# Hands-on Experiment # 12: Worksheet

Section\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

No more than 3 students per one submission of this worksheet.

Student ID \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student ID \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student ID \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Part A: Getting Familiar with Problem (Do not code here)

In this lab, we aim to write a program to draw many geometric shapes (Square, RightTriangle, Triangle) using standards keyboard characters. In order to draw a figure, there are 2 input parameters: character and the number of rows. Assume *rows* is 5,

* For Square, the number of characters in each row and column must be 5.
* For RightTriangle and Triangle, the number of characters is increased by 1 every row (up to 5).

|  |  |  |
| --- | --- | --- |
| \*\*\*\*\* \*\*\*\*\* \*\*\*\*\* \*\*\*\*\* \*\*\*\*\* | % %%  %%%  %%%% %%%%% | #  # #  # # #  # # # #  # # # # # |
| Square | RightTriangle | Triangle |

Assume the size is 6 rows using a character ‘\*’, **draw** the following shapes and **compute** their perimeters and areas.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Square | RightTriangle | Triangle |
| Draw | \*\*\*\*\*\*  \*\*\*\*\*\*  \*\*\*\*\*\*  \*\*\*\*\*\*  \*\*\*\*\*\*  \*\*\*\*\*\* | \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\*  \*\*\*\*\*\* | \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\*  \*\*\*\*\*\* |
| Perimeter | Character = ‘\*’  Row = 6 | Character = ‘\*’  Row = 6 | Character = ‘\*’  Row = 6 |
| Area | 36 | 18 | 18 |

**Draw** the above RightTriangle when it is vertical flip and draw the above Trianlge when it is horizontal flip.

|  |  |  |
| --- | --- | --- |
|  | RightTriangle (Vertical Flip) | Triangle (Horizontal Flip) |
| Draw | \*\*\*\*\*\*  \*\*\*\*\*  \*\*\*\*  \*\*\*  \*\*  \* | \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\*  \*\*\*\*\*\* |

Assume we can draw each shape at a position (x, y), where x is an indent (the number of spaces) and y is the starting row. Please draw a rectangle at the position (5, 2) when *rows*=6 and *character*=’\*’. From this example, there are 5 indents (x) and the starting row is 2 (y).

|  |  |  |  |
| --- | --- | --- | --- |
|  | Square | RightTriangle | Triangle |
| Draw | \*\*\*\*\*\*  \*\*\*\*\*\*  \*\*\*\*\*\*  \*\*\*\*\*\*  \*\*\*\*\*\*  \*\*\*\*\*\* | \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\*  \*\*\*\*\*\* | \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\*  \*\*\*\*\*\* |

## Part B: Design Your Class (Do not code here)

The below figure shows a part of the program: Shape and Square. Shape is a superclass of any shapes and there are 2 *protected* variables (rows and character) – represented by the “#” symbol.



Class “Shape”

* There are two properties (variables): *rows* and *character*
* There are 2 constructors.
* There are getter & setter methods for all properties (variables).
* toString() shows all variables’ value; e.g., “rows=5 and character=\*”

Class “Square”

* There are 2 constructors.
* draw(): to draw a square without indent and starting row.
* draw(int x, int y): to draw a square with *x* indents and starting row at *y*.
* getArea() and getPerimeter() to compute area and perimeter of the object.
* toString() shows object’s information; e.g., “Square: rows=5 and character=\*”.

If the variables (*rows* and *character*) in Shape are *private*, can the following code inside Square still be able to compile? If not, why?

|  |
| --- |
| // Inside the Square class  **public** **void** test1(){  **int** side = rows;  } |

No, it cannot because the attribute(row) is private from other classes.

Write UML diagram of all shapes including: Shape, Square, Triangle, RightTriangle

* In Triangle, there is a variable called “isHorizontalFlip”. If it is true, the figure is horizontal flipped.
  + In order to get and set this variable, there are 2 extra methods: boolean isHorizontalFlip() and void setHorizontalFlip(boolean isHorizontalFlip).
* In RightTriangle, there is a variable called “isVerticalFlip”. If it is true, the figure is vertical flipped.
  + In order to get and set this variable, there are 2 extra methods: boolean isVerticalFlip() and void setVerticalFlip(boolean is VerticalFlip).

|  |
| --- |
| Shape |
| #rows : int  #character : char |
| +Shape()  +Shape(in x : int, in c : char)  +getRows() : int  +setRows(in x : int)  +getCharacter() : char  +setCharacter(in character : char)  +toString() : String |

|  |
| --- |
| Square |
|  |
| +Square(in rows : int)  +Square(in rows : int, in character : char)  +draw()  +draw(in x : int, in y : int)  +getArea() : int  +getPerimeter() : int  +toString() : String |

|  |
| --- |
| Triangle |
| #isHorizontalFlip : boolean |
| +Triangle ()  +Triangle(in x : int, in c : char)  +draw()  +draw(in x : int, in y : int)  +getArea() : double  +getPerimeter() : double  +isHorizontalFlip() : Boolean  +setHorizontalFlip(in b : boolean)  +toString() : String |

|  |
| --- |
| RightTriangle |
| #isVerticalFlip : boolean |
| +RightTriangle ()  +RightTriangle(in x : int, in c : char)  +draw()  +draw(in x : int, in y : int)  +getArea() : double  +getPerimeter() : double  +isVerticalFlip() : Boolean  +setVerticalFlip(in b : boolean)  +toString() : String |

## Part C: Coding

Implement all classes based on your design in Part B. What is the result of TestDraw.java (code below)?

|  |
| --- |
| **public** **class** TestDraw {  **public** **static** **void** main(String[] args) {  Triangle head = **new** Triangle(7, '\*');  Square tail = **new** Square(5, '\*');  head.draw();  tail.draw(5, 0);  }  } |

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \* \*

Modify TestDraw.java to draw the following figure.

|  |
| --- |
| #  # #  # # #  # # # #  # # # # #  # # # # # #  # # # # # # #  %%%%%%%%%%%%%%%  %%%%%%%%%%%%%%%  %%%%%%%%%%%%%%%  %%%%%%%%%%%%%%%  %%%%%%%%%%%%%%%  %%%%%%%%%%%%%%%  %%%%%%%%%%%%%%%  %%%%%%%%%%%%%%%  %%%%%%%%%%%%%%%  %%%%%%%%%%%%%%%  %%%%%%%%%%%%%%%  %%%%%%%%%%%%%%%  %%%%%%%%%%%%%%%  %%%%%%%%%%%%%%%  %%%%%%%%%%%%%%%  & &  && &&  &&& &&&  &&&& &&&&  &&&&& &&&&&  &&&&&& &&&&&&  &&&&&&& &&&&&&& |

Include the screenshots below.

List all your source code here.

public class Shape {

protected int rows;

protected char character;

public Shape() {

};

public Shape(int *x*, char *c*) {

rows = x;

character = c;

}

public int getRows() {

return rows;

}

public void setRows(int *x*) {

this.rows = x;

}

public char getCharacter() {

return character;

}

public void setCharacter(char *character*) {

this.character = character;

}

public String toString() {

return "rows:" + rows + " character:" + character;

}

}

public class Square extends Shape {

public Square(int *rows*) {

this.rows = rows;

this.character = '\*';

}

public Square(int *rows*, char *character*) {

this.rows = rows;

this.character = character;

}

public void draw() {

for (int i = 1; i <= rows; i++) {

for (int j = 1; j <= rows; j++)

System.out.print(character + " ");

System.out.println();

}

}

public void draw(int *x*, int *y*) {

x++; *// adjusting x position*

y++; *// adjusting y position*

for (int i = 1; i <= rows + y; i++) {

if (i >= y) { *// only prints character after y rows*

for (int j = 1; j <= rows + x; j++) {

if (j >= x) *// only prints character after x spaces*

System.out.print(character + " ");

else

System.out.print(" "); *// otherwise prints whitespace*

}

}

System.out.println(); *// moves to next line*

}

}

public int getArea() {

return rows \* rows;

}

public int getPerimeter() {

return 4 \* rows;

}

public String toString() {

return "Square: rows:" + rows + " character:" + character;

}

}

public class Triangle extends Shape {

private boolean isHorizontalFlip;

public Triangle(int *rows*) {

super.rows = rows;

super.character = '#';

isHorizontalFlip = false;

}

public Triangle(int *rows*, char *character*) {

super.rows = rows;

super.character = character;

isHorizontalFlip = false;

}

public void draw() {

if (isHorizontalFlip) {

for (int i = rows; i >= 1; i--) {

for (int k = 1; k <= rows - i; k++)

System.out.print(" ");

for (int j = 1; j <= i; j++) {

System.out.print(character + " ");

}

System.out.println();

}

} else {

for (int i = 1; i <= rows; i++) {

for (int k = 1; k <= rows - i; k++)

System.out.print(" ");

for (int j = 1; j <= i; j++) {

System.out.print(character + " ");

}

System.out.println();

}

}

}

public void draw(int *x*, int *y*) {

x++;

y++;

if (isHorizontalFlip) {

for (int i = rows + y - 1; i >= 1; i--) {

if (rows - i + 1 >= y) {

if (isHorizontalFlip()) {

for (int k = 1; k <= rows - i; k++)

System.out.print(" ");

}

for (int j = 1; j <= i + x - 1; j++) {

if (j >= x)

System.out.print(character + " ");

else

System.out.print(" ");

}

}

System.out.println();

}

} else {

for (int i = 1; i <= rows + y - 1; i++) {

if (i >= y) {

for (int k = 1; k <= rows - i; k++)

System.out.print(" ");

for (int j = 1; j <= i + x - 1; j++) {

if (j >= x)

System.out.print(character + " ");

else

System.out.print(" ");

}

}

System.out.println();

}

}

}

public double getArea() {

return 0.5 \* rows \* rows / Math.tan(60);

}

public double getPerimeter() {

return 3 \* rows / Math.sin(60);

}

public String toString() {

return "Square: rows:" + rows + " character:" + character + " isHorizontalFlip:" + isHorizontalFlip;

}

public boolean isHorizontalFlip() {

return isHorizontalFlip;

}

public void setHorizontalFlip(boolean *isHorizontalFlip*) {

this.isHorizontalFlip = isHorizontalFlip;

}

}

public class RightTriangle extends Shape {

private boolean isVerticalFlip;

public RightTriangle(int *rows*) {

super.rows = rows;

super.character = '%';

isVerticalFlip = false;

}

public RightTriangle(int *rows*, char *character*) {

super.rows = rows;

super.character = character;

isVerticalFlip = false;

}

public void draw() {

for (int i = 1; i <= rows; i++) {

if (isVerticalFlip) { *// when vertical flipped print some whitespaces first to push the triangle to*

*// the right*

for (int k = 1; k <= rows - i; k++)

System.out.print(" ");

}

for (int j = 1; j <= i; j++) {

System.out.print(character + " ");

}

System.out.println();

}

}

public void draw(int *x*, int *y*) {

x++;

y++;

for (int i = 1; i <= rows + y; i++) {

if (i >= y) {

if (isVerticalFlip) {

for (int k = 1; k <= rows - i; k++)

System.out.print(" ");

}

for (int j = 1; j <= i + x; j++) {

if (j >= x)

System.out.print(character + " ");

else

System.out.print(" ");

}

}

System.out.println();

}

}

public void drawDoubleTriangle(int *space*) { *// for the two triangle at the last part*

for (int i = 1; i <= rows; i++) {

*// Left triangle*

for (int k = 1; k <= rows - i; k++)

System.out.print(" ");

for (int j = 1; j <= i; j++) {

System.out.print(character + " ");

}

for (int index = 1; index <= space; index++)

System.out.print(" ");

*// Right triangle*

for (int j = 1; j <= i; j++) {

System.out.print(character + " ");

}

System.out.println();

}

}

public double getArea() {

return 0.5 \* rows \* rows;

}

public double getPerimeter() {

return 2.0 \* rows + Math.sqrt(2.0 \* rows \* rows);

}

public String toString() {

return "RightTriangle: rows:" + rows + " character:" + character + " isVerticalFlip:" + isVerticalFlip;

}

public boolean isVerticalFlip() {

return isVerticalFlip;

}

public void setVerticalFlip(boolean *isVerticalFlip*) {

this.isVerticalFlip = isVerticalFlip;

}

}

Submit this worksheet (by only one member of the group) via <http://www.myCourseVille.com> (Assignments > Hands-on Experiment # 12) before noon of the day after your lecture.