オブジェクト指向プログラミング(2)

第4回

横山　孝典
E-mail: tyoko@tcu.ac.jp

Java言語に関する補足事項

- パッケージ
- 型の互換性
- instanceof 演算子

パッケージ(1)

パッケージ

- 複数のクラスを、パッケージにまとめることができる。
- クラスをパッケージに属させるには、クラスのソースファイルの先頭で、以下を宣言する。

  package パッケージ名;

  例:

```
package jp.ac.tcu.cs.shapes;

public class Line extends Shape {
   ....
}
```

  注）パッケージ名の宣言をしないと、名前なしパッケージとなる。

- パッケージに含まれるクラスをパッケージ外から参照する場合には、パッケージ名とクラス名の両者を指定する必要がある。
- ただし、import文を使用することで、クラス名のみで参照できるようになる。

  例:

```
import java.awt.Graphics;
```

パッケージjava.awt中のクラスGraphicsをクラス名のみで参照できるようにする

```
import java.awt.*;
```

パッケージjava.awt中の全てのクラスをクラス名のみで参照できるようにする

パッケージ（2）

参考）Java API のパッケージの例

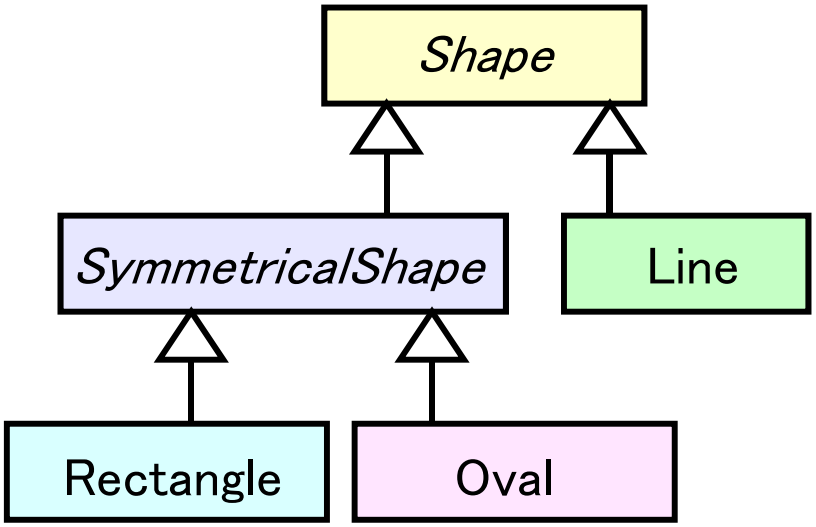
- java.awt（abstarct window toolkit）パッケージ
  - GUI関連のクラスをまとめたパッケージ
  - クラスの例

    java.awt.Graphics

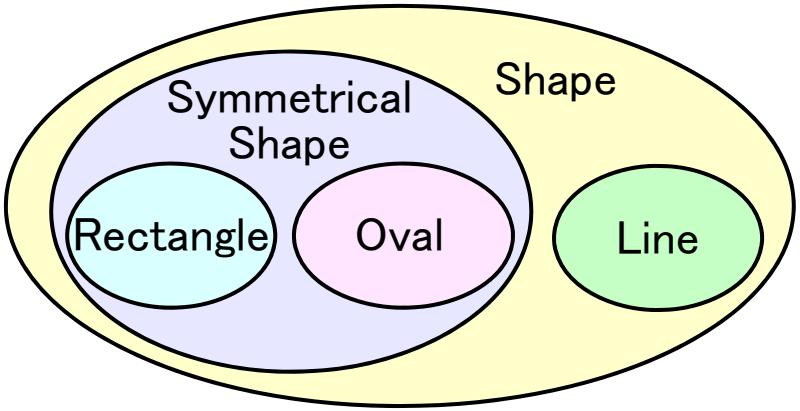
    メソッドの例（実装はサブクラスのものあり）
    - void setColor(Color c) ：そのグラフィックスコンテキストの色の設定
    - Color getColor(Color c) ：そのグラフィックスコンテキストの色の読み出し
    - void drawLine(int x1, int y1, int x2, int y2) ：点(x1,y1)から点(x2,y2)まで直線を描画
    - void drawRect(int x, int y, int width, int height) ：位置(x,y)に幅width高さheightの矩形を描画
    - void fillRect(int x, int y, int width, int height) ：位置(x,y)に幅width高さheightの矩形塗りつぶし
    - void drawOval(int x, int y, int width, int height) ：位置(x,y)に幅width高さheightの楕円を描画
    - void fillOval(int x, int y, int width, int height) ：位置(x,y)に幅width高さheightの楕円塗りつぶし
- java.awt.eventパッケージ
  - java.awt パッケージのサブパッケージで、イベント処理に関するクラスをまとめたパッケージ
- javax.swingパッケージ
  - GUIの機能を提供するSwingコンポーネントに関連するクラスに関するパッケージ

型の互換性

クラスの継承関係



型の包含関係



広い型への変換はそのまま可

```
Rectangle re = new Rectangle();
SymmetricalShape sy = re;
Shape sh = sy;
```

狭い型への変換は明示的な
キャストが必要

```
Shape sh = new Rectangle();
SymmetricalShape sy =
        (SymmetricalShape)sh;
Rectangle re = (Rectangle)sy;
```

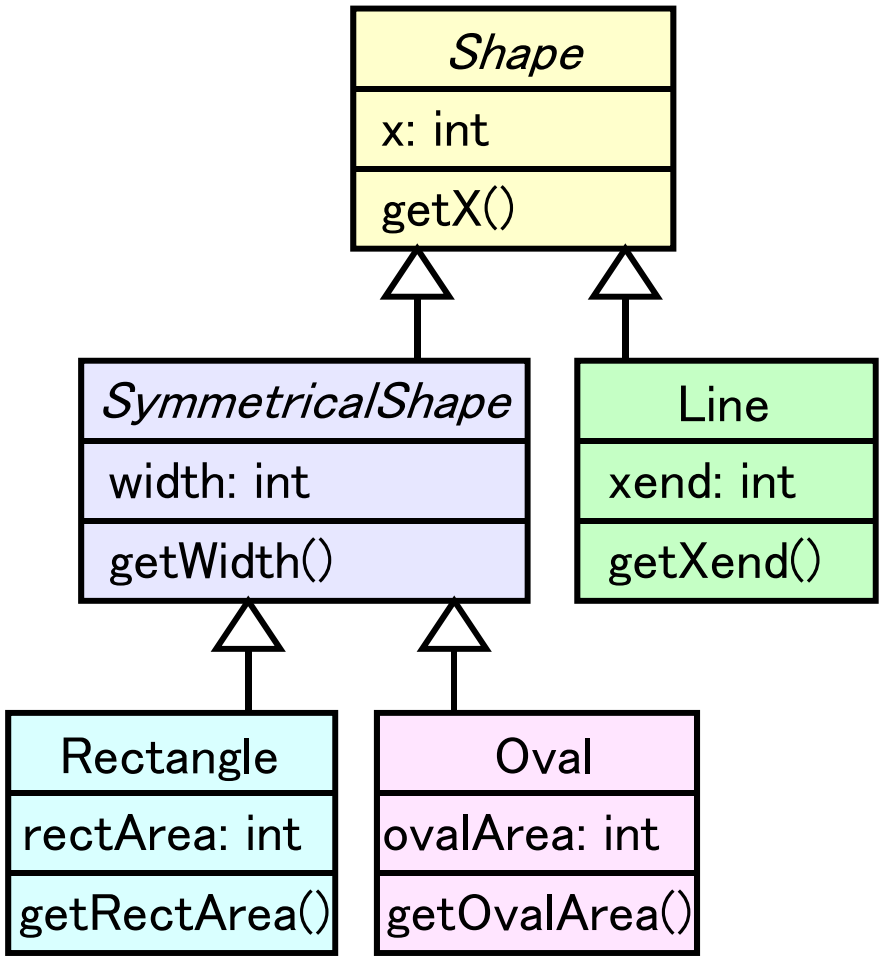
互換性のない型は明示的な
キャストをしても不可

```
Rectangle re = new Rectangle();
Oval ov = (Oval)re; // 不可

SymmetricalShape sy = new Oval();
Rectangle re = (Rectangle)sy; // 不可
```

型とメソッド呼び出し

クラスの継承関係



型に合ったメソッド呼び出しのみ可

```
Rectangle re = new Rectangle();
int x = re.getX();
int w = re.getWidth();
int r = re.getRectArea();
```

型に合わないメソッド呼び出しは不可

```
Shape sh = new Rectangle();
int x = sh.getX();
int w = sh.getWidth(); // 不可
int r = sh.getRectArea(); // 不可
```

注）互換性がある場合は、型を合わせて呼び出せば可

```
Shape sh = new Rectangle();
Rectangle re = (Rectangle)sh;
int w = re.getWidth();
int r = re.getRectArea();
w = ((SymmetricalShape)sh).getWidth();
r = ((Rectangle)sh).getRectArea();
```

instanceof 演算子

instanceof 演算子

- 対象インスタンスが、特定のクラスまたはそのサブクラスのインスタンスかどうか、あるいは、特定のインタフェースを継承したクラスのインスタンスかどうかを判定する演算子
- 結果はboolean（true か false）
- 使い方

インスタンスの参照 instanceof クラス名またはインタフェース名

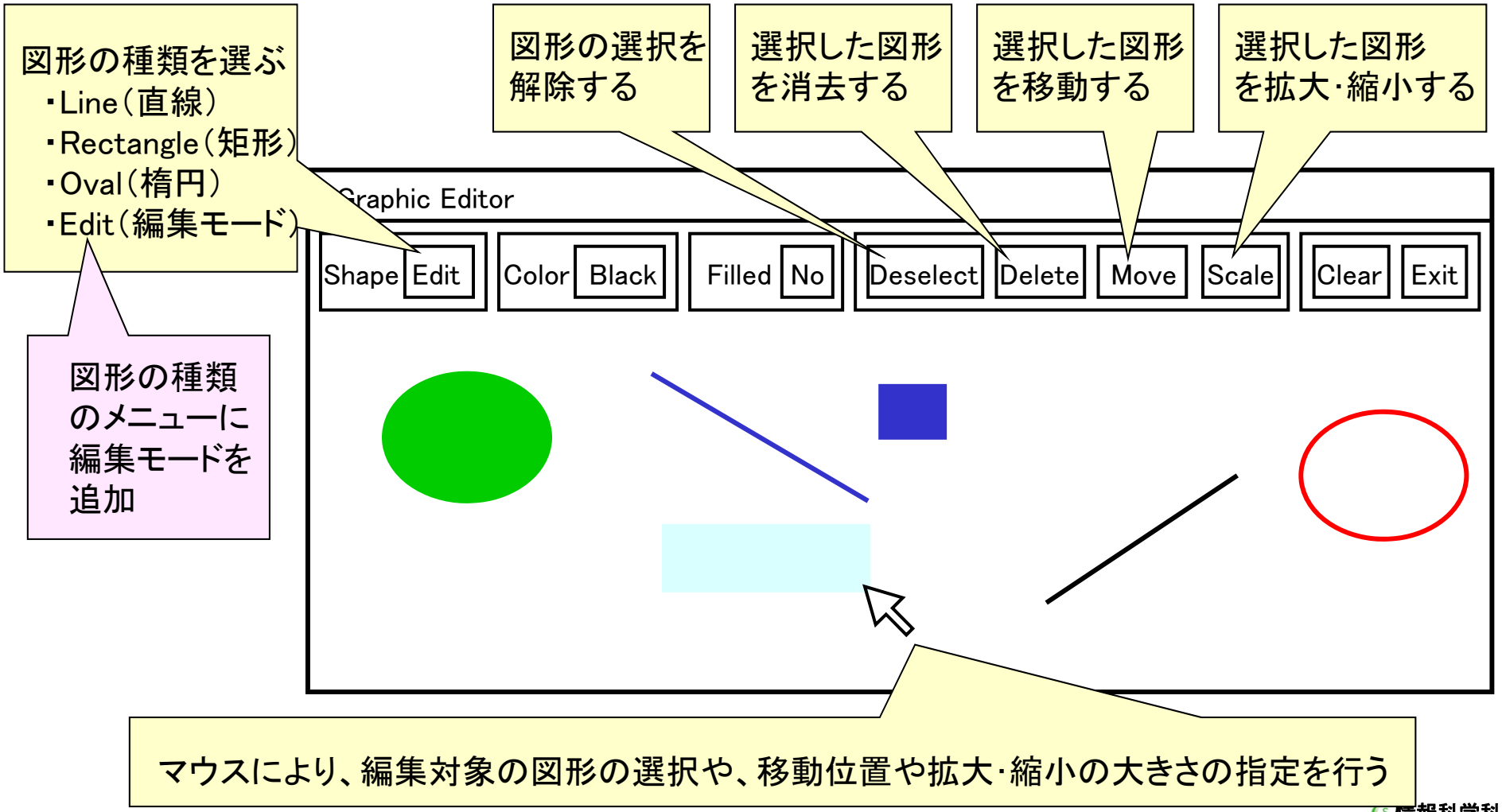
例：

```
Shape sh;
・・・・・
sh = new Rectangle( ・・・・ );
・・・・・
if ( sh instanceof Rectangle ) {
  Rectangle re = (Rectangle)sh;
  ・・・・・
} else if ( sh instanceof Oval ) {
  Oval ov = (Oval)sh;
  ・・・・・
}
```

演習問題(1)

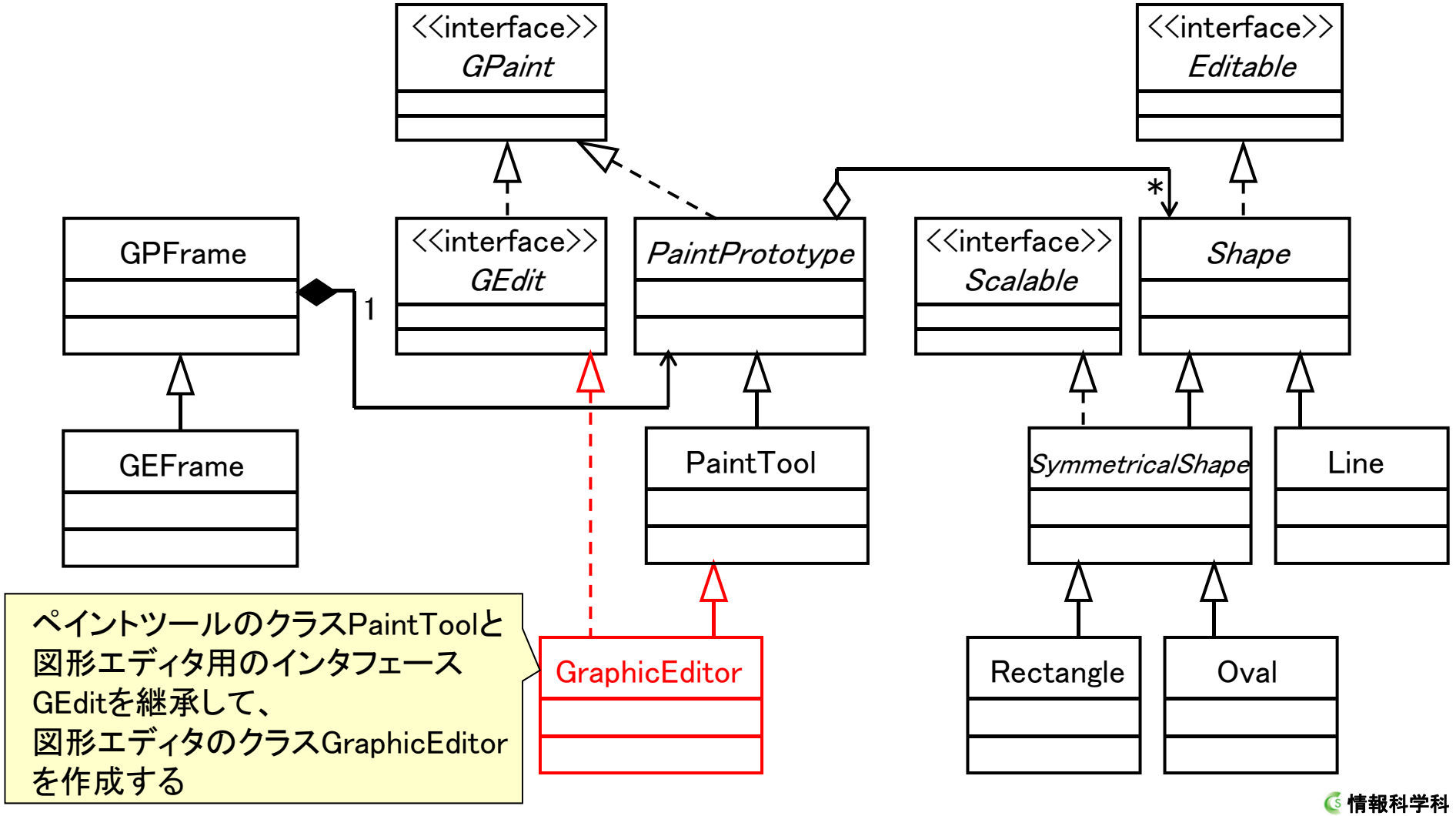
描画機能のみでなく、編集機能も持つ図形エディタを作成する。

図形を選択し、消去、移動、拡大・縮小が可能な、図形エディタを作成する。



演習問題(2)

黒線・黒字で記したクラスを与える。
赤線・赤字で記したクラス GraphicEditor を新規作成する。



演習問題(3)

図形エディタの仕様

- 図形の描画のみでなく、編集モード(Edit)を選ぶことができる
  - 図形の種類のメニューで、直線(Line)、矩形(Rectangle)、楕円(Oval)、編集モード(Edit)のいずれかを選ぶ
- 編集モード以外の場合の機能は、前々回のペイントツールと同じ
- 編集モードでは、図形に対して、以下の操作を可能とする
  - 図形の選択(選択した図形はハイライト表示する)
  - 選択解除(Deselect)
  - 消去(Delete)
  - 移動(Move)
  - 拡大縮小(Scale)

図形エディタの起動方法

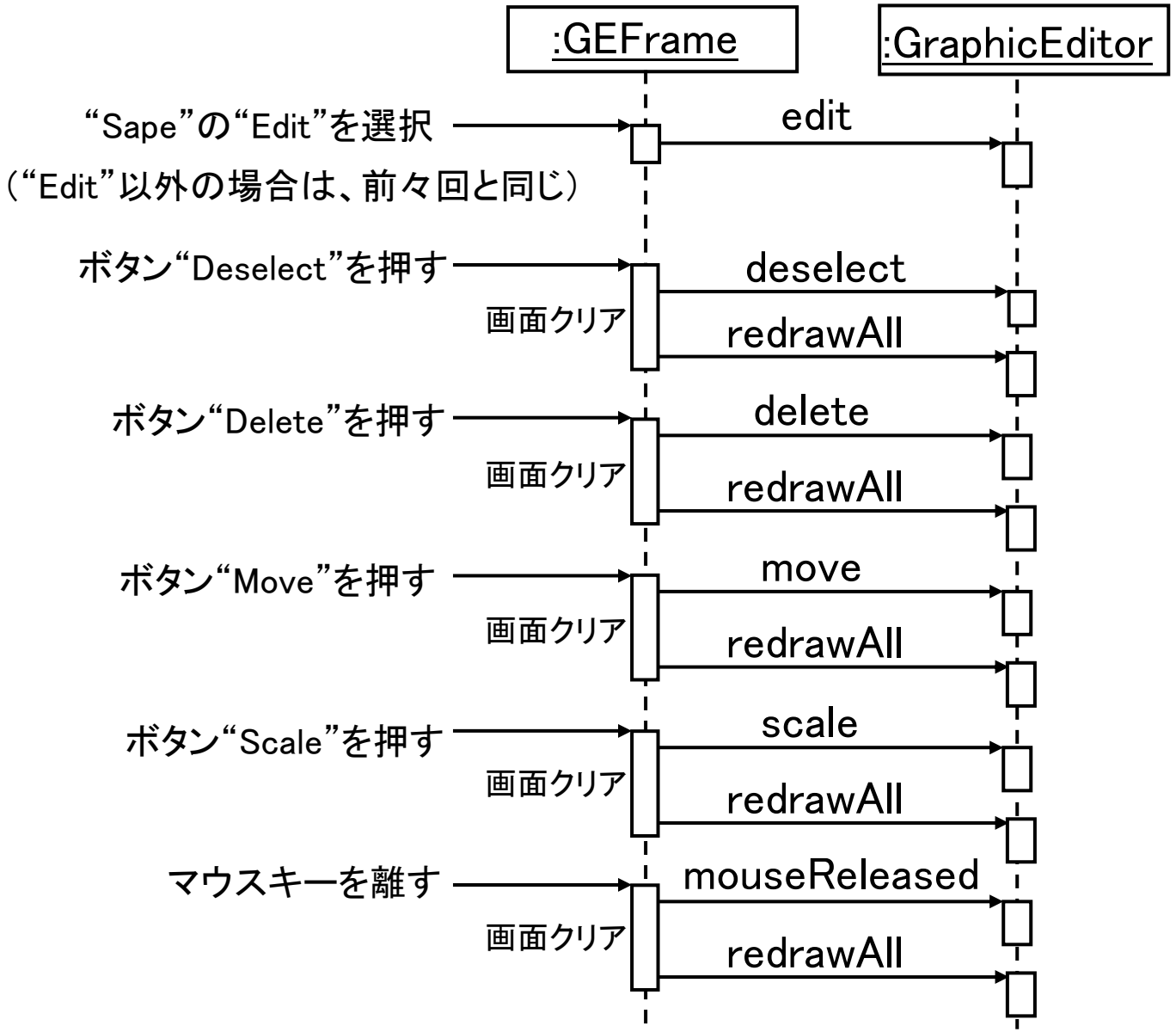
- 以下により起動できる

  Java GEFrame

  - 上記コマンドにより、GEFrameとGraphicEditorのインスタンスを生成し、動作を開始することができる

演習問題(4)

シーケンス図(追加または変更した機能のみ記載)



演習問題(5)

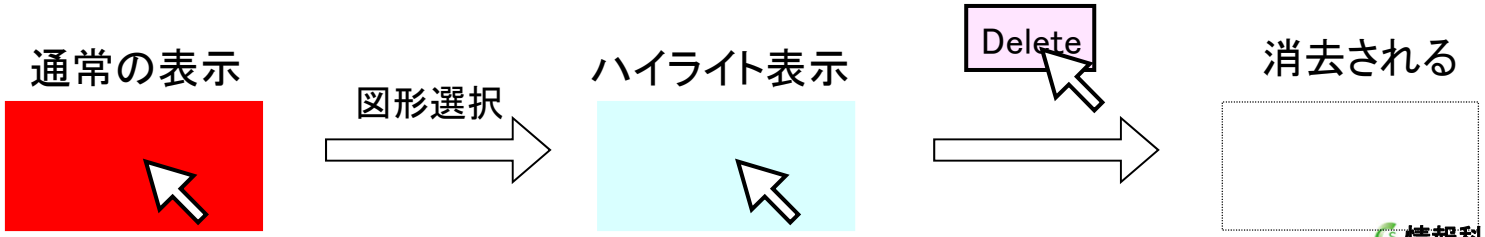
作業手順

- ステップ1
  - クラス GraphicEditorを作成し、編集モードに切り替えられるようにする。
    - まず、インタフェースGeditのソースコードを読んでよく理解する。
    - 前回作成したクラスPaintToolとインタフェースGeditの両者を継承したクラスGraphicEditorを作成する。
    - 編集モードが選ばれたとき(メソッド edit() が呼ばれたとき) 、shapeId(図形の種類)の値は、例えば、-1 とする。
    - この段階では、インタフェースGeditで指定されたメソッドのうち edit() のみ処理を記述し、他は空でよい。
    - 編集モードの以外のとき、継承しているPaintToolの機能を使うように、必要なメソッドの記述を行う。
    - 編集モードの以外のとき、ペイントツールと同じ機能が実現されていることを確認する。

演習問題(6)

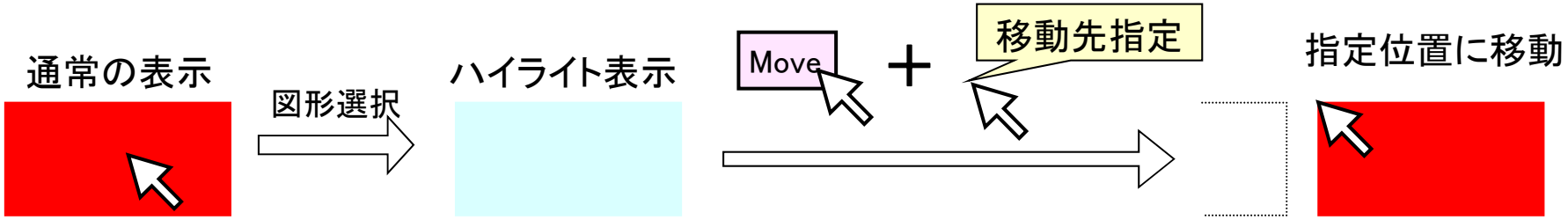
- ステップ2
  - クラスGraphicEditorを修正し、編集モードにおいて、マウスにより、図形を選択する機能を実現する。
    - 選択されると、ハイライト表示されるようにする。
    - 選択状態で“Deselect”ボタンを押すと、選択が解除され、ハイライト表示から本来の表示に戻るようにする。

- ステップ3
  - クラスGraphicEditorを修正し、編集モードにおいて、図形の消去ができるようにする。
    - 図形を選択した状態で、“Delete”ボタンを押すと、その図形が消去されるようにする。

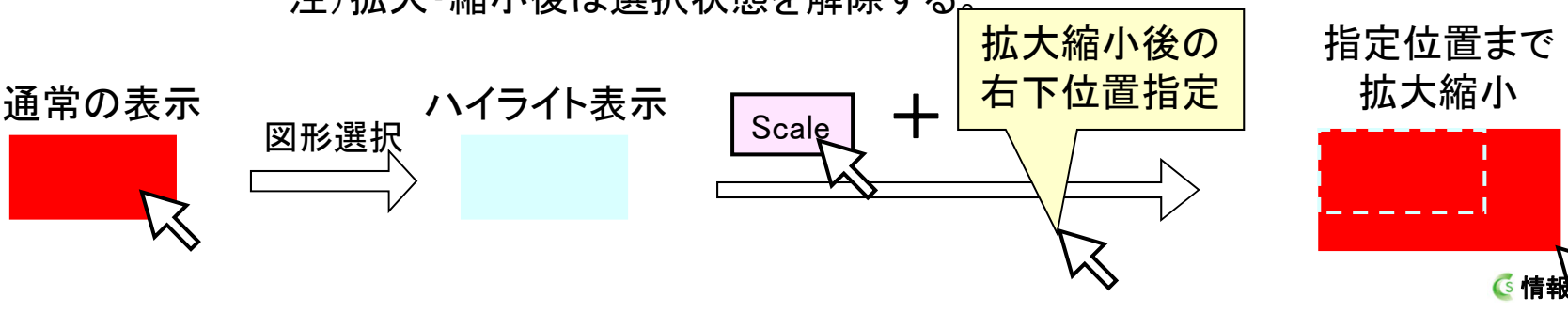


演習問題(7)

- ステップ4
  - 編集モードにおいて、図形の移動ができるようにする。
    - 図形を選択した状態で、“Move”ボタンを押し、さらに、マウスで位置を指定すると、図形が指定位置まで移動する。
      注)移動後は選択状態を解除する。



- ステップ5
  - 編集モードにおいて、図形の拡大・縮小ができるようにする。
    - 拡大・縮小の対象は、矩形と楕円のみとする。
    - 図形を選択した状態で、“Scale”ボタンを押し、さらに、マウスで位置を指定すると、指定位置に合わせて拡大・縮小する。
      注)拡大・縮小後は選択状態を解除する。

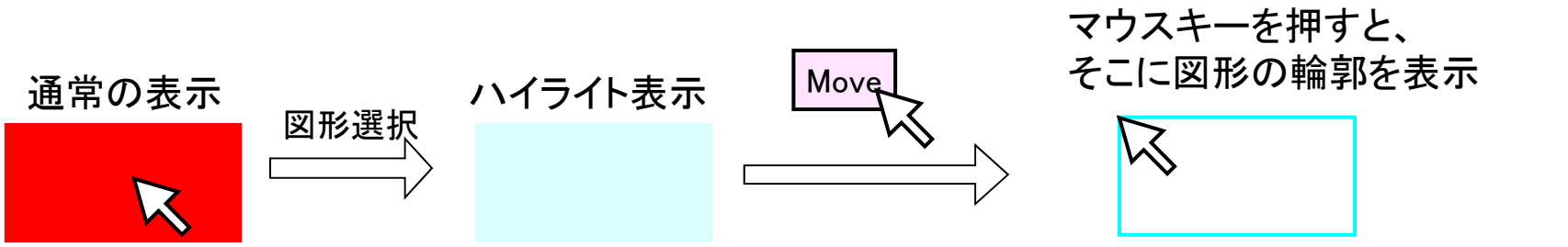


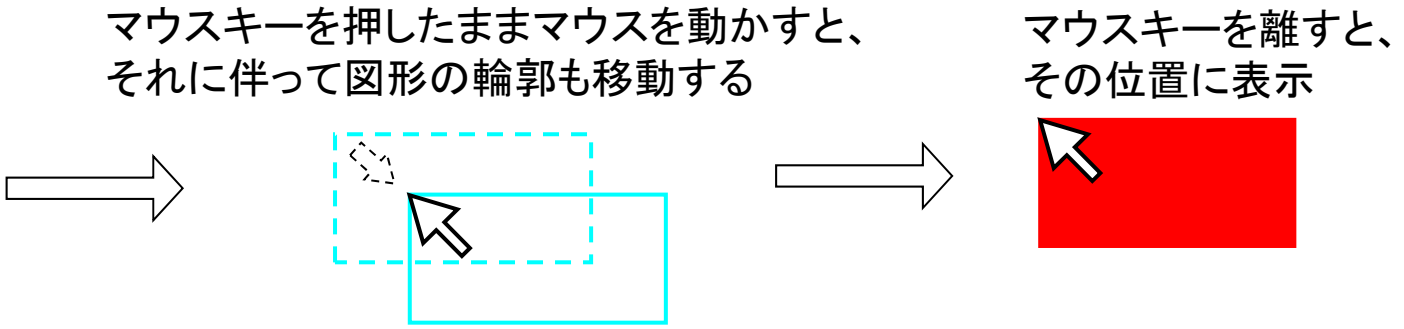
演習問題(8):オプション(1)

- ステップ6(これは必須ではありません(上級者向け)。
  第2回演習課題のラバーバンド未作成の場合は、まずそちらを作成し、
  それができてから、こちらに挑戦するのがよい。
- 図形エディタにドラッグ機能を追加する。

(1) 移動時のドラッグ

- マウスの移動とともに図形の輪郭を移動する。



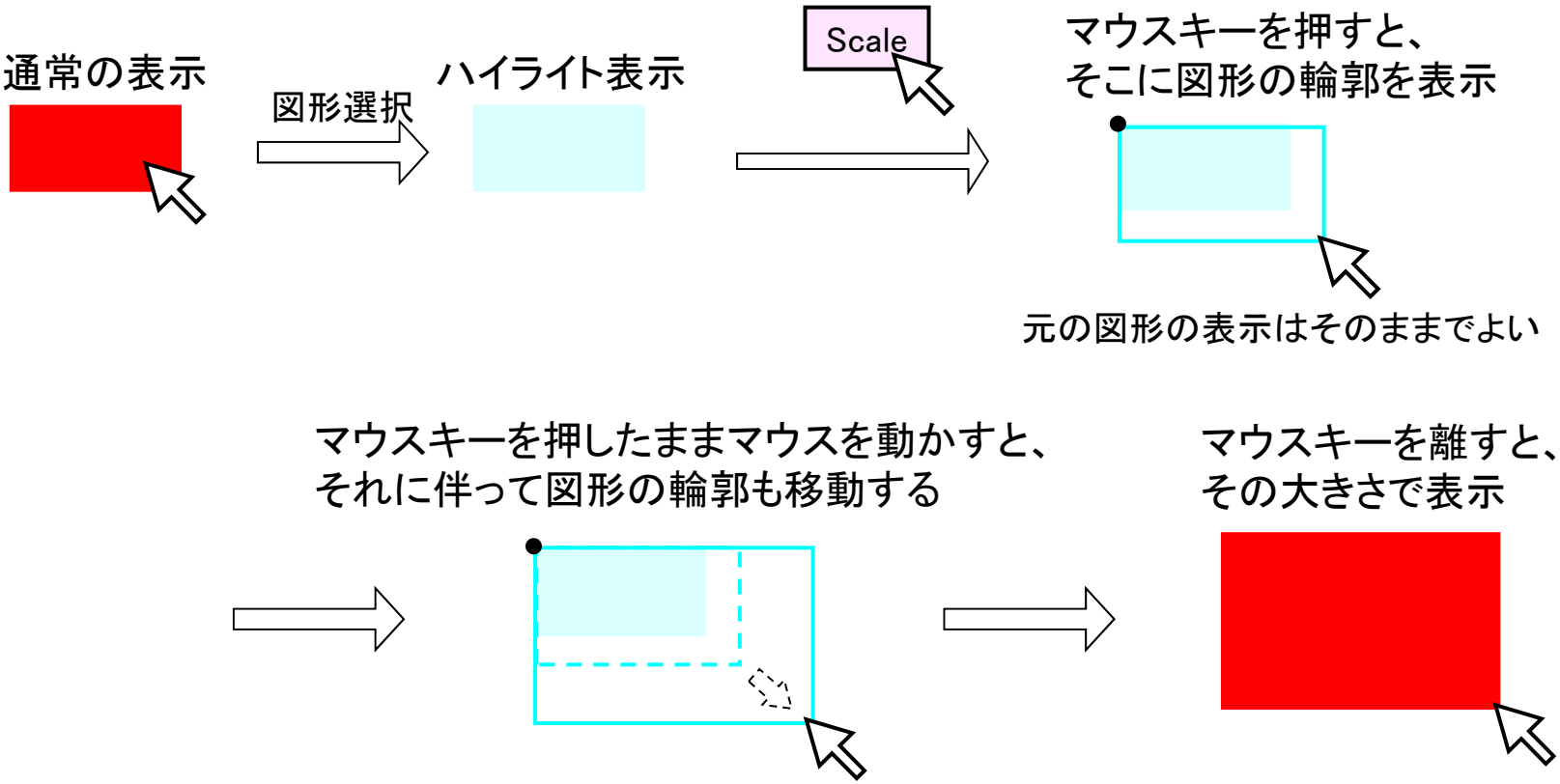


輪郭の色はシアン(Color.cyan)。ドラッグ中の画面表示は多少乱れてもOK。

演習問題(9):オプション(2)

(2) 拡大縮小時のドラッグ

・ マウスの移動とともに図形の輪郭(の大きさ)を変える



輪郭の色はシアン(Color.cyan)。ドラッグ中の画面表示は多少乱れてもOK。

演習問題(10)

グラフィックエディタは、かなり難しい(複雑な)問題です。
一度に作ろうとせず、1ステップずつ、確実に進めましょう。
今週は、少なくともステップ3までは作成するよう努力してください。
残りは、来週完成させましょう。
完成できる人は、完成させてください。

提出

- 今回作成したところまでの図形エディタのクラスのソースファイル“GraphicEditor.java”を提出する。