オブジェクト指向プログラミングⅠ演習問題解答例
2011 年 6 月 20 日

学習ポイント

- 発生する例外の種類を知る
- クラスの定義（フィールド・メソッド・コンストラクタ）
- キーボードからの入力
- 既存のクラスの合成
- 継承によるクラスの拡張

1. 下記の 6 月 6 日問題 3 は double 型の値を 2 つ入力することで線形変換後の座標を得ることができるプログラムです。さまざまな入力を試し、実行結果がどのようになるかを調べなさい。また、エラーとなるのはどのような入力を行った場合であり、どのようなメッセージが出力されるかも調べなさい。

6 月 6 日問題 3
x 座標と y 座標からなる点のクラス Point を定義し、以下に示す座標の線形変換を行うメソッド linearTransfer を定義しなさい。コマンドライン引数で与えた x 座標と y 座標の値を変換した結果を下記のように画面表示するプログラムを作成しなさい。

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 6 & 4 \\ -2 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

```
Input Point : (5.0, 10.0)
 => Point after linearTransfer : (70.0, 0.0)
```

下記のプログラムを実行します。

```
1:  class Main{
2:      public static void main(String[] args){
3:              double x = Double.parseDouble(args[0]);
4:              double y = Double.parseDouble(args[1]);
5:              Point before = new Point(x,y);
6:              Point after = before.linearTransfer();
7:              System.out.println("Input Point : " + before + "¥n => Point after linearTransfer : " +
    after);
8:      }
9:  }
```

まず、コマンドラインに引数の数を考えてみましょう。
引数を与えないで実行すると、下記のメッセージが出力されます。

```
>java -cp class Main
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
        at Main.main(Main.java:3)
```

main メソッド内で ArrayIndexOutOfBoundsException というタイプの例外が発生しているということです。
at Main.main(Main.java:3)はこの例外が at に続く場所で発生していることを示しています。
すなわち、Main.java の 3 行目で例外が発生しています。
3 行目をみると
3:　　double x = Double.parseDouble(args[0]);
となっています。
ArrayIndexOutOfBoundsException という名前からわかるように、配列（Array）の添字(Index)が境界から外れている (OutOfBounds)というエラーですね。すなわち、コマンドライン引数がないので、コマンドライン引数を格納している args という配列には、要素が入っていないということです。そこで、5 行目のように配列にアクセスしようとするとエラーになります。

引数を 1 つだけ与えた場合にも、同様に、下記のメッセージが出力されます。
ここでは、6 行目で例外が発生しています。

```
>java -cp class Main 1
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 1
        at Main.main(Main.java:4)
```

4 行目をみると
4:　　double y = Double.parseDouble(args[1]);
となっています。
この場合は args[0]には、値 "1" が入っていますが、args[1]には値が入っていないため、アクセスしようとしてエラーになっています。
また、以下のように、文字を入力して見ると、別のメッセージが出ますね。

```
>java -cp class Main a
Exception in thread "main" java.lang.NumberFormatException: For input string: "a"
        at sun.misc.FloatingDecimal.readJavaFormatString(Unknown Source)
        at java.lang.Double.parseDouble(Unknown Source)
        at Main.main(Main.java:3)
```

ここで発生した例外は NumberFormatException です。
発生した場所は
　　at sun.misc.FloatingDecimal.readJavaFormatString(Unknown Source)
　　at java.lang.Double.parseDouble(Unknown Source)
　　at Main.main(Main.java:3)
となっています。これは下から読んでいきます。
main メソッドの parseDouble メソッドの readJavaFormatStrin メソッドでエラーが起こっているということです。各メソッドの前についている　java.lang.Double sun.misc.FloatingDecimal　は
それぞれ、java.lang パッケージ内の Double　クラス、sun.misc パッケージ内の FloatingDecimal クラスの意味で、各メソッドはそれぞれのクラスのメソッドです。() 内には例外の発生したソースコード内の行が示されています。Unknown Source と書かれているのは、API のクラスのため、ソースコード内の行数は分からないためです。

さて、この例外は何でしょうか？本来実数を入力すべきところに文字を入れたことに問題があるのは明らかです。3 行目は
3:　　double x = Double.parseDouble(args[0]);

であり、この parseDouble 内の処理で例外が発生しています。readJavaFormatString とはどのようなメソッドでしょうか？

2. 下記の Input クラスを使用して、6 月 6 日の問題 3 の入力をキーボードから行えるように変更しなさい。コメントにあるように,インスタンスメソッド inputDouble(String header) を利用する。どのように利用すればよいかをヒントを参考にして、考えなさい。

```
import java.io.*;
/**
 * キーボードからの入力を行うためのクラス
*/
public class Input{
  private String prompt;
  private final String ERROR_INPUT_DOUBLE = "入力値は実数ではありません。";
  private final String ERROR_INPUT = "エラー：入力に誤りがあります。";
  private void setPrompt(String prompt){
        this.prompt = prompt;
  }
  private String input() throws IOException{
        String line;
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        line = reader.readLine();
        return line;
  }
/**
 * 実数の入力
 * 引数で与えられたメッセージを出力後、キーボードから入力された文字列を実数として返す
メソッドである。入力された文字列が実数として解釈できない場合には、再入力を促す。
このメソッドを利用して、キーボードから実数を入力する。
 * @param String prompt :入力を促すメッセージ
 * @exception Exception : 入力値が実数として解釈できない場合の例外を処理する
*/
public double inputDouble(String prompt){
        this.setPrompt(prompt);
        System.out.println(this.prompt);
        try{
                String value = this.input();
                double n = Double.parseDouble(value);
                return n;
        } catch (Exception s){
                System.out.println(ERROR_INPUT_DOUBLE + this.prompt);
                return this.inputDouble(prompt);
        }
  }
/**
 * 文字列の入力
```

```
 * 引数で与えられたメッセージを出力後、キーボードから入力された文字列を返すメソッドで
ある。入力における例外は発生した場合には、再入力を促す。
 * @param String prompt :入力を促すメッセージ
 * @exception Exception : 入力が失敗した場合の例外を処理する
*/
public String inputString(String prompt){
        this.setPrompt(prompt);
        System.out.println(this.prompt);
        try{
                String n = this.input();
                return n;
        } catch (Exception s){
                System.out.println(ERROR_INPUT);
                return this.inputString(prompt);
        }
  }
}
```

※Input の使い方のヒント：下記のクラスとともにコンパイルし、いろいろな値を入力して試してみてください。

```
public class Main{
        public static void main(String[] args){
                Input in = new Input();
                double n = in.inputDouble("実数を入力してください：");
                System.out.println("入力された実数　" + n + "です。");
        }
}
```

解答例

```
class Main{
 public static void main(String[] args){
                Input input = new Input();
                String select = "Y";
                do{
                        double x = input.inputDouble("実数を入力してください。");
                        double y = input.inputDouble("実数を入力してください。");
                        Point before = new Point(x,y);
                        Point after = before.linearTransfer();
                        System.out.println("Input Point : " + before +
                                "¥n => Point after linearTransfer : " + after);
                        select = input.inputString("続けますか？Y/N を入力してください。");
                } while(select.equals("Y")||select.equals("y"));
        }
}
class Point{
        double x;
        double y;
```

```
        Point (){
                this.x = 0.0;
                this.y = 0.0;
        }
        Point(double x, double y){
                this.x = x;
                this.y = y;
        }
        Point linearTransfer(){
                double x0 = 6*this.x + 4* this.y;
                double y0 = (-2) * this.x + 1* this.y;
                return new Point(x0,y0);
        }
        public String toString(){
                return ("(" + this.x + ", " + this.y + ")");
        }

}
```

□Input クラスは省略

3. 「線分」は２つの端点で定義される。5 月 31 日の課題 3 で定義した「点」クラス Point を用いて「線分」を表す LineSegment クラスを定義し、線分の 2 つの端点を課題 3 で定義した線形変換で変換し、新たな線分を得るメソッド lineTransfer を定義しなさい。このクラスを用いて、１つの線分をキーボードからの入力を使って定義し、その線形変換後の線分を以下のように画面表示する main メソッドをもつクラス Main を定義しなさい。

```
Input LineSegment (5.0,10.0)->(10.0,20.0)
 => LineSegment after linearTransfer : (70.0, 0.0) -> (140.0,0.0)
```

解答例

```
class Main{
 public static void main(String[] args){
                Input input = new Input();
                String select = "Y";
                while(select.equals("Y")||select.equals("y")){
                        double x1 =
                        input.inputDouble("線分の端点 1 の x 座標を表す実数を入力してください。");
                        double y1 =
                        input.inputDouble("線分の端点 1 の y 座標を表す実数を入力してください。");
                        double x2 =
                        input.inputDouble("線分の端点 1 の x 座標を表す実数を入力してください。");
                        double y2 =
                        input.inputDouble("線分の端点 1 の y 座標を表す実数を入力してください。");
                        Point point1 = new Point(x1,y1);
                        Point point2 = new Point(x2,y2);
```

```
                        LineSegment beforeLine = new LineSegment(point1,point2);
                        LineSegment afterLine = beforeLine.linearTransfer();
                        System.out.println("Input LineSegment : " + beforeLine +
                                "¥n => LineSegment after linearTransfer : " + afterLine);
                        select = input.inputString("続けますか？Y/N を入力してください。");
                }
        }
}

class Point{
        double x;
        double y;
        Point (){
                this.x = 0.0;
                this.y = 0.0;
        }
        Point(double x, double y){
                this.x = x;
                this.y = y;
        }
        Point linearTransfer(){
                double x0 = 6*this.x + 4* this.y;
                double y0 = (-2) * this.x + 1* this.y;
                return new Point(x0,y0);
        }
        public String toString(){
                return ("(" + this.x + ", " + this.y + ")");
        }
}

class LineSegment{
        Point p1;
        Point p2;
        String image = "-->";
        LineSegment (double x, double y){
                this.p1 = new Point();
                this.p2 = new Point(x,y);
        }
        LineSegment (double x1, double y1,double x2, double y2){
                this.p1 = new Point(x1,y1);
                this.p2 = new Point(x2,y2);
        }

        LineSegment (Point p1, Point p2){
                this.p1 = p1;
```

```
            this.p2 = p2;
        }
        LineSegment linearTransfer(){
            return new LineSegment(p1.linearTransfer(),p2.linearTransfer());
        }
        public String toString(){
            return (this.p1 + this.image + this.p2);
        }
}
```

□Input クラスは省略

4. 2つの端点が(0,0) (2,2)である LineSegment を 1 つ作成し、順次、linearTransfer した線分を 10 個作成し、10 個の線分の形状を画面表示するプログラムを作成しなさい。10 個の線分は、後に利用することを考慮し、配列に格納しなさい。

解答例

```
class Main{
        public static void main(String[] args){
            LineSegment[] linesegments = new LineSegment[10];
            LineSegment l = new LineSegment(0,0,2,2);
            linesegments[0] = l;
            for(int i = 0; i < linesegments.length-1; i++){
                linesegments[i+1] = linesegments[i].linearTransfer();
            }
            for(int i = 0; i < linesegments.length; i++){
                System.out.println((i+1) + "番目の線分は" + linesegments[i]);
            }
        }
}

class Point{
        double x;
        double y;
        Point (){
            this.x = 0.0;
            this.y = 0.0;
        }
        Point(double x, double y){
            this.x = x;
            this.y = y;
        }
        Point linearTransfer(){
            double x0 = 6*this.x + 4* this.y;
            double y0 = (-2) * this.x + 1* this.y;
            return new Point(x0,y0);
        }
```

```
        public String toString(){
                return ("(" + this.x + ", " + this.y + ")");
        }

}

class LineSegment{
        Point p1;
        Point p2;
        String image = "-->";
        LineSegment (double x, double y){
                this.p1 = new Point();
                this.p2 = new Point(x,y);
        }
        LineSegment (double x1, double y1,double x2, double y2){
                this.p1 = new Point(x1,y1);
                this.p2 = new Point(x2,y2);
        }
        LineSegment (Point p1, Point p2){
                this.p1 = p1;
                this.p2 = p2;
        }
        LineSegment linearTransfer(){
                return new LineSegment(p1.linearTransfer(),p2.linearTransfer());
        }
        public String toString(){
                return (this.p1 + this.image + this.p2);
        }
}
```

5. LineSegment クラスを継承して、つぎの機能をもつクラス ThickLineSegment を定義しなさい。
   1) 線分の幅を表す属性 width を追加する。
   2) 線分の幅を設定するメソッド setWidth を定義する。
   3) 3 種類のコンストラクタを定義する。
      ① ２つの端点の x 座標と y 座標を指定したもの（幅の初期値は１とする）
      ② ２つの端点の x 座標と y 座標および幅を指定したもの
      ③ ２つの端点および幅を指定したもの
   4) 線分が幅を持つことを示すために、形状を生成するメソッド makeForm を定義する。指定された幅に応じて以下のような文字列を生成する。ただし、線分の長さを求めるメソッド length を適切なクラスに定義して、利用すること。
      - 幅が 1 の場合：文字"-" を線分の長さの小数点以下を切り上げた整数値の個数分連接した文字列
      - 幅が 2 の場合：文字"=" を線分の長さの小数点以下を切り上げた整数値の個数分連接した文字列
      - 幅が 3 以上の場合：文字"#" を線分の長さの小数点以下を切り上げた整数値の個数分連接した文字列

5) toString は端点を 4）で生成した形状の情報で連結して文字列化する。

以下は出力例

```
Input LineSegment : (10.0, 10.0)======================(20.0, 30.0)
 => LineSegment after linearTransfer : (100.0, -10.0)==========================
================================================================================
================================(240.0, -10.0)
```

解答例

```
class Main{
 public static void main(String[] args){
                Input input = new Input();
                String select = "Y";
                while(select.equals("Y")||select.equals("y")){
                        double x1 =
                         input.inputDouble("線分の端点 1 の x 座標を表す実数を入力してください。");
                        double y1 =
                         input.inputDouble("線分の端点 1 の y 座標を表す実数を入力してください。");

                        double x2 =
                         input.inputDouble("線分の端点 1 の x 座標を表す実数を入力してください。");
                        double y2 =
                         input.inputDouble("線分の端点 1 の y 座標を表す実数を入力してください。");
                        int w = input.inputInteger("線分の太さを表す正の整数を入力してください。");
                        Point point1 = new Point(x1,y1);
                        Point point2 = new Point(x2,y2);
                        ThickLineSegment beforeLine = new ThickLineSegment(point1,point2,w);
                        ThickLineSegment afterLine = beforeLine.linearTransfer();
                        System.out.println("Input LineSegment : " + beforeLine +
                                "¥n => LineSegment after linearTransfer : " + afterLine);
                        select = input.inputString("続けますか？Y/N を入力してください。");
                }
        }
}

class Point{
        double x;
        double y;
        Point (){
                this.x = 0.0;
                this.y = 0.0;
        }
        Point(double x, double y){
                this.x = x;
                this.y = y;
```

```
        }
        Point linearTransfer(){
                double x0 = 6*this.x + 4* this.y;
                double y0 = (-2) * this.x + 1* this.y;
                return new Point(x0,y0);
        }
        public String toString(){
                return ("(" + this.x + ", " + this.y + ")");
        }
}

class LineSegment{
        Point p1;
        Point p2;
        String image = "-->";
        LineSegment (double x, double y){
                this.p1 = new Point();
                this.p2 = new Point(x,y);
        }
        LineSegment (double x1, double y1,double x2, double y2){
                this.p1 = new Point(x1,y1);
                this.p2 = new Point(x2,y2);
        }
        LineSegment (Point p1, Point p2){
                this.p1 = p1;
                this.p2 = p2;
        }
        LineSegment linearTransfer(){
                return new LineSegment(p1.linearTransfer(),p2.linearTransfer());
        }
        double length(){
                return Math.sqrt(Math.pow(p1.x-p2.x,2) + Math.pow(p1.y-p2.y,2));
        }
        public String toString(){
                return (this.p1 + this.image + this.p2);
        }
}

class ThickLineSegment extends LineSegment{
        int width;
        ThickLineSegment(double x1, double y1,double x2, double y2){
                super(x1, y1, x2, y2);
                this.width = 1;
        }
        ThickLineSegment(double x1, double y1,double x2, double y2, int w){
```

```
                super(x1, y1, x2, y2);
                this.width = w;
        }
        ThickLineSegment(Point p1, Point p2, int w){
                super(p1,p2);
                this.width = w;
        }
        void setWidth(int width){
                this.width = width;
        }
        ThickLineSegment linearTransfer(){
                return new ThickLineSegment(p1.linearTransfer(),p2.linearTransfer(),this.width);
        }
        String makeForm(){
                int n = (int)Math.ceil(this.length());
                String symbol = "-";
                if(this.width == 2) {
                        symbol = "=";
                } else if (this.width >=3){
                        symbol = "#";
                }
                String form = "";
                for(int i = 0 ; i < n ; i++){
                        form = form + symbol;
                }
                return form;
        }
        public String toString(){
                this.image = this.makeForm();
                return (super.toString());
        }
}
```

※Input クラスは省略