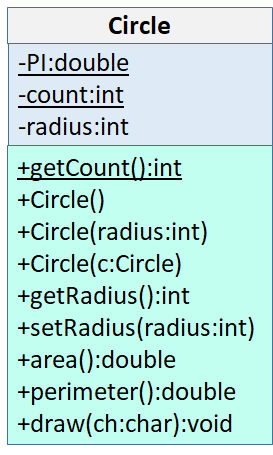
Lab 13

**Rules:**

* You are asked to implement **Circle.java** as described below.
* Do not forget to take your work with you when you leave the lab by either copying your work files to your own USB flash disk, or by e-mailing them to yourself.

You are asked to implement the “**Circle**” class as follows:



As you can see, the **Circle** class has two **static attributes** shown underlined in the UML class diagram:

1. private static final double **PI** = 3.14159; // This is a final static variable storing the value of PI
2. private static int **count** = 0; // You will use this variable to keep track of the number of Circle objects created so far

The **Circle** class also has a private attribute “**radius**” that stores the radius of the circle.

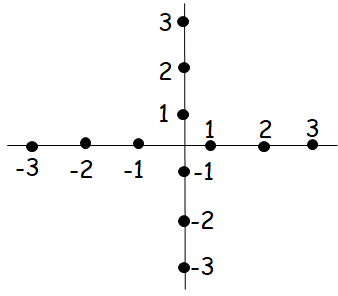
The **Circle** class has 3 constructors:

1. A no-arguments constructor that must initialize radius to 1.
2. A parametrized constructor that takes in the user-supplied radius and initializes the circle with it. In case the user passes a radius < 1, you must throw an IllegalArgumentException.
3. A copy constructor that takes in a Circle object, and initializes the Circle’s radius to that of the passed Circle.

Make sure that you use **constructor chaining** to implement these constructors. Also make sure that you keep track of the total number of Circle objects created so far using the static “**count**” member variable.

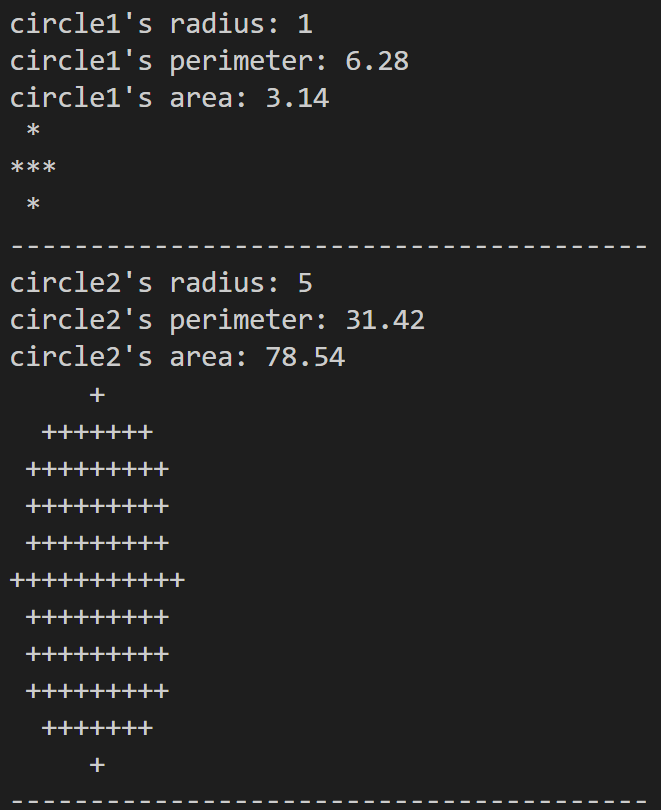
**Circle** class has 1 public static method, and 5 public methods:

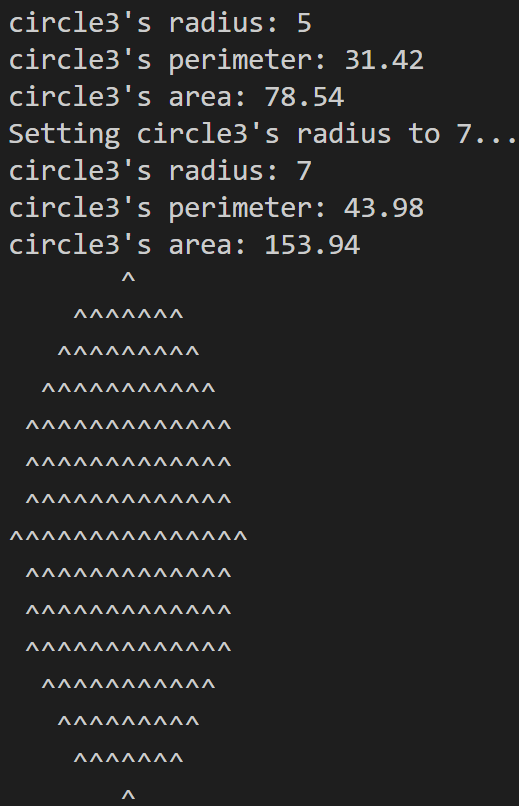
1. **getCount()**, which is the static method that returns the number of Circle objects created so far.
2. **getRadius()**, which is the getter for radius.
3. **setRadius()**, which is the setter for radius. Make sure that you validate the radius. If the user-supplied radius is less than 1, then you must throw an IllegalArgumentException(), by the following statement: “throw new IllegalArgumentException()”.
4. **area()**, which returns the area of the circle computed as PI\*radius\*radius.
5. **perimeter()**, which returns the perimeter of the circle computed as 2\*PI\*radius.
6. **draw()**, which draws the circle on the screen using the given character. To draw the circle, assume that you are drawing a square from radius >= y >= -radius && -radius <= x <= radius as shown below for a radius of 3:

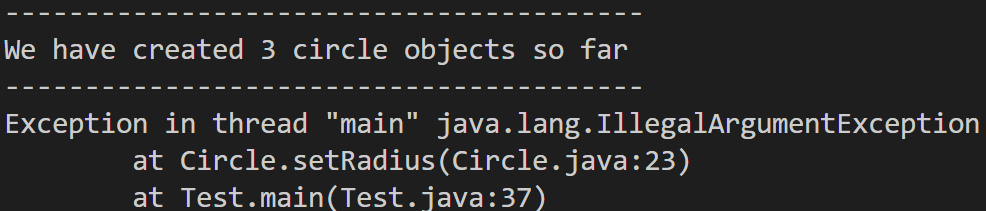


You write an outer loop where the y coordinate goes from +radius to –radius, and an inner loop where x coordinate goes from –radius to +radius. You now have a square. To see if the point (x, y) is inside the circle, do the following comparison: x\*x + y\*y <= radius\*radius. If this comparison is true, you are inside the circle and put the drawing char on the screen. Otherwise, put a space char on the screen.

Test your class, we are giving you a driver code (**Test.java**) that creates 3 circles using different constructors and then uses them to test if things work correctly. Here is the expected output of your code when run against **Test.java**:







You are advised to implement your own test code. When grading, we may use a different Test. Make sure that your code works under all circumstances.

Lab Work Submission:

* You can continue to work on this lab after our lab class, on your own, at home.
* Submit your lab work via Blackboard on or before: **Wednesday, October 18, 2023, 11:59pm**.
* The only accepted submission method!
* Once you submit your assignment you will not be able to resubmit it!
* Make absolutely sure the Java files you want to submit are the Java files you want graded.
* You will not be able to submit your lab work under any circumstances once **Lab13** disappears at **12:00 a.m.** on **Thursday, October 19, 2023**.
* There will be **NO** exceptions to these rules!
* To submit your lab work, upload **Circle.java** (**with .java extension**) you did for this lab to the **Lab13** assignment in the **Labs** tab in your Lab section’s presence in Blackboard.
* Then, make sure you click the **Submit** button to submit your lab work.
* This lab is worth **5 points**.