Canvas

General introduction

This assignment is to exercise inheritance and polymorphsim in java programming language. In this assignment, the class names should be the same as requirements and do not modify or remove any fields or methods that have been already defined in document. You can add other classes, fields and methods if you need. You can also override superclass' methods if necessary.

What to Submit?

Location.java

Direction.java

Shape.java

Circle.java

RightTriangle.java

Class Location

```
public class Location {
private int x;
private int y;
public Location(int x, int y) {
this.x = x;
this.y = y;
}
public int getX() {
return x;
}
public int getY() {
return y;
}
public String toString() {
return String.format("(%d,%d)", x, y);
```

Fields:

The two private data fields x, y represent the vertical position and horizontal position respectively.

Enum Class Direction

```
public enum Direction {
LEFT_UP, LEFT_DOWN, RIGHT_UP, RIGHT_DOWN
}
```

The direction of the right angle of a right triangle, for example:

LEFT_UP:

**
*
LEFT_DOWN:
*
**

RIGHT_UP:

**
*
RIGHT_DOWN:
*
**

Abstract Class Shape
It is an abstract class, which is the super class of
Circle and
RightTriangle .
This class is based on the layout design of Java Swing, with the top-left corner as the origin. The dimensions of the grids can be used as the width and height of the graphics.
Fields:
grids
protected char[][] grids;
grids 表示一个矩形的网格，它的行数等于矩形中竖直方向最长的长度，它的列数表示矩形水平方向最宽的宽度。grids 是由 char 类型字符组成，里面每一个小格子是一个像素，如果有元素则用 pattern 表示，如果没有元素则用一个空格表示。
grids represents a grid of a rectangle, where its count of rows equals the longest vertical length within the rectangle, and the count of columns equals the widest horizontal width of the rectangle.
grids is composed of characters of
char type, where each small cell inside it
represents a pixel. If there is an element, it is represented by a
pattern , and if there is no
element, it is represented by a space.
pattern
protected char pattern;
The pattern of the shape, and it is used to fill the shape in grids.
location
protected Location location;
The location of current shape. The location means the left-up corner of grids. For example, if the location of the shape is (5, 6) and the shape is in a canvas, so that the pixel
location of canvas is (5, 6). The class
grids[0][0] in the
Location doesn't have any specific role in the current
assignment, because there is no requirement of any class about canvas in this assignment, but it

will be used in assignment 6.
Methods:
Two-parameter constructor()
public Shape(Location location, char pattern);
Set the original value of Location and the concrete pattern in Shape
getGrids()
public char[][] getGrids();
Return the reference of the attribute
grids
fillGrids()
public abstract void fillGrids();
Fill in the specific shapes with the pattern style into the grids. How to fill grids is determined by the concrete subclass of shape.
enlarge()
public abstract void enlarge();
Enlarge the shape. How to enlarge a shape is determined by the concrete subclass of shape.
shrink()
public abstract void shrink();
Shrink the shape. How to shrink a shape is determined by the concrete subclass of shape
area()
public abstract int area();
Return the count of patterns that being filled in grids.
Class Circle and Class RightTriangle
Circle and RightTriangle are two subclasses of Shape.
Fields in Circle
radius
private int radius;
The radius of Circle. The value of radius is in
The height and width of grids are all
.
.
Fields in RightTriangle
width
private int width;
The width of RightTriangle, and it is also the width of grids. The value of width is in
height
private int height;
.
The height of RightTriangle, and it is also the height of grids. The value of height is in
d
private final Direction d;
.
The direction of the right angle of a right triangle. The example of which is in the introduction of Enum Class Direction.
Methods
constructor()

In Circle, design a constructor with three attributes such as:
public Circle(Location location, char pattern, int radius);
Then given those three fields a initial value.
In RightTriangle, design a constructor with five attributes such as:
public RightTriangle(Location location, char pattern, int width, int height,
Direction d);
Then given those five fields a initial value.
Both Circle() and RightTriangle() should initial
grids by the original value, which mean
you'd better invoke the method fillGrids() in constructor.
fillGrids()
All subclasses of
Shape should implement the method
you should instantiate the attribute
fillGrids() . First of all, in this method,
grids , which means you need create a two-dimensional
array of char type for the attribute
grids .
In Circle, the length and width of the two-dimensional array are all
the grids with
pattern including:
Inside the circle
, and then fill
Intersect with the circumference(圆周) of the circle.
For other grids, including tangent(相切) to the circle, fill with a space.
For example:
If the radius = 5, the height and width of grids are 10 and 10 respectively, and the grids to be filled
are signed by blue color below.
In RightTriangle, the length and width of the two-dimensional array equals to the height and
width of rightTriangle, and then fill the grids with
pattern including:
Inside the three sides of right-triangle.
Intersect with the hypotenuse(斜边) of right-triangle.
For other grids, fill with a space.
For example:
If the height = 8, width = 4, and the direction of right-angle is
signed by blue color below.
LEFT_DOWN , the grids to be filled are
enlarge()
All subclasses of
Shape should implement the method enlarge().
In Circle, the radius should be increased by 1.
In RightTriangle, the height and width should be increased by 1.
Both Circle and RightTriangle should update
grids by the enlarged fields.
shrink()

All subclasses of
Shape should implement the method shrink().
In Circle, the radius should be decreased by 1.
In RightTriangle, the height and width should be decreased by 1.
Both Circle and RightTriangle should update
area()
All subclasses of
grids by the shrinked fields.
Shape should implement the method area().
Both Circle and RightTriangle should return the count of filled grids.
toString()
public String toString();
You can design it in the super class
Shape or in both subclasses
Circle and
RightTriangle , as
long as when invoking the toString method, it will return a String as the format below.
[Classname]: ([x],[y]) area=[area] pattern=[pattern]
For example:
Circle: (1,0) area=60 pattern=*
or
RightTriangle: (0,1) area=12 pattern=X
Code to be Test

```
public static void main(String[] args) {
Location p1 = new Location(1, 0);
Shape s1 = new Circle(p1, '*', 5);
char[][] grids = s1.getGrids();
for (char[] grid : grids) {
System.out.println(grid);
}
System.out.printf("Grids: height = %d, width = %d\n", grids.length,
grids[0].length);
System.out.println(s1);
s1.shrink();
grids = s1.getGrids();
for (char[] grid : grids) {
System.out.println(grid);
}
System.out.printf("Grids: height = %d, width = %d\n", grids.length,
grids[0].length);
System.out.println(s1);
}
```

Result:

Grids: height = 10, width = 10
Circle: (1,0) area=88 pattern=*

Grids: height = 8, width = 8
Circle: (1,0) area=60 pattern=*
Test for RightTriangle:

```
public static void main(String[] args) {
Location p1 = new Location(0, 1);
Shape s1 = new RightTriangle(p1, 'X', 7, 3, Direction.RIGHT_UP);
char[][] grids = s1.getGrids();
for (char[] line : grids) {
System.out.println(line);
}
System.out.printf("Grids height = %d, width = %d\n", grids.length,
grids[0].length);
System.out.println(s1);
s1.enlarge();
grids = s1.getGrids();
for (char[] line : grids) {
System.out.println(line);
}
System.out.printf("Grids height = %d, width = %d\n", grids.length,
grids[0].length);
System.out.println(s1);
}
```

Result:

```
XXXXXXX
XXXXX
XXX
Grids height = 3, width = 7
RightTriangle: (0,1) area=15 pattern=X
XXXXXXXX
XXXXXX
XXXX
XX
Grids height = 4, width = 8
RightTriangle: (0,1) area=20 pattern=X
```

Introduce
This part is to exercise the interface, and implement classes of interface and Comparable or Comparator Interface.
In this part, we will continue to use the
Direction ,
Location ,
Shape ,
Circle ,
RightTriangle classes that have been designed . You can use the classes designed by yourself, or use the classes we provided.
In this assignment, the names class like
ShapeCanvas ,
AvoidConflictShapeCanvas and
OverLapShapeCanvas should be the same as requirements and do not modify or remove any fields or methods that have been already defined in document. You can add other classes, fields and methods if you need. You can also override superclass' methods if necessary.
What to Submit?
Location.java
Direction.java
Shape.java
Circle.java
RightTriangle.java
ShapeCanvas.java
AvoidConflictShapeCanvas.java
OverLapShapeCanvas.java
Interface ShapeCanvas
It is an interface.
Methods
addShape()
public boolean addShape(int x, int y, char pattern, int... params);
Implement the method in its implement classes.
getSpaceGridCount()
public int getSpaceGridCount();
Implement the method in its implement classes.
This method is to return an int value, which represents how many space value in canvas.
getShapeCount()
public int getShapeCount();
Implement the method in its implement classes.
This method is to return an int value, which represents how many shapes being added into canvas successfully.
getShapesByArea()
public List getShapesByArea();
Implement the method in its implement classes.

This method is to return a List, which contains all shapes in canvas, and sort the shape as :
Ascending order of the area of shapes.
If shapes with a same area, sorted by the ascending order of its character value of pattern.
Shapes with a same area and same pattern cannot appear in any test case in this assignment.
getShapesByLocation()
public List getShapesByLocation();
Implement the method in its implement classes.
This method is to return a List, which contains all shapes in canvas, and sort the shape as :
Ascending order of the x value in location.
If shapes with a same value of x, sorted by the ascending order of its y value in location.
If shapes with a same location, sorted by the ascending order of its character value of
pattern.
Shapes with a same location and same pattern cannot appear in any test case in this assignment.
getCanvas()
public char[][] getCanvas();
Implement the method in its implement classes.
This method is to return a
' '
char[][] array, which represents a painted canvas. If a cell should be
painted, it will be the pattern. If a cell haven' t been paint, it will be a space like
AvoidConflictShapeCanvas
It is the concrete implement class of the interface
ShapeCanvas
Attributes
shapes
private List shapes;
Design a attribute shapes, which is to store the successfully added shapes.
canvas
private char[][] canvas;
Using a
char[][] array to represent the canvas. The initial value of each grid in canvas is a space
' ' , which mean an empty grid.
Methods:
You need implement all abstract methods in the interface
Constructor()
public AvoidConflictShapeCanvas(int rows, int cols)
ShapeCanvas
Design a constructor with two parameters, rows represents the height of canvas and cols
represents the width of canvas. Those are also the rows and columns of the private field
canvas .
addShape()
@Override
public boolean addShape(int x, int y, char pattern, int... params)
This method is to add a shape into canvas, and the parameter means:
x and y: the location x and y of shape
pattern: the pattern of shape

params: The length of params only be 1 or 3 in all test cases in this assignment.
For circle: the length of params is 1, which represents the radius of circle
For RightTriangle: the length of params is 3.
The first one is the width of RightTriangle
The second one is the height of RightTriangle
LEFT_UP direction.
The third one is the index of Direction. For example, if the third value is 0, it
means
The return value is determined by whether the shape can be added successfully, which means if it has no conflict when adding a shape, the method returns true, otherwise it returns false.
The conflict includes:
The shape is out of the bound of canvas.
Overlap Conflict: Has non-space grid of shape in corresponding location of the canvas is also non-space.
Test code：

```
public static void main(String[] args) {
ShapeCanvas shapeCanvas = new AvoidConflictShapeCanvas(20, 20);
System.out.println(shapeCanvas.addShape(0, 2, 'A', 5, 3, 1));
System.out.println(shapeCanvas.addShape(6, 8, 'B', 5, 7, 2));
System.out.println(shapeCanvas.addShape(8, 12, 'C', 5));
System.out.println(shapeCanvas.addShape(6,6,'D',5,7,1));
System.out.println(shapeCanvas.addShape(0,8,'E',3));
shapeCanvas.getShapesByArea().forEach(System.out::println);
shapeCanvas.getShapesByLocation().forEach(System.out::println);
for (char[] line:shapeCanvas.getCanvas()) {
System.out.println(line);
}
}
```

Result:

```
true
true
false
true
true
RightTriangle: (0,2) area=11 pattern=A
RightTriangle: (6,8) area=23 pattern=B
RightTriangle: (6,6) area=23 pattern=D
Circle: (0,8) area=36 pattern=E
RightTriangle: (0,2) area=11 pattern=A
Circle: (0,8) area=36 pattern=E
RightTriangle: (6,6) area=23 pattern=D
RightTriangle: (6,8) area=23 pattern=B
AA EEEEEE
AAAA EEEEEE
AAAAA EEEEEE
EEEEEE
```

EEEEEE
EEEEEE
D BBBBB
DDBBBBB
DDDBBBB
DDD BBB
DDDDBBB
DDDDDBB
DDDDD B
OverLapShapeCanvas
It is the concrete implement class of the interface
ShapeCanvas
Attributes
shapes
private List shapes;
Design a attribute shapes, which is to store the successfully added shapes.
canvas
private char[][] canvas;
Using a
char[][] array to represent the canvas. The initial value of each grid in canvas is a space
' ' , which mean an empty grid.
Methods:
You need implement all abstract methods in the interface
Constructor()
public OverLapShapeCanvas(int rows, int cols)
ShapeCanvas
Design a constructor with two parameters, rows represents the height of canvas and cols
represents the width of canvas. Those are also the rows and columns of the private field
canvas .
addShape()
@Override
public boolean addShape(int x, int y, char pattern, int... params)
This method is to add a shape into canvas, and the parameter means:
x and y: the location x and y of shape
pattern: the pattern of shape
params:
For circle: the length of params is 1, which represents the radius of circle
For RightTriangle: the length of params is 3.
The first one is the width of RightTriangle
The second one is the height of RightTriangle
LEFT_UP direction.
The third one is the index of Direction. For example, if the third value is 0, it
means
Return value: Has one non-space grid of shape in corresponding location of canvas is in the
bound, in this case it returns true, otherwise returns false.
In this class, we allow shapes to be stored overlappingly, and shapes added later can be placed

overlapping on top of those being added.
When placed overlappingly, the space grid in shape means empty and it cannot cover the original non-space grid.
Test code:

```
public static void main(String[] args) {
ShapeCanvas canvas1 = new OverLapShapeCanvas(15, 15);
canvas1.addShape(0, 0, 'A', 6);
canvas1.addShape(1, 1, 'B', 5);
canvas1.addShape(2, 2, 'C', 4);
canvas1.addShape(3, 3, 'D', 3);
canvas1.addShape(10, 5, 'E', 4, 6, 2);
canvas1.addShape(14, 14, 'F', 4, 6, 3);
canvas1.addShape(10, 5, '0', 3, 2, 1);
canvas1.addShape(10, 5, '1', 1, 1, 2);
for (char[] line : canvas1.getCanvas()) {
System.out.println(line);
}
System.out.println(canvas1.getShapeCount());
System.out.println(canvas1.getSpaceGridCount());
canvas1.getShapesByArea().forEach(System.out::println);
canvas1.getShapesByLocation().forEach(System.out::println);
}
```

Result:

```
AAAAAAAA
AABBBBBBAA
AABCCCCCCBAA
ABCDDDDDDCBA
ABCDDDDDDCBA
ABCDDDDDDCBA
ABCDDDDDDCBA
ABCDDDDDDCBA
ABCDDDDDDCBA
AABCCCCCCBAA
AABB10EEAA
AAA000EA
EEE
EE
EE
```

```
RightTriangle: (10,5) area=1 pattern=1
RightTriangle: (10,5) area=5 pattern=0
RightTriangle: (10,5) area=16 pattern=E
Circle: (3,3) area=36 pattern=D
Circle: (2,2) area=60 pattern=C
Circle: (1,1) area=88 pattern=B
Circle: (0,0) area=132 pattern=A
```

Circle: (0,0) area=132 pattern=A
Circle: (1,1) area=88 pattern=B
Circle: (2,2) area=60 pattern=C
Circle: (3,3) area=36 pattern=D
RightTriangle: (10,5) area=5 pattern=0
RightTriangle: (10,5) area=1 pattern=1
RightTriangle: (10,5) area=16 pattern=E