**Project 1 Test Plan**

**Program Goals and Objectives**

The purpose of this program is to calculate the area of a regular polygon with user provided side length and number of sides using a given area formula.

**Program Functional Requirements**

1. The user needs to be prompted for number of sides and side lengths to the polygon.
2. The program has to calculate the area of the polygon using the given formula area = n \* s ^ 2 / (4 \* tan(pi / n)).
3. The program has to output “the area of the polygon is “ + area.

**Test Matrix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case | Input / Output | Expected Result | Actual Result | Outcome (pass/fail) |
| 1 | 5, 6.5 | The area of the polygon is 72.69017017488385 | The area of the polygon is 72.69017017488385 | pass |
| 2 | 3, 10 | The area of the polygon is 43.301270189221945 | The area of the polygon is 43.301270189221945 | pass |
| 3 | Three, 10 | java.util.InputMismatchException | java.util.InputMismatchException | fail |
| 4 | -5, 6.5 | The area of the polygon is 72.69017017488385 | The area of the polygon is 72.69017017488385 | fail |
| 5 | 2, 6.5 | The area of the polygon is 1.2935331815993918E-15 | The area of the polygon is 1.2935331815993918E-15 | fail |

**Pseudocode**

Function Main

Declare Int sides

Declare Double length

Declare Double area

Output "Enter the number of sides"

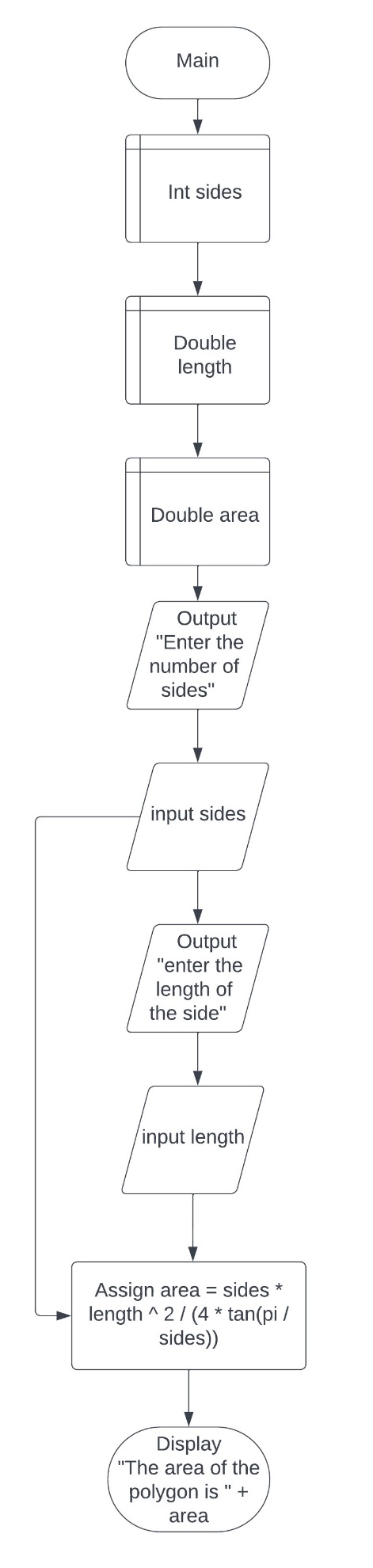
Input sides

Output "Enter the length of the side"

Input length

Assign area = sides \* length ^ 2 / (4 \* tan(pi / sides))

Output "The area of the polygon is " + area



**Project 2 Test Plan**

**Program Goals and Objectives**

The purpose of this program is to return whether a social security number input by the user is valid social security number formatting (###-##-####).

**Program Functional Requirements**

1. The user needs to be prompted for a social security number in the format (###-##-####).
2. The program must check that the input social security number is the correct length.
3. The program must check that the characters within the number are of the digit type.
4. The program must check that there are 2 dashes (-) between the 3 numbers within the social security number.
5. The program must output whether the social security number is valid or invalid.

**Test Matrix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case | Input / Output | Expected Result | Actual Result | Outcome (pass/fail) |
| 1 | 1 | 1 is an invalid social security number | 1 is an invalid social security number | pass |
| 2 | 123-45-6789000 | 123-45-6789000 is an invalid social security number | 123-45-6789000 is an invalid social security number | pass |
| 3 | 123/45/6789 | 123/45/6789 is an invalid social security number | 123/45/6789 is an invalid social security number | pass |
| 4 | Abc-45-6789 | Abc-45-6789 is an invalid social security number | Abc-45-6789 is an invalid social security number | pass |
| 5 | 123-45-6789 | 123-45-6789 is a valid social security number | 123-45-6789 is a valid social security number | pass |

**Pseudocode**

Function Main

Declare String ssn

Declare Boolean ssnStart

Declare Boolean ssnMid

Declare Boolean ssnEnd

Declare Boolean ssnDashes

Output "Enter a SSN"

Input ssn

If ssn.length() = 11

ssnStart = Character.isDigit(ssn.charAt(0)) && Character.isDigit(ssn.charAt(1)) &&

Character.isDigit(ssn.charAt(2))

ssnMid = Character.isDigit(ssn.charAt(4)) && Character.isDigit(ssn.charAt(5))

ssnEnd = Character.isDigit(ssn.charAt(7)) && Character.isDigit(ssn.charAt(8)) &&

Character.isDigit(ssn.charAt(9)) && Character.isDigit(ssn.charAt(10))

ssnDashes = ssn.charAt(3) == '-' && ssn.charAt(6) == '-'

If (ssnStart && ssnMid && ssnEnd & ssnDashes)

Output ssn + " is a valid social security number"

False:

Output ssn + " is an invalid social security number"

False:

Output ssn + " is an invalid social security number"

Diagram

Description automatically generated