オブジェクト指向の考えを利用した Java のペア・プログラミング課題

文理学部情報科学科
5419045 高林 秀 5419071 出川慎吾

2021 年 8 月 7 日

概要

本稿では、今年度オブジェクト指向プログラミングの課題研究 2 として、前回の課題研究で制作した Java プログラムを、パートナーのソースコードと統合し、加えて Zoog 以外の新しいキャラクターの描画を行うものである。本演習には、Java を使用した。結果は、、、で、、、であった。

1 目的

本稿は、今年度オブジェクト指向プログラミングの課題研究として、Java のペア・プログラミングを行うものである。今回の課題は、前回の課題研究 1 で作成した Zoog クラスを、パートナーの Java ソースコードと統合すること、並びに Zoog とは異なる形、動作をする新しいキャラクターを描画すること、である。その際、Zoog と同一の interface 等を利用し、呼び出し側を共通化する。

2 課題概要

本課題は下記に示す、2 種類の課題から構成されている。以下にそれぞれの詳細を示す。

1. 課題研究 1 で制作したプログラムを 1 つのプログラムに統合する。
   自分とパートナーの課題研究 1 で作成した Zoog プログラムを比較し、1 つのプログラムで呼び出しが行えるよう統合する。その際、共通の抽象クラスや interface を利用するなど、適切な継承関係を与えること。
2. 新しいキャラクターを追加する。
   Zoog と見た目、動き方、攻撃方法 (なにをすると動きが止まるか) が異なる新しいキャラクターを描画する。その際、1 で使用した抽象クラスや interface を利用するなどし、呼び出し側を共通化することが望まれる。

なお担当者は、1 を出川慎吾、2 を高林秀が担当した。

3 設計方針

前章で示した各課題ごとに対する設計方針をまとめる。

1. 課題 1
   高林のプログラムに Zoog クラスとそれを継承した SecondZoog クラスの二つがあり、出川のプログラムもそれに類似した内容だったのでその二つをまとめることを考える。Zoog クラスを HorizontalZoog クラスという名前にし、HorizontalZoog クラスと SecondZoog クラスの上に抽象クラスである Zoog クラスを作成し move メソッドを抽象メソッドとする。そして二つのクラスが Zoog クラスを継承し、move メソッドをオーバーライドすることで異なる動きをする 2 つの Zoog を作成する。
2. 課題 2
   課題 2 も、指定された位置に特定の図形を描画しそれを動かすという意味では、課題 1 で作成した Zoog と同じである。そこで、新たに Zoog クラスと、新しいキャラクターのクラスを統合する Charactor と呼ばれる interface を作成し、呼び出し側の共通化を図る。その interface を implements した NewCaractor クラスそ作成し、その中で見た目、動き方、攻撃方法を定義している。なお動き方に関しては、既存の Zoog クラス系のオブジェクトでは、ウィンドウの端で跳ね返り、上下左右に移動するもの、ただ単に左右に移動するもの、の 2 種類となっている。そこで、新しいキャラクターはウィンドウの左端に到達するまで高速で右端から左端に移動する、といった動作を考えた。

4　ソースコード

課題 1, 課題 2 で作成したクラス、interface のコードを以下に示す。動作に必要なすべてのソースコードは巻末付録に添付している GoogleDrive あるいは GitHub からダウンロードいただきたい。

1. 課題
   - Zoog.java

```
package myprogram.Zoog_Obj;
import processing.core.*;

abstract class Zoog implements Charactor{
    public PApplet p;
    protected float x, y;
    protected float mx, my;
    protected int colorBW;
    protected boolean rightEye = true;
    protected boolean leftEye = true;

    public Zoog(PApplet pa) {
        p = pa;
        x = p.random(100, 200);
        y = p.random(100, p.height / 2);
        mx = p.random(1, 3);
        my = p.random(1, 3);
```

```
}

public void draw() {

    //zoog body
    p.stroke(0);
    p.fill(colorBW);
    p.rect(x, y, 20, 100);
    //zoog head
    p.fill(colorBW);
    p.ellipse(x, y-30, 60, 60);
    //zoog eye
    p.fill(0);
    if(leftEye == true) {
      p.ellipse(x-19, y-30, 16, 32);
    }
    if(rightEye == true) {
      p.ellipse(x+19, y-30, 16, 32);
    }
    p.fill(0);
    //zoog leg
    p.stroke(0);
    p.line(x-10, y+50, x-20, y + 60);
    p.line(x+10, y+50, x+20, y + 60);
}

//public void move();//抽象メソッド

/*
 public void move() {//zoog を動かすメソッド
    if(x + 20 >= p.width + 20 || x < 0) {//zoog がウィンドウの端に来た
際に跳ね返る動作を行う
        mx *= -1;
    }
     x += mx;//zoog を動かす

    if(leftEye == false && rightEye == false) {
       mx = 0;
    }
}
```

```
   */

    public void judgeClick(int MouseX, int MouseY) {
    if( (MouseX > x-27 &&  MouseX < x-11)  && (MouseY > y-46 && MouseY < y-14) ) {
      leftEye = false;
    }
    if( (MouseX > x+11 && MouseX < x+27) && (MouseY > y-46 && MouseY < y-14) ) {
      rightEye = false;
    }
  }

   public boolean isInRange(int MouseX, int MouseY) {
      return (p.dist(MouseX, MouseY, x-19, y-30) < 16 || p.dist(MouseX, MouseY, x-19,
   }
}
```

- SecondZoog.java

```
package myprogram.Zoog_Obj;

import processing.core.*;

public class SecondZoog extends Zoog{
    public SecondZoog(PApplet pa) {
        super(pa);
        colorBW = 255;
    }
    @Override
    public void move() {//move メソッドをオーバーライドする
        if(x < 10 || x > p.width) {
            mx *= -1;
        }
        if(y < 50 || y > p.height - 55) {
            my *= -1;
        }
        x += mx;//zoog を動かす
        y += my;
        if(leftEye == false && rightEye == false) {
           mx = 0;
           my = 0;
        }
```

```
    }
}
```

- HorizontalZoog.java

```
package myprogram.Zoog_Obj;

import processing.core.*;

public class HorizontalZoog extends Zoog {
   public HorizontalZoog(PApplet pa) {
      super(pa);

      colorBW = (int)p.random(1, 255);
   }

    public void move() {//move メソッドをオーバーライドする
        if(x + 20 >= p.width + 20 || x < 0) {//zoog がウィンドウの端に来た
際に跳ね返る動作を行う
            mx *= -1;
        }
         x += mx;//zoog を動かす

        if(leftEye == false && rightEye == false) {
           mx = 0;
        }
    }

}
```

2. 課題 2

- Charactor.java

```
package myprogram.Zoog_Obj;

interface Charactor {
    void draw();
    void move();
    void judgeClick(int MouseX, int MouseY);
    boolean isInRange(int MouseX, int MouseY);
}
```

- NewCharacotr.java

```
package myprogram.Zoog_Obj;
import processing.core.*;

public class NewCharactor implements Charactor{

    private PApplet p;
    private boolean moving = true;
    private boolean draw_text = false;
    private float x, y, mx;
    private boolean rightEye = true;
    private boolean leftEye = true;



    public NewCharactor(PApplet pa) {
        //super(pa);
        p = pa;
        x = p.random(100, 200);
        y = p.random(100, p.height / 2);
        mx = p.random(1, 3);
        //my = p.random(1, 3);
    }

    public void draw() {
        //body
        p.fill(190);
        p.rect(x, y, 100, 20);
        //head
        p.fill(100);
        p.ellipse(x-30, y, 60, 60);
        //eye
        p.fill(0);
        if(leftEye) {
            p.ellipse((x-30)-10, y, 15, 20);
        }
        if(rightEye) {
            p.ellipse((x-30)+10, y, 15, 20);
        }
        if(draw_text) {
            displayText("HEY!!Mr.Kitahara!!.How are you ??.", 60);
```

```
        }
        //leg
        //p.line(x+10, y+20, (x+50)-20, (y+20)+40);
        //p.line(x+40, y+20, (x+50)+20, (y+20)+40);
    }

    public void move(){
        if(moving) {
            if(x <= 0) {
                x = p.width;
            } else {
                x -= 3 * mx;
            }
        }
    }

    private void displayText(String text, int size) {
        p.textAlign(p.CENTER);
        p.textSize(size);
        p.fill(170, 20, 20);
        p.text(text, x, y + 20);
    }
    public boolean isInRange(int MouseX, int MouseY){
        return (p.dist(MouseX, MouseY, x-19, y-30) < 16 || p.dist(MouseX, MouseY, x-19
    }

    public void judgeClick(int MouseX,int MouseY) {
        mx = 0;
        draw_text = true;
    }
}
```

なお、今回のディレクトリ構成は以下の通りとなっている。・・・.java は Java ファイルを、.java が添付されていない名称はフォルダを示す。

- Report2
  - MyApplet.java
  - myprogram
    - Zoog_Obj
      - Charactor.java

- Zoog.java
- HorizontalZoog.java
- SecondZoog.java
- NewCharacotr.java

起動するには、親ディレクトリ Report2 に存在する MyApplet.java をコンソールで呼び出して起動する。

5　実行結果

以下、MyApplet.java 起動時の結果を示す。

5.1　結果のスクリーンショット

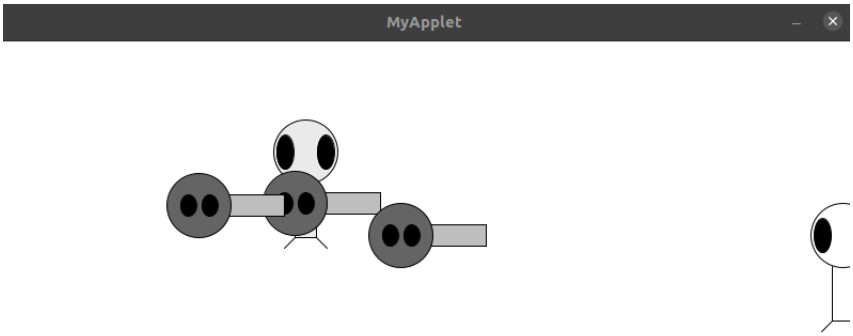

図 1　実行時の画像 1

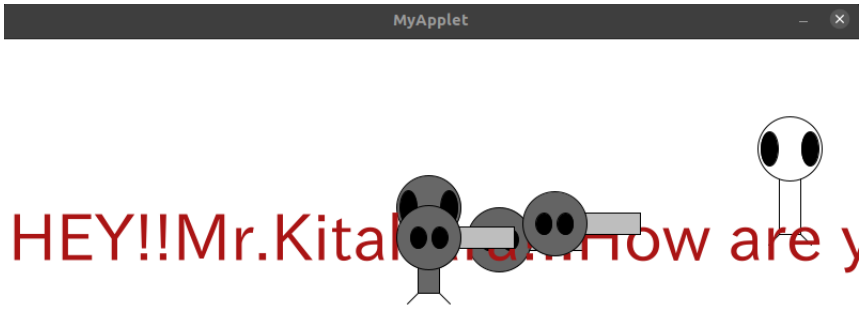

図 2　実行時の画像 2

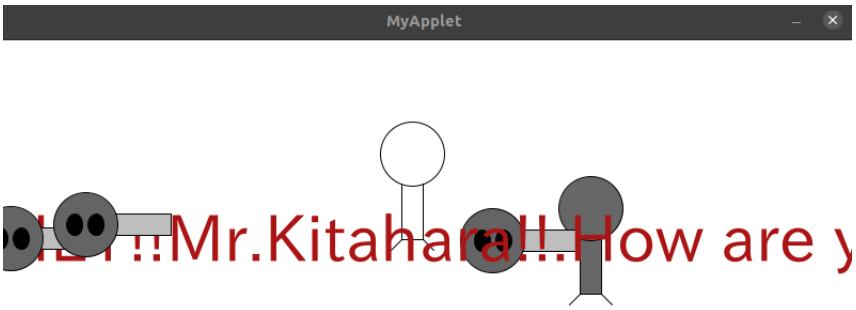

図 3　実行時の画像 3

5.2　結果の説明

起動直後、2 体の Zoog と、3 体の新しいキャラクターが描画される※画像 1。なお出現位置は、x 座標が 100 以上 200 未満の間でランダムに、y 座標が 100 以上ウィンドウの高さ ÷2 未満の間でランダムに決定する。

Zoog は左右に移動しウィンドウの端に到達すると跳ね返るものと、上下左右に移動しウィンドウの端で跳ね返るもの、の 2 種類の動作をする。眼の部分をマウスでクリックすると、眼が消滅するようになっており、両目を消失するとその Zoog の動作が止まるようになっている※画像 3。

3 体の新しいキャラクター (以降 ZoogFish と呼称する) は、ウィンドウ右端から左端へ高速に移動している。なお、3 体のうち 1 体の、頭をクリックすると、その動作が停止しウィンドウ上に"HEY!!Mr.Kitahara!!.How are you ??."の文字列を描画する※画像 2。その他 2 体はクリックしても、動作が停止しないようになっている（プログラム実行時に、どの ZoogFish が文字列を出力するか当てて見てほしい)。

なお、実行時の動画を下記の URL に掲載する。

- 動画リンク :`https://drive.google.com/file/d/12epGAvaELqIP-hwvUmeL9nQ_mFHwg4Rj/view?usp=sharing`

6　考察

7　まとめ

本稿では今年度オブジェクト指向プログラミングの課題研究 2 として、Java でのペア・プログラミングを行った。内容は、課題研究 1 で作成した Java プログラムを、1 つのい Java プログラムに統合すること、並びに新たなキャラクターを制作し統合した Zoog クラスと呼び出し側の共通化を図ることであった。全体的な結果としては、プログラムが正常に動作しているのでコードの統合はうまく行っていると言って良いだろう。

8　各メンバーの貢献内容

- 高林秀
  - 課題 2 の企画設計、コーディング担当。レポートの執筆。
- 出川慎吾
  - 課題 1 の企画設計、コーディング担当。課題 2 の設計考案。

9　巻末付録

本稿で使用している画像ファイルや、Java ファイルは以下のアドレスから入手することができる。必要に応じてダウンロードいただきたい。

- GoogleDrive:`https://drive.google.com/drive/folders/1QEt-NBptDGq2J1BgOyFnGUx8SMwL5oNc?usp=sharing`
- GitHub リポジトリ：`https://github.com/tsyu12345/OOP-report2`