オブジェクト指向プログラミングⅠ演習問題

2017 年 6 月 12 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 5 月 29 日問題 4 は double 型の値を 4 つ入力することで線形変換後の線分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、Scomb のアンケートで報告しなさい。

5 月 29 日問題 4

「線分」は 2 つの端点で定義される。問題 2 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 つの端点を問題 2 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 linearTransfer を定義しなさい。このクラスを用いて、1 つの線分を定義するのに必要な値をコマンドライン引数で与えて、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)
```

2. 下記の Input クラスを使用して、5 月 29 日問題 4 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);
              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。どのような動作をしますか？

```
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。");
        }
}
```

3. LineSegment クラスを継承して、つぎの機能をもつクラス DottedLineSegment を定義しなさい。
   1) 線分の間隔を表す属性 interval を追加する。
   2) 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（間隔の初期値は 0 とする）
      ② 2 つの端点の x 座標と y 座標および間隔を指定したもの
      ③ 2 つの端点および間隔を指定したもの
   3) 線分が間隔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。指定された間隔に応じて、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、間に間隔で指定された個数の空白文字を挟んで連接した文字列を生成する。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。
   4) 線分の 2 つの端点を端点の線形変換で変換し、新たな線分を得るメソッド linearTransfer を再定義しなさい。
   5) toString はこれまでの LineSegment の情報に加えて 4）で生成した形状の情報も付加して文字列化する。

以下は間隔 2 の場合の出力例である。キーボードからいろいろな値を入力して確認しなさい。

```
Input LineSegment : (10.0, 10.0)-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  (20.0, 20.0)
 => LineSegment after linearTransfer : (100.0, -10.0)-  -  -  -  -  -  -  -  -  -  -  -  -  -  -
-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  (200.0, -20.0)
```