* **Title Page: cs 1011-051 – Lab 4 PiEstimate**
* Cody Mott
* 10/04/2022

**I. Objectives**

This program asks the user to input a radius, then an image of a semicircle with that radius is generated in the command console, and the value of PI is estimated using the value of the area and the radius of that semicircle. The user is then asked for an error tolerance and the program estimates the value of pi to within that error tolerance.

**II. Requirements**

* The program must be named PiEstimate
* The program shouldn’t crash if the user enters a valid integer for the radius. If the user doesn’t enter an integer, the program should report the problem and ask the user again to enter the radius
* the program shouldn’t crash if the user enters a double or an int for the error tolerance. If the user enters anything other than a double or an int, an error message should display and they should again be asked for the error tolerance

**III. Design:**

START main

call getRadius

call generateImage

call estimatePi

call getErrorTolerance

call estimatePiWithErrorTolerance

END main

START getRadius

try  
 ask user for radius  
 get radius

catch  
 tell user to enter an integer

repeat if user didn’t enter integer

END getRadius

START generateImage

for current row, radius +1 to 0

if first row or last row

print row “+-----+”

else

print semicircle

calculate area of semicircle

END generateImage

START estimatePi

estimate pi using area and radius

print estimation to console

END estimatePi

START getErrorTolerance

do

ask user for error tolerance

get error tolerance

if user enter invalid input

tell user to enter a double

while input is invalid

END getErrorTolerance

START estimatePiWithErrorTolerance

initialize radius as 1

initialize area as 0

do

for rows

for columns

add one to area

}

}

estimate pi

add 1 to radius

reset area value

}while pi estimate is not within tolerance

print estimation of pi and radius of semicircle used

END estimatePiWithErrorTolerance

**III. Test Plan/Test Strategy**

For each method, I tested the code as I went. So, whenever I would write code that did something, I would run the code and see if it did what it was supposed to. Whenever I wrote code asking for user input, I ran the code to make sure it printed to the console. Whenever I got user input, I would print the input back to the console to make sure it matched what was entered. Whenever I ran into an instance where code wasn’t working properly, I would change one or two things and rerun the code and I would repeat this process, trying different stuff until I got the code to work. Whenever I had calculations or variables that changed, I would print those to the console to see if they were behaving how they should.

**IV. Results**

Text

Description automatically generated

**V. Discussion**

I learned that to not get stuck in an infinite loop when handling exceptions in a loop, you need to clear the buffer with nextLine().

I had a weird issue when estimating the value of PI using the error tolerance. In my original code I had piEstimation = 4 \* area/radius\*radius. The odd thing was that on ever single loop iteration, though radius was increasing, radius was getting treated as equal to 1 in that equation. I even verified that radius was infact increasing every loop iteration. To fix it I just squared the radius in a variable radiusSquared and inserted that new variable into the piEstimation equation.