# 第１回レポート課題

学生証番号：2600170522-8

名前：　CAI Ming

## 1. 課題の説明

今回の課題を通して、JAVA 言語に基づいて、グラフィックを描くために使用できるツールを完成させました。

ソフトウェアインターフェイスを以下に示します（図 1)。

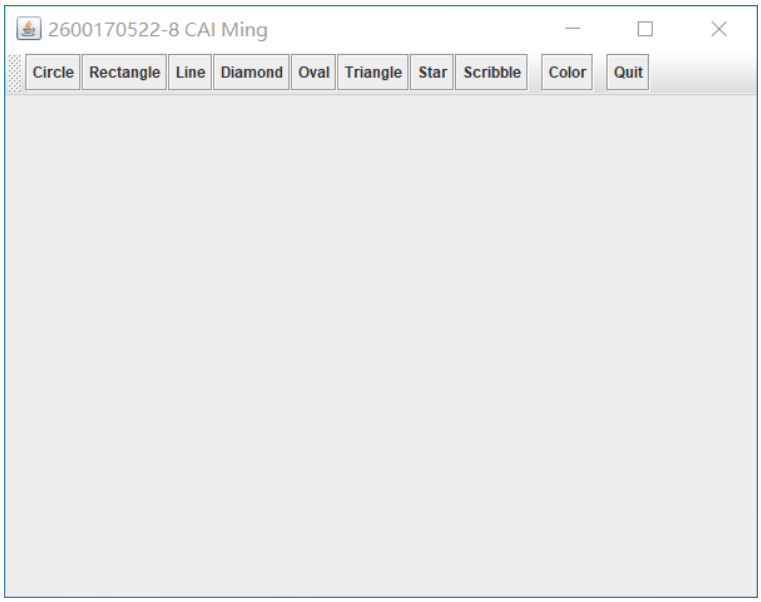


図 1

ソフトウェアの主要な UML 図を図 2 と図 3 に示している

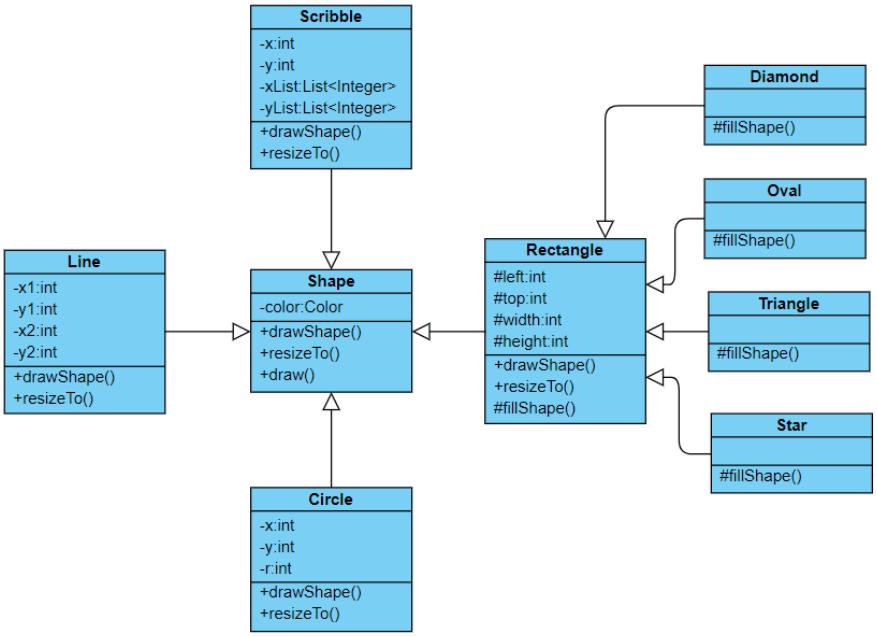


図 2

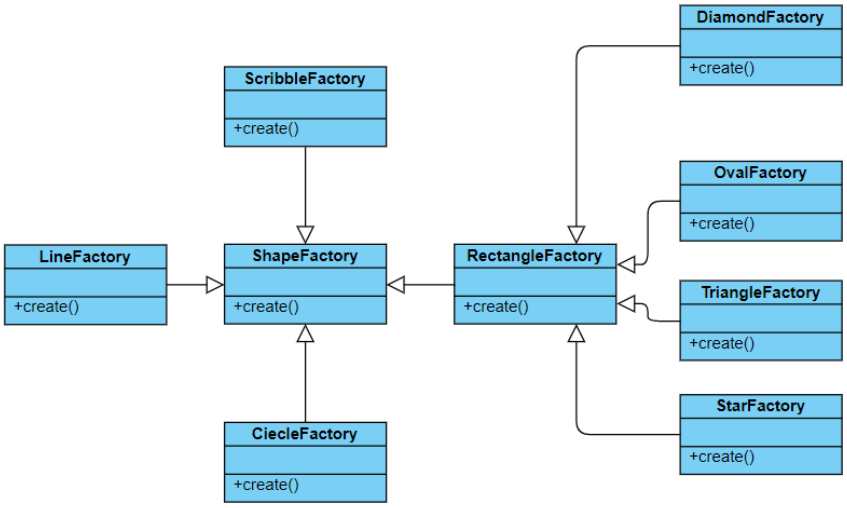


図 3

## 2. 実装した機能

### 1) それぞれの図形を描画できること

i. 円、長方形、線などの描画をできる

これまでの講義によると、円、長方形、線などの描画機能が実現されているので、もう繰り返し説明をしない。

ii. 楕円、菱形以などの描画をできる

これらは今回のレポートの基本要求であり、講義に従って実装した。

両方のクラスは Rectangle クラスから継承された。次に、これら 2 つのクラスを具体的に紹介する。

Oval と Diamond クラス図は図 4 のように表す。

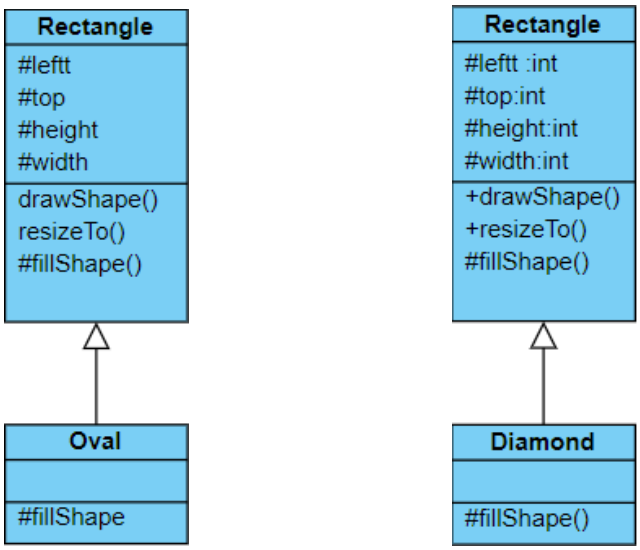


図 4

両方が fillShape()メソッドを override した。Oval の fillShape()は簡単に「java.awt.Graphics」の fillOval というメソッドを利用する（図 5）

```
public class Oval extends Rectangle {
  /* ここから */
  public Oval(int left, int top, int width, int height, Color color) {
      super(left, top, width, height, color);
  }

  @Override
  protected void fillShape(Graphics g, int left, int top, int width, int height) {
      g.fillOval(left, top, width, height);
  }
  /* ここまで */
}
```

図 5

Diamond の fillShape 関数は「java.awt.Graphics」の fillOval という関数を利用するため、全ての頂点座標の配列と座標の数が必要である。Diamond だから、4 つの頂点があり、頂点は四角形の辺の中点である。そして、fillShape メソッドを定義する（図 6）。

```java
public class Diamond extends Rectangle {
  /* ここから */
  public Diamond(int left, int top, int width, int height, Color color) {
      super(left, top, width, height, color);
  }
  @Override
  public void drawShape(Graphics g) {
      int[] xpoints = new int[4];
      int[] ypoints = new int[4];
      xpoints[0] = this.left + this.width / 2;
      xpoints[1] = this.left + this.width;
      xpoints[2] = xpoints[0];
      xpoints[3] = this.left;
      ypoints[0] = this.top;
      ypoints[1] = this.top + this.height / 2;
      ypoints[2] = this.top + this.height;
      ypoints[3] = ypoints[1];
      g.fillPolygon(xpoints, ypoints, 4);
  }
  /* ここまで */
}
```

図 6

iii. 三角形、星、落書きなどの描画をできる

三角形も星も Rectangle クラスから継承されたで、落書きは Line クラスから継承された。Triangle と Star と Scribble のクラス図は図 7 のように表す。

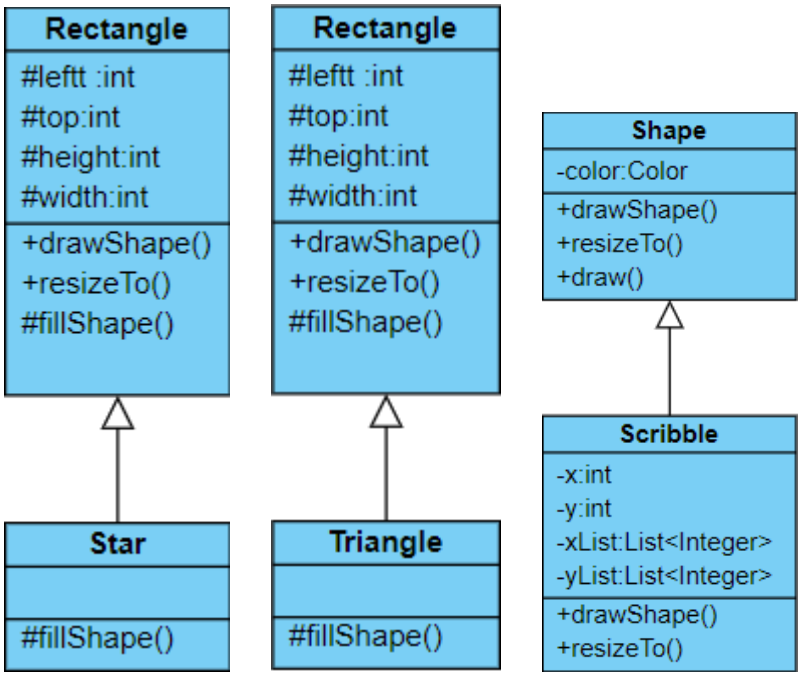


図 7

Triangle と Star は Diamond と Oval と同じで、fillShape メソッドだけを override した。

Triangle の関数 fillShape も全ての頂点座標の配列と座標の数が必要である。三つの頂点があり、一番上の頂点は四角形の上の辺の中点で、他の二つの頂点は四角形の下の角の座標である（図 8)。

```
public class Triangle extends Rectangle {
    /* ここから */
    public Triangle(int left, int top, int width, int height, Color color) {
        super(left, top, width, height, color);
    }
    @Override
    public void drawShape(Graphics g) {
        int[] xpoints = new int[3];
        int[] ypoints = new int[3];

        //(left + width/2, top), (left, top + heigth), (left + width, top + heigth)

        xpoints[0] = this.left + this.width / 2;
        xpoints[1] = this.left;
        xpoints[2] = this.left + this.width;

        ypoints[0] = this.top;
        ypoints[1] = this.top + this.height;
        ypoints[2] = this.top + this.height;

        g.fillPolygon(xpoints, ypoints, 3);
    }
    /* ここまで */
```

図 8

Star の fillShape メソッド順番に五つの頂点を配列に作り、「java.awt.Graphics」の drawPolygon の利用し、順に星の形を描く（図 9)。

```
public class Triangle extends Rectangle {
    /* ここから */
    public Triangle(int left, int top, int width, int height, Color color) {
        super(left, top, width, height, color);
    }
    @Override
    public void drawShape(Graphics g) {
        int[] xpoints = new int[3];
        int[] ypoints = new int[3];

        //(left + width/2, top), (left, top + heigth), (left + width, top + heigth)

        xpoints[0] = this.left + this.width / 2;
        xpoints[1] = this.left;
        xpoints[2] = this.left + this.width;

        ypoints[0] = this.top;
        ypoints[1] = this.top + this.height;
        ypoints[2] = this.top + this.height;

        g.fillPolygon(xpoints, ypoints, 3);
    }
    /* ここまで */
```

図 9

Scribble クラスは Line クラスから継承された。マウスの軌跡の座標の並びを保存するインスタンス変数を定義．resizeTo メソッドによる、そ

の時点のマウスの座標を上記の変数に追加する． drawShape メソッドによる、上記の変数に保存された座標の並びにしたがって、「java.awt.Graphics」の drawPolygon の利用し、線を描画していく。図 10 は Scribble の追加されたメソッド定義である。

```
@Override
public void drawShape(Graphics g) {
    int length = this.xList.size();
    int[] xListNew = new int[length];
    int[] yListNew = new int[length];
    System.out.println(xList);
    for(int i = 0; i < length; ++i){
        xListNew[i] = xList.get(i);
        yListNew[i] = yList.get(i);
    }
    g.drawPolyline(xListNew, yListNew, length);
}

@Override
public void resizeTo(int x, int y) {
    this.xList.add(x);
    this.yList.add(y);
}
```

図 10

2) Color を選べること

最初に設計する Color というボタンを設計した。JButton の addActionListener メソッドを利用して、マウスがこのボタン上にあるかどうかを確認。そして、showDialog メソッドによる、色を選ぶ画面を切り替える。図形の色を選べる。

コードは図 11 のようにある。

```
JButton colorButton = new JButton("Color");
colorButton.addActionListener((ae) -> {
  Color color = JColorChooser.showDialog(canvas, "Choose Color", canvas.getColor());
  if (color != null) {
    // キャンバスに色を設定する.
    canvas.setColor(color);
  }
});
toolBar.add(colorButton);
```

図 11

3) 大きさを変えること

マウスの移動によると、図形の大きさと位置を変化することができる。Rectangle の実現に基づいて、説明をあげる。

まずは、Rectangle の drawShape メソッドである（図 12)。

```java
@Override
public void drawShape(Graphics g) {
  int left = this.left;
  int top = this.top;
  int width = this.width;
  int height = this.height;
  if (width < 0) {
    left = left + width;
    width = -width;
  }
  if (height < 0) {
    top = top + height;
    height = -height;
  }
  // Ovalのの部分を切り出す.
  this.fillShape(g, left, top, width, height);
}

protected void fillShape(Graphics g, int left, int top, int width, int height) {
  g.fillRect(left, top, width, height);
}
```

図 12

四角形の大きさと位置は、left、top、width、height によって決まる。マウスをクリックするたびに新しい図形の位置が変化。Resize メソッドが width、height に影響し、大きさを決まる（図 13)。

```java
@Override
public void resizeTo(int x, int y) {
  this.width = x - this.left;
  this.height = y - this.top;
}
}
```

図 13

x と y はマウスの最後の座標である。

4) ToolBar を使用できる

まずはそれぞれのボタン（ToolButton）を作り、ToolBar の中に追加する。ToolButton のクラス図は図 14 である。

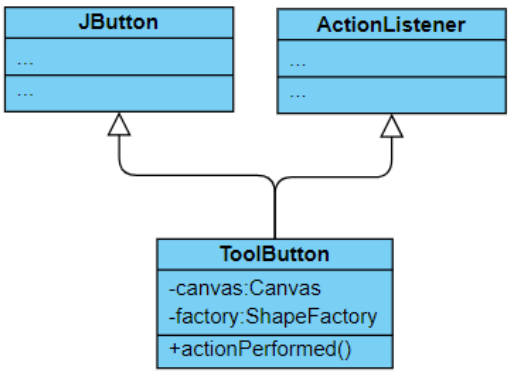


図 14

図 12 で、Canvas のクラス図は図 15 である。

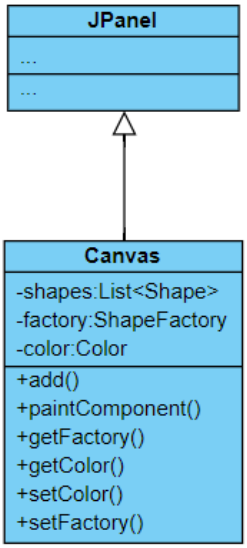


図 15

3. 実行結果（図 16、図 17）

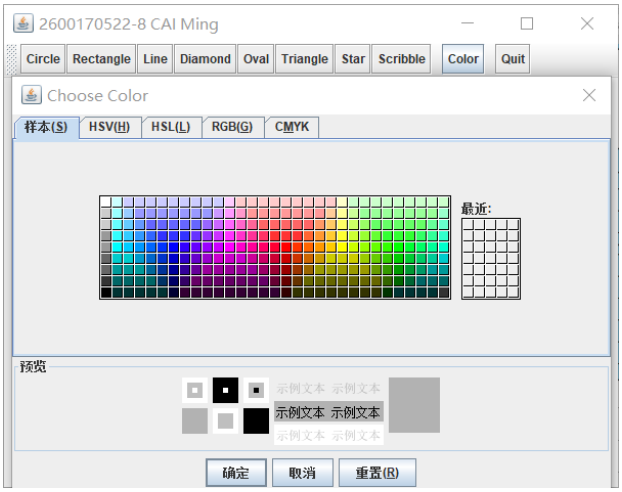


図 16

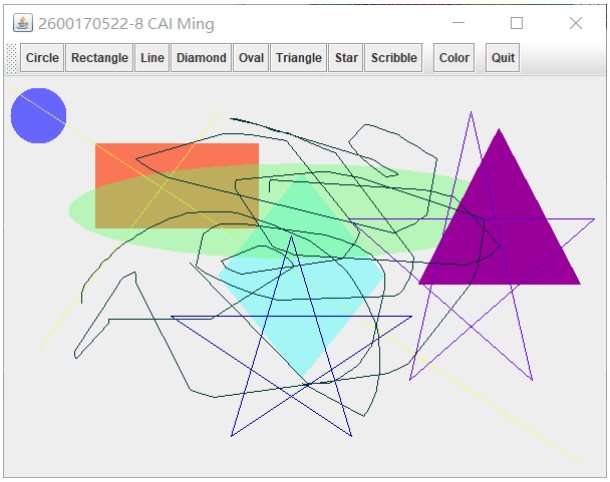


図 17

## 4. 感想

今回の課題を通して、java の使用を学びました。自分でこのソフトウェアをつくるのがうれしいです。

でも、ちょっと難しいと思います。理由は Java に精通していないでしょう。だから、この授業は講義によると教えるじゃないで、デモンストレーションしながら生徒を教えた方がいいと思います。ほかに、JAVA の原理も知りたいです。例えば、JAVA の VM とか、マルチスレッドです。使用方法しかを教えないで、Java を深く理解していないだろうと思います。