Small Assignment #2
Due: Wednesday, 2/5/2025 by 11:59 PM

Submission: You need to submit in two places.

(1) Put the following items into a folder, zip it up, and submit it on D2L.
- A PDF with your written answers to these questions.
- The code – including the provided files (with your alterations) and any new files (i.e. test code).

(2) Submit the PDF on Gradescope as well to help speed up the grading process.

Instructions.
These questions refer to the code provided in the zipped folder. You are welcome to open that code in an IDE and work with it there. You may also use online resources to help with these questions, but you must cite any sources you use, and you must answer the questions in your own words.

Question 1.
The `RectangleTest` class has a couple of `junit` tests for the `Rectangle` class. What is the purpose of the `@Test` annotation at the beginning of each of these tests?

Question 2.
The `RectangleTest` class has a couple of `junit` tests for the `Rectangle` class. Look at the `testPerimeter` method, and identify the following:
- the unit under test (UUT)
- the input
- the oracle

Question 3.
Is it reasonable to do exhaustive testing for the methods in the `Rectangle` class? Explain why or why not.

Question 4.
This question gets into aspects of inheritance, so it will be graded for completion, but it will not hurt you to start thinking about these issues. Feel free to look up answers to this particular question online if necessary, but write your answers in your own words.

(a) What is the purpose of the `@Override` annotation at the beginning of the equals method in the `Rectangle` class? Does the code work the same way if it is removed?
(b) If you leave everything in the equals method the same except that you change the type of the parameter to `Rectangle`, what happens? Why?
(c) What does it mean to override a method, and what is required for it to be done correctly?

Question 5.
If you run the junit tests, you'll see that you do not get very good coverage with the testEquals method. Calculate the following in terms of how much of the equals method is actually covered.

(a) statement coverage
(b) branch coverage
(c) path coverage

Question 6.
Note that in both the Rectangle and the Circle classes, the equals method is comparing double values with ==. This is not necessarily the best approach for comparing floating point values for equality. (In fact, some programming languages won't even let you do it.) Explain why this is a potential problem and explain how you can compare two doubles for equality in a more reliable way.

Question 7.
Write `junit` tests for the following classes: `Rectangle.java, Circle.java, Shape.java`
Your tests should follow the guidelines given in class for writing good tests, and they must achieve at least 95% coverage using a branch coverage metric.

Question 8.
You can probably get the required test coverage above by writing a single test for the `translate` method in `Shape.java`. But notice that the definition of "position" is different for Rectangle and Circle, which affects what "translate" means for each of those classes. This is a case, where it makes sense to write additional tests to make sure that the Rectangle and Circle objects can be translated correctly. Add additional tests for that.