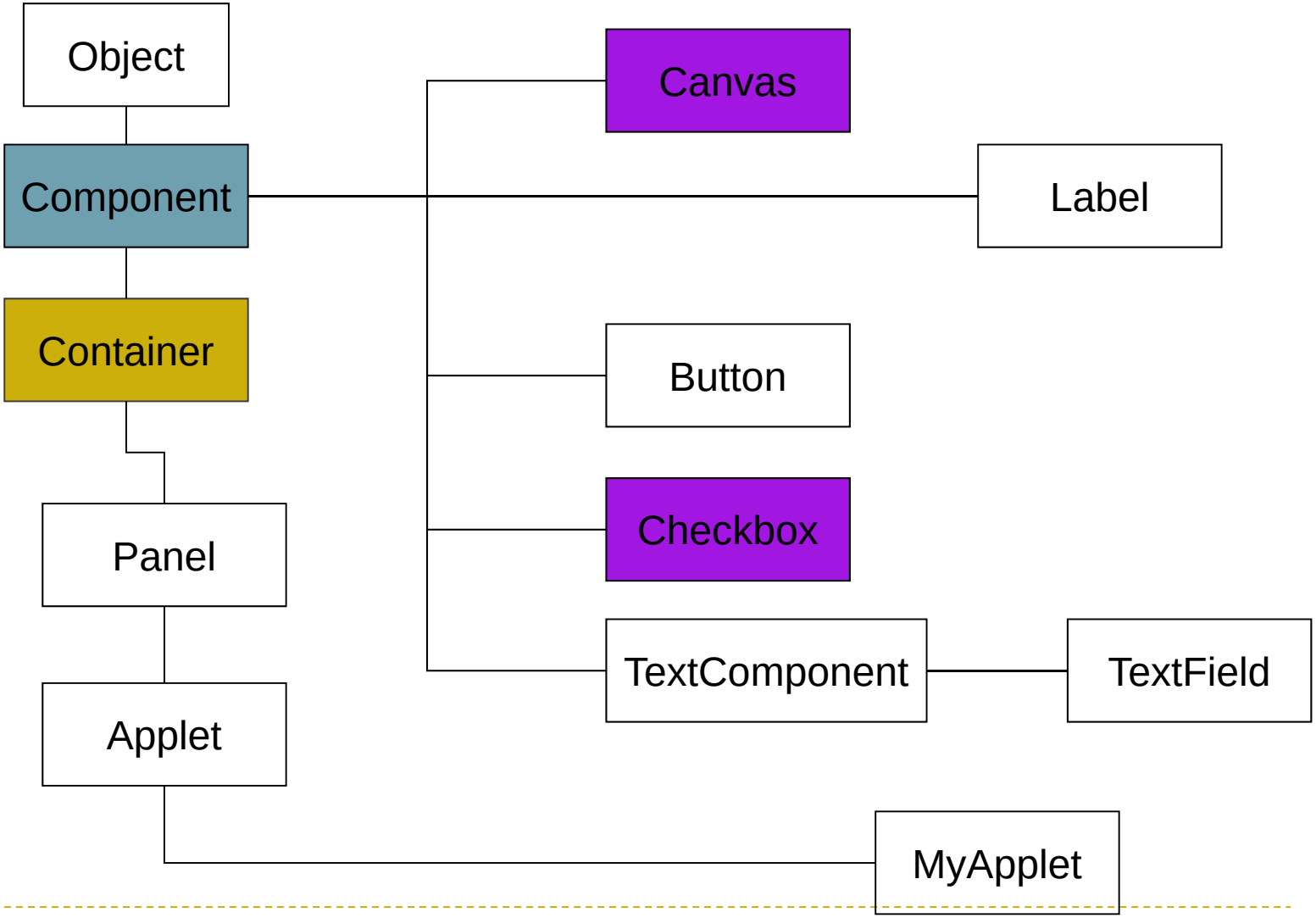
アプレット上に表示するもの

- 一組になった複数のボタン・出力イメージ

- １つの入れ物に複数の部品（Component）を追加する
- add: Container Componentを追加する
- ContainerはComponentの一種であり、Canvas,Button,Checkbox等の集合体



使用するクラスの関係


Object
Component
Container
Panel
Applet
MyApplet
Canvas
Label
Button
Checkbox
TextComponent
TextField