CSC110 Assignment 2:
for-loops and console input

Objectives

Upon completion of this assignment, you need to be able to:

- Update and modify existing Java code.
- Use a `for-loop` for repeated operations.
- Do a few simple operations with the Java `String` datatype.
- Continue accumulating good programming skills.

Introduction

In this assignment, we add a little more functionality to the first assignment. There is now a personal `user` message to an ASCII drawing. We also tighten up our approximation to π to resemble a more accurate approximation to use to calculate the area of a circle.

Quick Start

(1) Download this pdf file and store it on your computer.
(2) Create a directory / folder on your computer specific to this assignment. for example, `CSC110/assn2` is a good name.
(3) Save a file called `SecondProgram.java` in your directory. Fill the source code with the class name and the following methods:
   - `public static void main(String[] args)`
   - `public static void printMessage()`
   - `public static void printOwl()`
   - `public static void areaCircle()`

   Make the methods empty by adding a { and a } with only blank space between them.

(4) Copy and paste `printOwl` from Assignment One into the `SecondProgram.printOwl` method. Copy and paste `FirstProgram.approxPi` into the `SecondProgram.areaCircle` method. You will be making some minor updates to these methods.
(5) Compile and run the program frequently, preferably after each of the detailed instructions below.

Detailed Instructions

Part I : Updating the Ascii article

The new version of your owl will include a speech bubble that contains a bit of wisdom. This wise owl will say exactly what it is told to say by *input from the user*, entered on the console. In the example below, the owl is not very wise, but that is because the word `hoot` was what the user typed in when prompted.

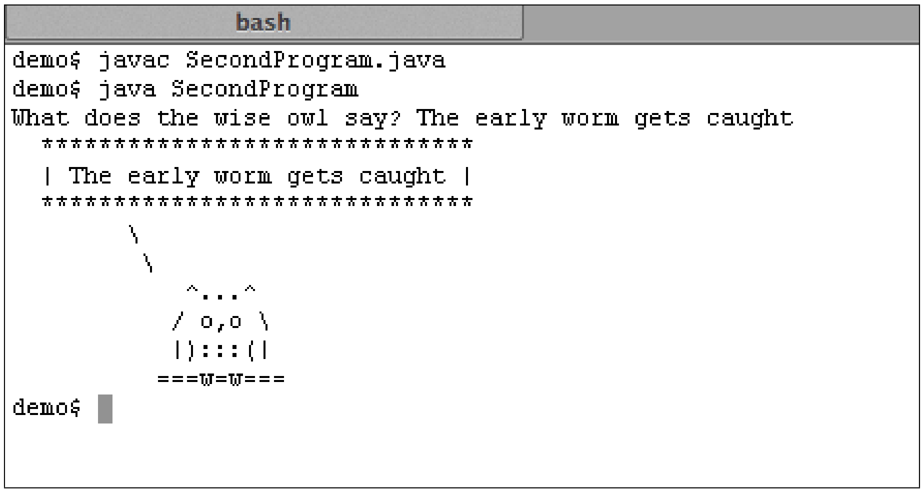
```
*********
| Hoot! |
*********
     \
      \
         ^...^
        / o,o \
        |):::(|
       ===w=w===
```

The `printOwl` method must call the `printMessage` method; this method constructs the speech bubble and the backslashes in the drawing shown above. Inside the `printMessage` method, do the following:

- Print a message to the console, instructing the user to type a wise message for an owl to say.
- Collect that message into an appropriately named `String` variable.
- Declare and initialize a single `Scanner` object. Use its `nextLine()` method to capture the user's input string from the console.
- Based on the length of the message, print out the line of stars for the top and bottom border of the text box, using a `for-loop`. Print the message with a single straight line character on either side, leaving a space between the first and final characters of the message.
- Add the call to `printMessage` inside the `printOwl` method. Note that this is the only modification necessary to the Assignment One version of `printOwl`.

- Test `printMessage` inside the `main` method; once you see it works fine, then you can test `printOwl` to see the complete drawing.

The following screenshot shows an example of the compilation of the source code, the running of the program and the output on the console. All tests except the one call to `printOwl` are commented out. Note the image includes the actual input as typed in by the user.



```
demo$ javac SecondProgram.java
demo$ java SecondProgram
What does the wise owl say? The early worm gets caught
  ******************************
  | The early worm gets caught |
  ******************************
        \
         \
            ^...^
           / o,o \
           |):::(|
          ===w=w===
demo$ 
```

Part II : Calculating the area of a circle

Recall the series used to approximate the value of π from the previous assignment: In many programming libraries, this number is approximated with the following calculation:

$$\pi = 4 \times (1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + 1/13 - 1/15 + \ldots)$$

We saw in the last assignment that 8 terms yields a value of approximately 3.017, which is not very accurate. In actual fact, the convergence of this harmonic series is pretty slow. See the table below to the exponential variations in the number of iterations with minuscule results:

iterations	value
10	3.0418396189294032
100	3.1315929035585537
1000	3.140592653839794
10,000	3.1414926535900345
100,000	3.1415826535897198
1,000,000	3.1415916535897743
10,000,000	3.1415925535897915
100,000,000	3.141592643589326
1,00,000,000	3.1415926525880504

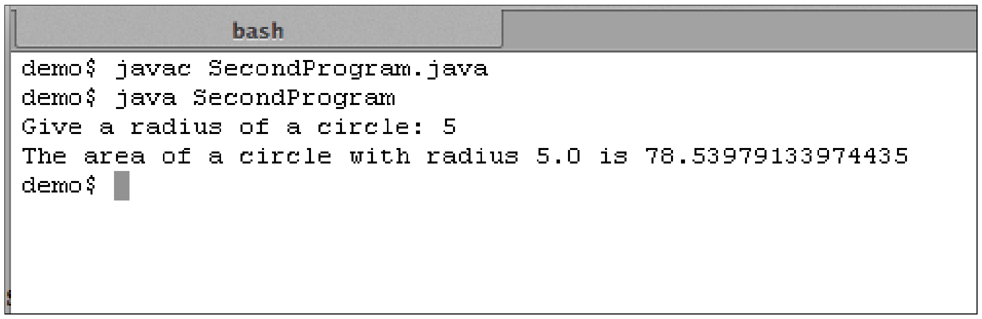
The actual value for π is $3.141592653589\ldots$, but for this assignment, we are happy with 3.14159 as an approximation. This happens after one million iterations.

Inside `areaCircle` make the following modifications to the code that was copied from `approxPi`:

- Instead of calculating the first 8 terms of the sequence, use the first $1,000,000$ terms. You must use a `for-loop` to do this.
- Remembering that the formula for a circle is πr^2, where r is the radius of the circle, ask the user to provide the radius. The code for this action is similar to the `printMessage` method, where you created a `Scanner` object and asked the user for the value. In this case, the value we want is a number, a `double` in fact, so instead of using the `Scanner`'s `nextLine()` method, use `nextDouble()` instead.
- Calculate the area of the circle where the radius is the user's input and π is the value produced by the million iterations of the series. [*]
  Print this value to the console.
- Test the `areaCircle` method by adding a call in the `main` method.

The following screenshot shows an example of the compilation of the source code, the running of the program and the output. In this case, the call to `printMessage` and `printOwl` was commented out in `main`.

[*]In your exploration of the Java Libraries, you may have come across a more accurate approximation for π in the `Math` class. However, the purpose of this exercise includes *using* methods that we create, so resist using `Math.PI` in this exercise.



```
demo$ javac SecondProgram.java
demo$ java SecondProgram
Give a radius of a circle: 5
The area of a circle with radius 5.0 is 78.53979133974435
demo$
```

Submission

Submit the following completed file to the Assignment folder on conneX.

- `SecondProgram.java`

Please make sure you have submitted the required file(s) and conneX has sent you a confirmation email. Do not send `[.class]` (the byte code) files. Also, make sure you *submit* your assignment, not just save a draft. Draft copies are not made available to the instructors, so they are not collected with the other submissions. We *can* find your draft submission, but only if we know that it's there.

A note about academic integrity

It is OK to talk about your assignment with your classmates, and you are encouraged to design solutions together, but each student must implement their own solution.

Grading

Marks are allocated for . . .

- No errors during compilation of the source code.
- Output produced exactly as requested in this document.
- Methods are written as specified.
- Methods are all tested in `main`; they may be commented out but not removed.
- Style of the source code meets the requirements outlined in the coding convention document available in the Resources folder of conneX.