UNIVERSITÉ Concordia UNIVERSITY

Engineering

&

Computer Science

SOEN-6011

Aruna Devi Pala, Student ID - 40184469

July 29, 2022

Scientific Calculator - Power Function ab^x

supervised by

Professor - Pankaj Kamthan

Contents

1 Problem 1

1.1 Description

A power function is of the form:

$$f(x) = ab^x \tag{1}$$

where x is a real number, a and b are constants.

1.2 Domain

The domain is set of all real numbers, (- ∞,∞)

1.3 Co-domain

The co-domain is also set of all real numbers.

1.4 Characteristics of Power Function.

1. For any exponential function, the domain is the set of all real number, however range is bounded by the horizontal symptote of the graph of f(x).
2. The behaviour of power function effects the exponential growth and decay.
3. When b greater than 1, the graph accelerates towards y-axis contributing to exponential growth.
4. When b greater than 1 and less than 0, the graph decreases towards y-axis contributing to exponential decay.
5. When modeling real-world situations with an exponential function, the domain and range can be limited to numbers that make sense in the context. The domain and range can be stated using the inequalities for a continuous interval in these cases.

1.5 Context of use model

Developers:
The developers of the system use it frequently to check if the system is working as expected.

Testers:
The testers of the system test the functions are working correctly.

Assessing Team:
The assessor team will need to assess the project implementation.

Students and Peers:
Students and peers of the course may use the calculator to find the results of the value of ab^x. It can serve as a future reference.

Researchers and Other Academicians:
Researchers and other academicians can be interested in the system to calculate the result of the function ab^x. Identify differences in implementation, the behaviour of the system to compare and contrast.

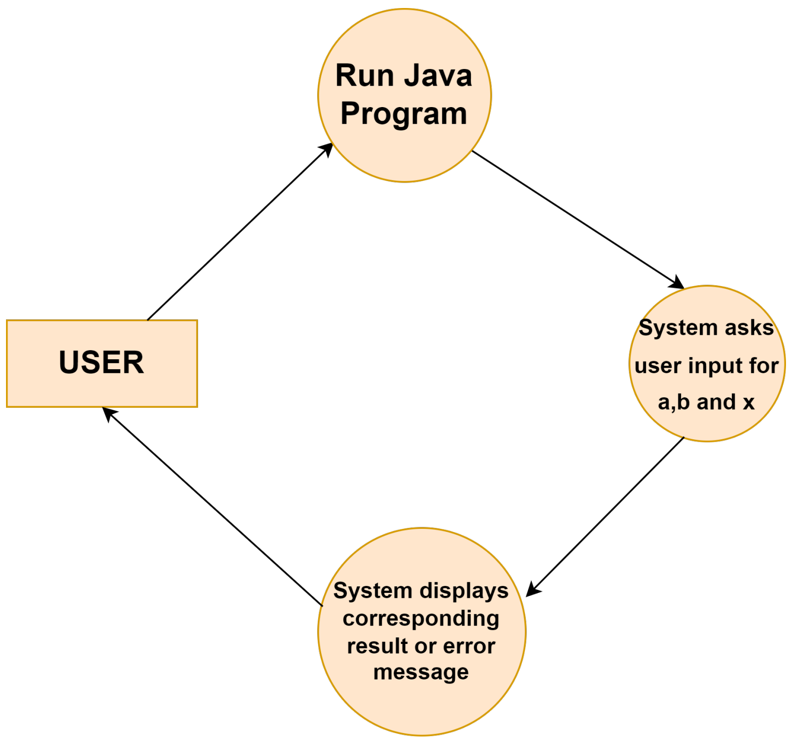

Run Java Program
System asks user input for a,b and x
System displays corresponding result or error message
USER

Figure 1: Context of Use model: Power Function ab^x

The key elements of use model are:

Boxes:
External entities that the main entity interacts with (organizations, departments, systems, or processes)

Connectors:
Interactions between entities

Flows:
Interactions are labelled to show the flow of data, physical objects between the entities.

2 Problem 2

2.1 Assumptions

- In function $f(x) = ab^x$, the output will always be in decimals.
- The output of the function will be greater than zero, if a and b constants are not equal to zero.
- The range for the values of x can be from -100000 $\leq$ x $\leq$ +100000.
- x can be negative but not in decimal.
- The value for a and b ranging from -100000 $\leq$ a $\leq$ 100000 and -100000 $\leq$ b $\leq$ 100000.
- a and b are constants, the calculator accepts the magical constants such as e.

2.2 Requirements

1. **First Requirement**
   - **ID =** FR1
   - **Type =** Functional Requirements
   - **Version =** 1.0
   - **Difficulty =** Easy
   - **Priority =** 1
   - **Description =** System shall take an input x as a real number.
   - **Rationale =** The rationale behind this requirement is to calculate the function only for real numbers.
2. **Second Requirements**
   - **ID =** FR2
   - **Type =** Functional Requirements
   - **Version =** 1.0
   - **Difficulty =** Easy
   - **Priority =** 1
   - **Description =** System should validate the input.
   - **Rationale =** The rationale behind this requirement is to check domain value for function.
3. **Third Requirement**
   - **ID =** FR3
   - **Type =** Functional Requirements

- **Version =** 1.0
- **Difficulty =** Easy
- **Priority =** 3
- **Description =** System shall output the value within the expected range.
- **Rationale =** The rationale behind this requirement is to get result in the range of the function.

4. **Fourth Requirement**
   - **ID =** FR4
   - **Type =** Functional Requirements
   - **Version =** 1.0
   - **Difficulty =** Easy
   - **Priority =** 1
   - **Description =** System shall show relevant error messages if any.
   - **Rationale =** The rationale behind this requirement is to handle error handling.
5. **Fifth Requirement**
   - **ID =** FR5
   - **Type =** Functional Requirements
   - **Version =** 1.0
   - **Difficulty =** Medium
   - **Priority =** 1
   - **Description =** System shall accepts magical constant e as constant.
   - **Rationale =** The rationale behind this requirement is the acceptability of the constants.
6. **Sixth Requirement**
   - **ID =** FR6
   - **Type =** Non-Functional Requirements
   - **Version =** 1.0
   - **Difficulty =** Easy
   - **Priority =** 1
   - **Description =** System shall show relevant success messages or confirmation results.

- **Rationale =** The rationale behind this requirement is to show relevant messages and contributes to usability.

7. **Seventh Requirement**
    - **ID =** FR7
    - **Type =** Non-Functional Requirements
    - **Version =** 1.0
    - **Difficulty =** Easy
    - **Priority =** 2
    - **Description =** System shall complete the calculation on expected time.
    - **Rationale =** The rationale behind this requirement is to have good performance of the calculator.

3 Problem 3

3.1 Algorithms Used and Pseudocode

Following are the algorithms used for function ab^x

1. Recursive approach

2. Iterative approach

3. Divide and Conquer approach

3.2 Algorithm Description

3.2.1 Algorithm 1 - Recursive approach

The details of algorithm 1 is given below:
Complexity = $O(logn), O(n)$
Space Complexity = $O(1)$
Approach = Recursion
Rationale = a and b are the constants, depending on the value of b the complexities is being calculated such that if b = any integer then powerfunction() will be called. So, the time complexity will be $O(logn)$ and space complexity to 1. If b = e then taylor series will execute calling exponential() resulting $O(n)$ time complexity and $O(1)$ space complexity.
Advantages

1. $O(logn)$ is the time complexity of the recursive algorithm. This recursive technique has tail recursive to avoid stack overflow issues and handle large inputs more efficiently.
2. Algorithm 1 has more readability and maintainable code which is easy to handle.

Disadvantages

1. It uses more memory because of the function calling itself and add to stack in each recursion and it stores the value until recursion completes.
2. Sometimes recursion can be slow if the function is not implemented properly. Analyzing recursion code is complex to solve a difficult problem to simpler way to solve.

3.2.2 Algorithm 2: Iterative approach

The details of algorithm 2 is given below:
Complexity = $O(n), \Omega(n)$
Space Complexity = $O(1)$
Approach = Iterative
Rationale = The a and b are the constants, depending on the value of b the complexities is being calculated such that if b = any integer then calling powerfunction(). So, the time complexity will be $O(n)$ and space complexity to 1. If b = e then taylor series will execute calling exponential() resulting $\Omega(n)$ time

Algorithm 1 Recursive Approach - ab^x

```
procedure $function5(a, b, x)$
    in: String a, b, x
    out: double result
    result = 0
    if ((a || b) == "0") then return result
    else
        if (b == "e") then
            result = a * exponential(x)
        else
            result = a * powerfunction(b,x)
        return result;

procedure $exponential(n)$
    in: int n
    out: double result
    sum = 1
    x = 10
    for i <= $x - 1$
    $sum = 1 + n * sum/i$
    return sum

procedure $powerfunction(x, n)$
    in: double x, int n
    out: double result
    if (($n < 0$) then
        return $1.0/powerHandler(x, Math.abs(n))$
    else
        return $powerHandler(x, n)$

procedure $powerHandler(x, n)$
    in: double x, int n
    out: double result
    if ($n == 0$) then return 1
    if ($n == 1$) then return x
    if ($nmod2 == 0$) then
        return $powerHandler(x * x, n/2)$
    else
        return $x * powerHandler(x * x, n/2)$
```

Algorithm 2 Iterative Approach - ab^x

```
procedure $function5(a, b, x)$
    in: String a, b, x
    out: double result
    result = 0
    if ((a || b) == "0") then return result
    else
        if (b == "e") then
            sum = 1
            m = 1 and i = 1
            for i <= $x - 1$
            $sum = 1 + m * sum/i$
            end
            return a*sum
        else
            if ($x == 0$) then return 1
            if ($x == 1$) then return a*x
            else
                $i = 1andpow = 1$
                for i <= n
                $pow = pow * x$
                end
                return a*sum
```

complexity and $O(1)$ space complexity.

Advantages

1. Because all operations are performed on the heap, iterative algorithms which avoid stack overflow.
2. Space complexity is $O(1)$.

Disadvantages

1. The complexity is $O(n)$.
2. The algorithm is not efficient.
3. The algorithm has poor readability and maintainability.

3.2.3 Algorithm 3 : Divide and Conquer

The details of algorithm 3 is given below:
Complexity = $O(n), \Omega(n)$
Space Complexity = $O(1)$
Approach = Divide and Conquer
Rationale = a and b are the constants, depending on the value of b the complexities is being calculated such that if b = any integer then powerfunction() will

Algorithm 3 Divide and Conquer Approach - ab^x

```
procedure $Function5(a, b, x)$
    in: String a, b, x
    out: double result
    result = 0
    if ((a || b) == "0") then return result
    else
        if (b == "e") then
            sum = 1
            m = 1 and i = 1
            for i <= $x - 1$
            $sum = 1 + m * sum/i$
            end
            return a*sum
        else
            if ($x$ == 0) then return 1
            if ($x$ == 1) then return a*x
            else
                return a*powerHandler(b,x)

procedure $powerHandler(b, x)$
    in: double b, int x
    out: double result
    if (x == 1) then
        return 1
    else if (x mod 2 == 0) then
        return $powerHandler(b, x/2) * powerHandler(b, x/2)$
    else
        return $b * powerHandler(b, x/2) * powerHandler(b, x/2)$
```

be called. So, the time complexity will be $O(n)$ and space complexity to 1. If b = e then taylor series will execute calling exponential() resulting $\Omega(n)$ time complexity and $O(1)$ space complexity.
Advantages

1. The problem has been solved in divide and conquer approach may result in robustness.
2. Space complexity is $O(1)$.

Disadvantages

1. The complexity is $O(n)$.
2. The constant allocation of memory space, which leads to a stack overflow, has a substantial impact on efficiency.

3.3 Conclusion

By computing power(b, x/2) only once and storing it, the algorithm 2, algorithm 3 can be optimized to $O(logn)$. The approach used in algorithm1 stores power at once. Also algorithm1 avoids stack overflow while calculating the input which is one of the advantage for choosing algorithm 1 over other two.

4 Problem 4

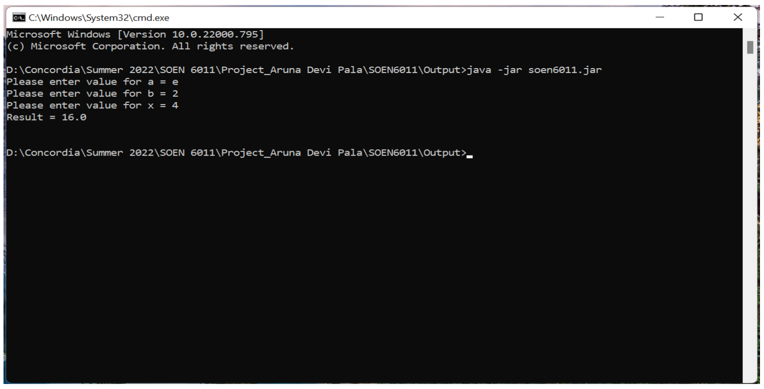
4.1 Standard Guidelines

The source code review is based on the standard practices that a developer has used while writing the code. The coverage of all standard guidelines on which the report has generated is listed below.

1. The code should eliminate nested if statement.
2. The code should follow a coding style guide.
3. The code should have meaningful method and variable names.
4. The author should annotate the comments not exceeding 10 to 20 lines.
5. The class program should not import unnecessary packages.
6. The code should have low coupling and high cohesion.
7. The code should avoid the global scoping for common variables and functions.
8. The code should avoid memory leakage.
9. The code should narrow the variable scope as far as possible in the code.
10. Usage of mutable variable over creating new variables in object oriented language.

4.2 Mind-Map for Programming style

Standard programming style used		
S.No.	Description	Result
1	No Error found in the code	Pass
2	No deadlock encountered	Pass
3	Use of annotation in the code	Pass
4	No usage of nested ifs statement	Pass
5	No Warning found in the code	Pass
6	No unnecessary package is included in the code	Pass
7	Usage of meaningful method and variable names	Pass
8	Not exceeding source line of code	Pass
9	No unused variable found	Pass
10	No memory leakage	Pass

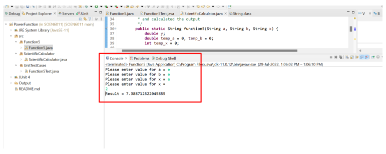


```
Microsoft Windows [Version 10.0.22000.795]
(c) Microsoft Corporation. All rights reserved.

D:\Concordia\Summer 2022\SOEN 6011\Project_Aruna Devi Pala\SOEN6011\Output>java -jar soen6011.jar
Please enter value for a = e
Please enter value for b = 2
Please enter value for x = 4
Result = 16.0


D:\Concordia\Summer 2022\SOEN 6011\Project_Aruna Devi Pala\SOEN6011\Output>_
```

Figure 2: Snapshot of Running Program using Command Prompt

Program submitted doesn't depend on any particular tool or IDE to compile. You can run and compile the program using **.jar** file using *java -jar soen6011.jar* as shown below:

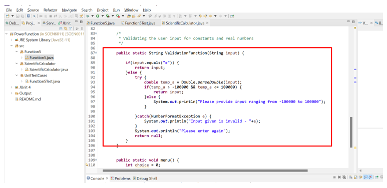
1. **Textual Interface** is being used in the program by which user gets to know if input given is valid/Invalid
2. After system gets input from the user it calculates the output and display to the user for the function given which is ab^x 3. Below figure shows the exact Interface how it looks like while running program 4. Used exception handling for whole function.Figure shows the error handling



```
     * and calculated the output
     */
    public static String function5(String a, String b, String x) {
        double y;
        double temp_a = 0, temp_b = 0;
        int temp_x = 0;
```

```
<terminated> Function5 [Java Application] C:\Program Files\Java\jdk-11.0.12\bin\javaw.exe (29-Jul-2022, 1:06:02 PM – 1:06:10 PM)
Please enter value for a = e
Please enter value for b = e
Please enter value for x = e
Please enter value for x = 
2
Result = 7.388712522045855
```

(a) Textual Inteface



```
    /*
     * Validating the user input for constants and real numbers
     */

    public static String ValidationFunction(String input) {

        if(input.equals("e")) {
            return input;
        }else {
            try {
                double temp_a = Double.parseDouble(input);
                if(temp_a > -100000 && temp_a <= 100000) {
                    return input;
                }else {
                    System.out.println("Please provide input ranging from -100000 to 100000");
                }

            }catch(NumberFormatException e) {
                System.out.println("Input given is invalid - "+e);
            }
            System.out.println("Please enter again");
            return null;
        }
    }


    public static void menu() {
        int choice = 0;
```

(b) Error Handling

Figure 3: Snapshot of Interface Error-handling

4.3 Eclipse Debugger

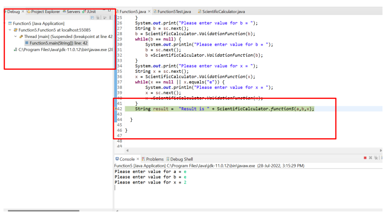
Debugging is the process of finding defects and fixing faults, flaws, and anomalies in software. It's an important tool for any Java developer because it aims in the detection of minor bugs that aren't obvious during code reviews or only occur when a specific condition is met.

Debugger Used : Inbuilt debugger offered by Eclipse Java Development Tools (JDT).

Following are the advantages and disadvantages of using debugger in the program.

Advantages

- Debugger promptly reports an erroneous state. This allows for earlier fault detection and makes the software development process stress-free and trouble-free.

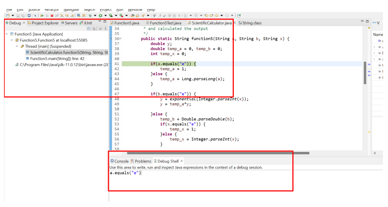


```
Function5 [Java Application]
  Function5.Function5 at localhost:55085
    Thread [main] (Suspended (breakpoint at line 42
      Function5.main(String[]) line: 42
  C:\Program Files\Java\jdk-11.0.12\bin\javaw.exe (28
```

```java
    }
    System.out.print("Please enter value for b = ");
    String b = sc.next();
    b = ScientificCalculator.ValidationFunction(b);
    while(b == null) {
        System.out.println("Please enter value for b = ");
        b = sc.next();
        b =ScientificCalculator.ValidationFunction(b);
    }
    System.out.print("Please enter value for x = ");
    String x = sc.next();
    x = ScientificCalculator.ValidationFunction(x);
    while(x == null || x.equals("e")) {
        System.out.println("Please enter value for x = ");
        x = sc.next();
        x =ScientificCalculator.ValidationFunction(x);
    }
    String result =  "Result is " + ScientificCalculator.function5(a,b,x);

  }

}
```

```
Function5 [Java Application] C:\Program Files\Java\jdk-11.0.12\bin\javaw.exe  (28-Jul-2022, 3:15:29 PM)
Please enter value for a = e
Please enter value for b = e
Please enter value for x = 2
```

(a) Debugger on Function5



```
Function5 [Java Application]
  Function5.Function5 at localhost:55085
    Thread [main] (Suspended)
      ScientificCalculator.function5(String, String, St
      Function5.main(String[]) line: 42
  C:\Program Files\Java\jdk-11.0.12\bin\javaw.exe (28
```

```java
     * and calculated the output
     */
    public static String function5(String a, String b, String x) {
        double y;
        double temp_a = 0, temp_b = 0;
        int temp_x = 0;

        if(a.equals("e")) {
            temp_a = 1;
        }else {
            temp_a = Long.parseLong(a);
        }

        if(b.equals("e")) {
            y = exponential(Integer.parseInt(x));
            y = temp_a*y;

        }else {
            temp_b = Double.parseDouble(b);
            if(x.equals("e")) {
                temp_x = 1;
            }else {
                temp_x = Integer.parseInt(x);
            }
```

```
Use this area to write, run and inspect Java expressions in the context of a debug session.
a.equals("e")
```

(b) Debugger on ScientificCalculator

Figure 4: Eclipse Built-in Debugger

- Using inbuilt debugger developer can step into code/out of/over code based on whether or not you think it matters.
- Debugging aids the developer in eliminating redundant and unwanted data.
- Debugging allows developers to avoid writing complex one-time testing code, which reduces resources and time during software development.

Disadvantages

- When the execution is halted inside an invariant, the debugger isn't particularly useful.
- If you use any of the previously unsupported expressions in a breakpoint condition, the condition will always return True since the evaluation is failing. In this instance, the debugger will come to a halt.

4.4 Achieved Quality Attributes

4.4.1 Robustness

- Usage of exception for exceptional test cases.
- Narrowed the variable scope as far as possible in the code.
- Error handling done on the code.
- Usage of mutable variable over creating new variables.

4.4.2 Maintainability

- Common practices has been adapted within the implemented function5 to avoid ambiguities.
- Useful comments added in the code where it is required.
- Avoided global scoping for common variables and functions.
- Refactored code wherever possible and merged to the github branch.

4.4.3 Usable

- Simple console interface provided for user input.

- Error messages are given in wrong input.
- Success messages are given for results.
- Suggestions are given when user encountered any difficulties while using to calculate ab^x.

4.4.4 Correctness

- Coding standards is followed.
- Proper testing practices has been done on the function assigned.
- JUnit Testing has implemented to check the correctness of eacj function the program by adding negative and positive testcases.

4.4.5 Efficiency

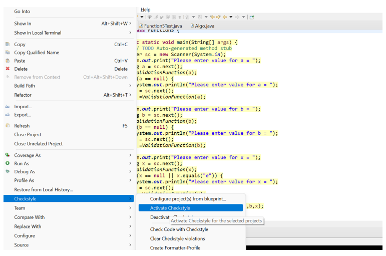
- The main focus on readability is given to the code.
- The program takes not less than two or three nanoseconds.

4.4.6 Portability

- The main focus is used to run the program on different platforms.Java Program is Platform independent hence have achieved portable.
- Program submitted is unlike most other programming languages, where a different version of a program must be made for each variety of computer, and an executable program that runs on one type of computer will not run on another.

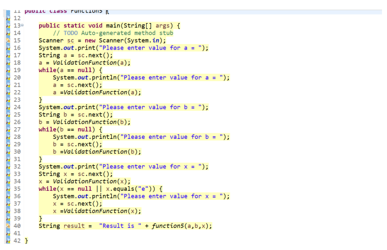
4.5 Quality Check of Source Code

It's a programming tool that helps programmers write Java code that follows a set of rules. It automates the process of inspecting Java code, saving humans the time and effort of doing so. It's ideal for projects that want to enforce a coding standard.



```
c static void main(String[] args) {
/ TODO Auto-generated method stub
er sc = new Scanner(System.in);
m.out.print("Please enter value for a = ");
g a = sc.next();
alidationFunction(a);
(a == null) {
ystem.out.println("Please enter value for a = ");
 = sc.next();
 =ValidationFunction(a);

m.out.print("Please enter value for b = ");
g b = sc.next();
alidationFunction(b);
(b == null) {
ystem.out.println("Please enter value for b = ");
 = sc.next();
 =ValidationFunction(b);

m.out.print("Please enter value for x = ");
g x = sc.next();
alidationFunction(x);
(x == null || x.equals("e")) {
ystem.out.println("Please enter value for x = ");
 = sc.next();
```

(a) Activating checkstyle



```
public class Function5 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
    Scanner sc = new Scanner(System.in);
    System.out.print("Please enter value for a = ");
    String a = sc.next();
    a = ValidationFunction(a);
    while(a == null) {
        System.out.println("Please enter value for a = ");
        a = sc.next();
        a =ValidationFunction(a);
    }
    System.out.print("Please enter value for b = ");
    String b = sc.next();
    b = ValidationFunction(b);
    while(b == null) {
        System.out.println("Please enter value for b = ");
        b = sc.next();
        b =ValidationFunction(b);
    }
    System.out.print("Please enter value for x = ");
    String x = sc.next();
    x = ValidationFunction(x);
    while(x == null || x.equals("e")) {
        System.out.println("Please enter value for x = ");
        x = sc.next();
        x =ValidationFunction(x);
    }
    String result =  "Result is " + function5(a,b,x);

}
```

(b) checkstyle on code

Figure 5: Snapshot of Checkstyle

Advantages

- The checkstyle is portable between different IDEs.

- Easily integrate as a pre-commit hook or into your build tool into your Software Configuration Management.
- Checkstyle is a stand-alone framework, integrating it with your other tools is considerably easier.
- ability of creating user own rules. Eclipse defines a large set of styles, but checkstyle has more, and you can add your own custom rules.

Disadvantages

- Checkstyle is a static analysis tool for a single file.
- Checkstyle cannot properly handle a file's binary Byte Order Mark (BOM), like UTF8's Unicode.

4.6 Review Approach

Used Approach - Lightweight Code Review

- Walk-through the code.
- Executed the code in run time environment and also used debugger.
- Inspection on comment length, Checking dependencies among the components, Check on unused variables.
- Observing global scoping of the functions and variables.

4.7 Conclusion

Overall, the code is well-structured and well-written. It is clear from the code that readability is a developer's primary focus. The only suggestion I have is to retain logs because they can help with debugging.

5 Problem 5

5.1 Implemented Unit Test

- Unit Testing Framework Used: JUnit
- The sample practice used in the test cases are as follows:

  ID = TC1
  Test Case = description about the test case.
  Function Used = name of the function used in the test case.
  Functional Requirement ID Used = FRn

5.2 Results

No.	Description	Result
1	Test Cases covers of all possible cases	PASS
2	Test Cases covers the exception handling	PASS
3	Requirement mentioned in the problem2 has covered	PASS
4	Working of calculating sin() function	PASS
5	Working of calculating cos() function	PASS
6	Working of calculating tan() function	PASS
7	Test on checking input within range	PASS
8	Test cases follows assumptions	Not Applicable

5.3 Standard Guidelines

The following are the guidelines implemented in the unit testing for function 5 :

$$f(x) = ab^x \tag{2}$$

- Each test case should given a test case id.
- Each test should contains the unit test of each function involved in the implementation of function 5.
- Each test case is followed by one line statement justifying it's description.
- Each test case contains the function name used in the unit test case.
- Test case shall cover all the cases expected.
- Test Case should mention a expected value and actual value.
- Atleast one test case should be consider into a delta precision values.
- Each test case should show message or log for the sake of readability.
- Each test case should be traceable.
- Test case should map to the functional requirement.

5.4 Conclusion

Almost all the methods of function5 have been tested and confirmed to be effective and corrective. During the testing process, no major mistakes were discovered. All of the requirements given in Problem 2 has been covered in the mentioned test cases. Furthermore, all the test cases are to the point and covered all possibilities.

5.5 Suggestions

Assigning each test case a unique id would make contribution to readability.

References

[1] Wikiedia: Exponent Function,
https://en.wikipedia.org/wiki/Exponential_function
[2] Geeksforgeeks: Power Function,
https://www.geeksforgeeks.org/write-a-c-program-to-calculate-powxn/
[3] Understanding context: Context use model,
https://users.encs.concordia.ca/ kamthan/courses/soen-6011/understanding$_context.pdf$
[4]$Recursivealgorithm : SpaceandTimeComplexities$,
https://www.enjoyalgorithms.com/blog/time-complexity-analysis-of-recursion-in-progr