Platform-based Programming

#4 Interface

2018년 2학기

Program Output

- Make a Java program that can manage Circle and Line

```
Enter Operation String! addc
10 20 10
[[10, 20] 10 314]
Enter Operation String! addl
10 10 20 40
[[10, 10] [20, 40] 32]
Enter Operation String! list
[[[10, 20] 10 314], [[10, 10] [20, 40] 32]]
Enter Operation String! sorta
Enter Operation String! list
[[[10, 10] [20, 40] 32], [[10, 20] 10 314]]
Enter Operation String! clear
Enter Operation String! list
[]
```

Problem Output

```
Enter Operation String! addl
10 10 20 30
[[10, 10] [20, 30] 22]
Enter Operation String! addc
10 20 30
[[10, 20] 30 2827]
Enter Operation String! list
[[[10, 10] [20, 30] 22], [[10, 20] 30 2827]]
Enter Operation String! sortd
Enter Operation String! list
[[[10, 20] 30 2827], [[10, 10] [20, 30] 22]]
Enter Operation String! quit
Bye
```

Program Skeleton

```
enum OperationKind {ADD_C, ADD_L, LIST, CLEAR, SORT_A, SORT_D,
      QUIT, INVALID} ;
enum SortKind {ASCENDING, DESCENDING} ;

public class SortInterfaceTest {

      private static Scanner scanner = new Scanner(System.in);
      private static List<MyComparable> comparableList =
            new ArrayList<MyComparable>() ;

      public static void main(String[] args) {
            while ( true ) {
                  final OperationKind op = getOperation() ;
                  if ( op == OperationKind.QUIT ) {
                        System.out.println("Bye") ;
                        break;
                  }
                  if ( op == OperationKind.INVALID ) {
                        System.out.println("Invalid Operation!") ;
                        continue ;
                  }
```

```
switch ( op ) {
                case ADD_L : {
                        final Line newLine = createLine() ;
                        System.out.println(newLine) ;
                        break ;
                }
                case ADD_C : {
                        final Circle newCircle = createCircle() ;
                        System.out.println(newCircle) ;
                        break ;
                }
                case SORT_A:
                        sortList(comparableList, SortKind.ASCENDING) ;
                        break ;
                case SORT_D:
                        sortList(comparableList, SortKind.DESCENDING) ;
                        break ;
                case CLEAR:
                        comparableList.clear() ;
                        break ;
                case LIST:
                        System.out.println(comparableList) ;
                        break ;
                }
        }
    }
```

```
[[10, 10] [20, 40] 32]
```

```
[[10, 20] 10 314]
```

```
[[[10, 10] [20, 40] 32], [[10, 20] 10 314]]
```

```
private static OperationKind getOperation() {
        System.out.print("Enter Operation String! ") ;
        final String operation = scanner.next() ;

        OperationKind kind = OperationKind.INVALID ;
        if ( operation.equalsIgnoreCase("ADDL"))
            kind = OperationKind.ADD_L ;
        if ( operation.equalsIgnoreCase("ADDC"))
            kind = OperationKind.ADD_C ;
        else if ( operation.equalsIgnoreCase("LIST"))
            kind = OperationKind.LIST ;
        else if ( operation.equalsIgnoreCase("SORTA"))
            kind = OperationKind.SORT_A ;
        else if ( operation.equalsIgnoreCase("SORTD"))
            kind = OperationKind.SORT_D ;
        else if ( operation.equalsIgnoreCase("CLEAR"))
            kind = OperationKind.CLEAR ;
        else if ( operation.equalsIgnoreCase("QUIT"))
            kind = OperationKind.QUIT ;
        return kind ;
    }
```

```
    private static Circle createCircle() {
        final int x = scanner.nextInt() ;
        final int y = scanner.nextInt() ;
        final int radius = scanner.nextInt() ;

        final Circle newCircle = new Circle(new Point(x, y), radius) ;
        comparableList.add(newCircle) ;
        return newCircle ;
    }
    private static Line createLine() {
        final int x1 = scanner.nextInt() ;
        final int y1 = scanner.nextInt() ;

        final int x2 = scanner.nextInt() ;
        final int y2 = scanner.nextInt() ;

        final Line newLine = new Line(new Point(x1, y1), new Point(x2,
y2)) ;
        comparableList.add(newLine) ;
        return newLine ;
    }
```

```
private static void sortList(
        final List<MyComparable> comparableList, final SortKind sortKind) {
            // You need to implement this method
        }
```

```
// Point.java
public class Point {
        private int x, y ;
        …
}
```

```
// MyComparable.java
public interface MyComparable {
        public int compareTo(final MyComparable other) ;
        public long getSize() ;
}
```

```
// Line.java
public class Line implements MyComparable {
        private Point point1, point2 ;
        …
}
```

```
// Circle.java
public class Circle implements MyComparable {
        private Point center ;
        private int radius ;
        …
}
```

How to Submit

- Upload a zipped file named “ShapeInterface.zip”