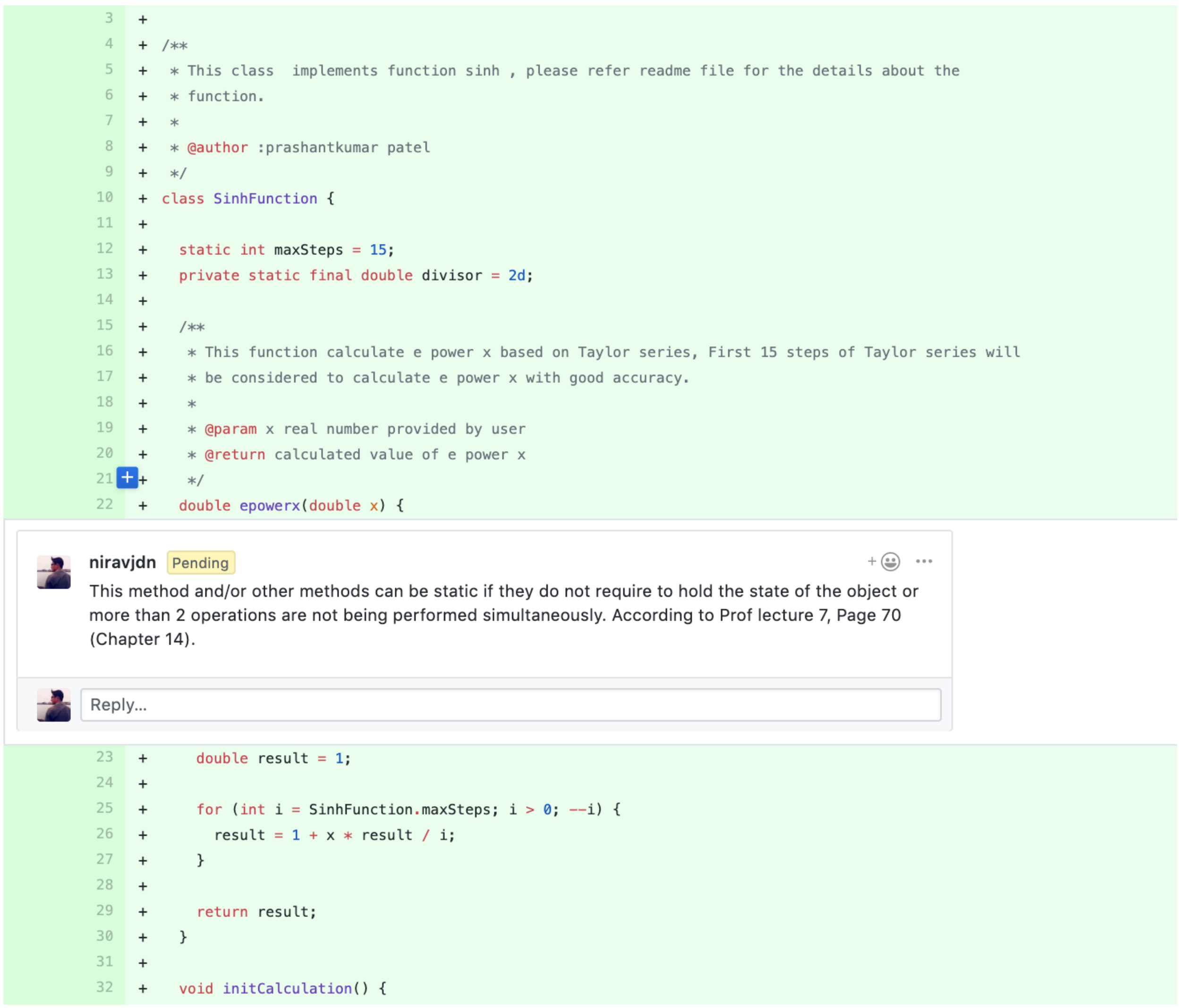
Code Review of F3

Code Review Done By: Nirav Patel (40081268)
Function Developed By: Prashant Patel

F3 : Sinh(x)

Manual Code Review

1. For the method defined in SinhFunction.java, `double epowerx(double x)`
   This method and/or other methods can be static if they do not require to hold the state of the object or more than 2 operations are not being performed simultaneously. According to Prof lecture 7, Page 70 (Chapter 14).



```
+
+ /**
+  * This class  implements function sinh , please refer readme file for the details about the
+  * function.
+  *
+  * @author :prashantkumar patel
+  */
+ class SinhFunction {
+
+   static int maxSteps = 15;
+   private static final double divisor = 2d;
+
+   /**
+    * This function calculate e power x based on Taylor series, First 15 steps of Taylor series will
+    * be considered to calculate e power x with good accuracy.
+    *
+    * @param x real number provided by user
+    * @return calculated value of e power x
+    */
+   double epowerx(double x) {
```

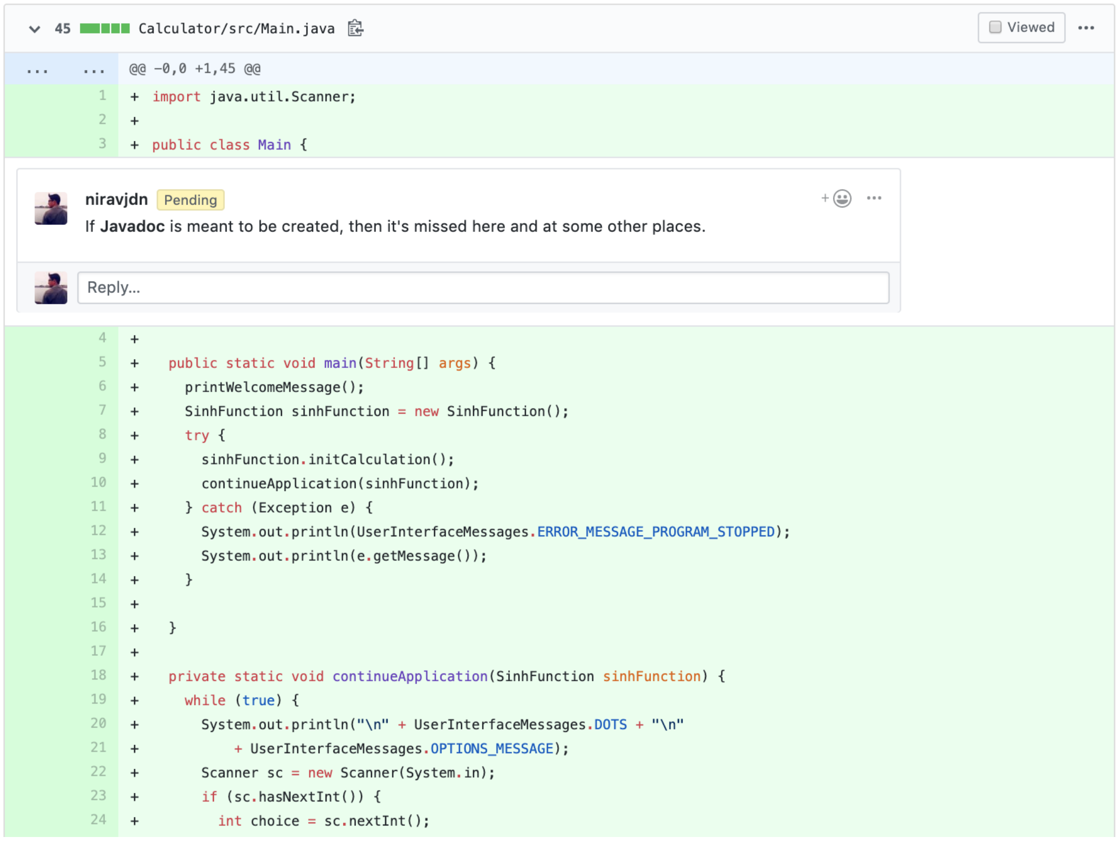
niravjdn Pending

This method and/or other methods can be static if they do not require to hold the state of the object or more than 2 operations are not being performed simultaneously. According to Prof lecture 7, Page 70 (Chapter 14).

Reply...

```
+     double result = 1;
+
+     for (int i = SinhFunction.maxSteps; i > 0; --i) {
+       result = 1 + x * result / i;
+     }
+
+     return result;
+   }
+
+   void initCalculation() {
```

2. If Javadoc is meant to be created, then it's missed here and at some other places.



45 Calculator/src/Main.java

Viewed

```
@@ -0,0 +1,45 @@
+ import java.util.Scanner;
+
+ public class Main {
```

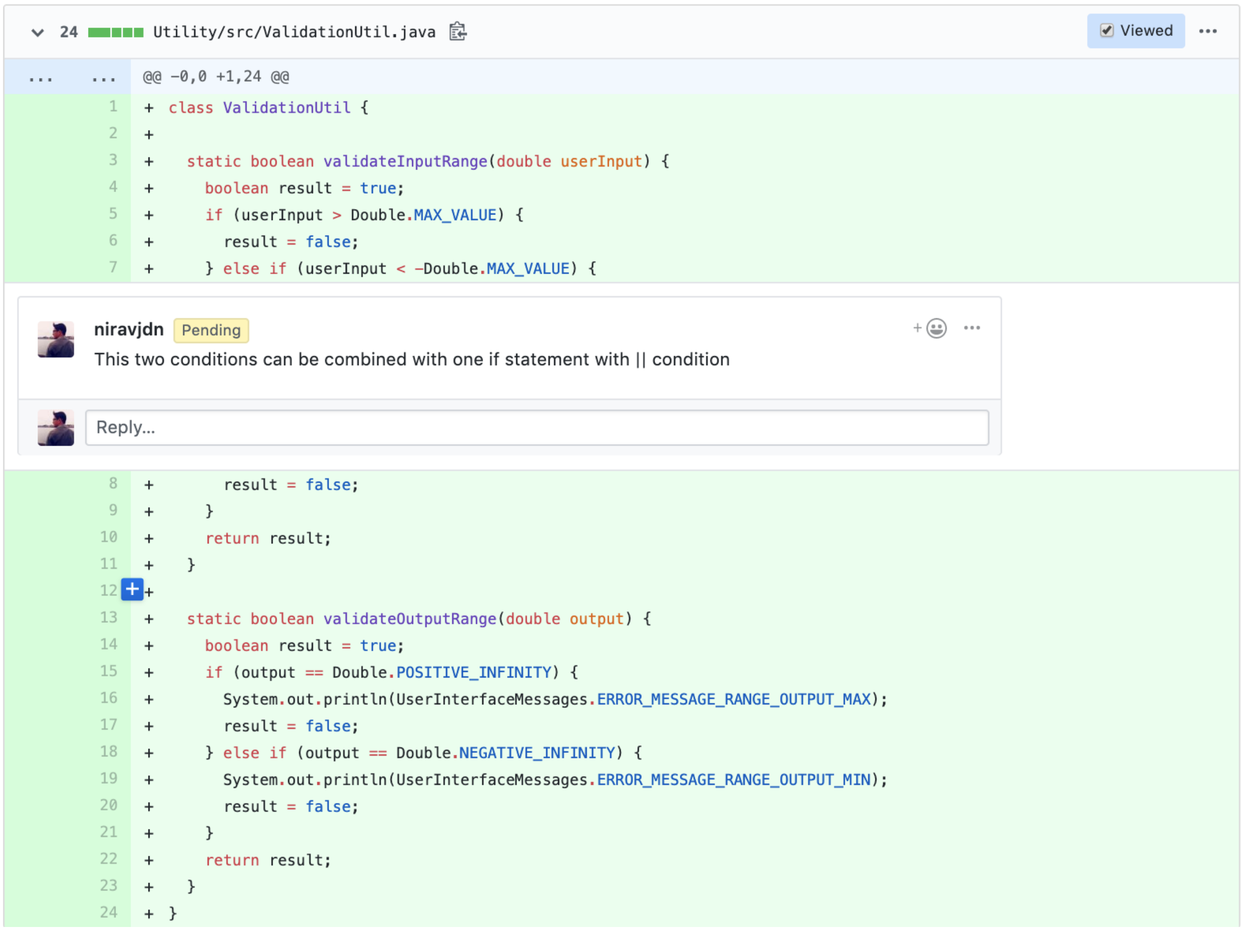
niravjdn Pending

If **Javadoc** is meant to be created, then it's missed here and at some other places.

Reply...

```
+
+   public static void main(String[] args) {
+     printWelcomeMessage();
+     SinhFunction sinhFunction = new SinhFunction();
+     try {
+       sinhFunction.initCalculation();
+       continueApplication(sinhFunction);
+     } catch (Exception e) {
+       System.out.println(UserInterfaceMessages.ERROR_MESSAGE_PROGRAM_STOPPED);
+       System.out.println(e.getMessage());
+     }
+
+   }
+
+   private static void continueApplication(SinhFunction sinhFunction) {
+     while (true) {
+       System.out.println("\n" + UserInterfaceMessages.DOTS + "\n"
+           + UserInterfaceMessages.OPTIONS_MESSAGE);
+       Scanner sc = new Scanner(System.in);
+       if (sc.hasNextInt()) {
+         int choice = sc.nextInt();
```

3. For the validatonUtils.java, it validates whether the input and output is in valid range or not, it checks for the different condition and prints the same message, which can be combined in single if condition by making use of logical OR (||) condition.



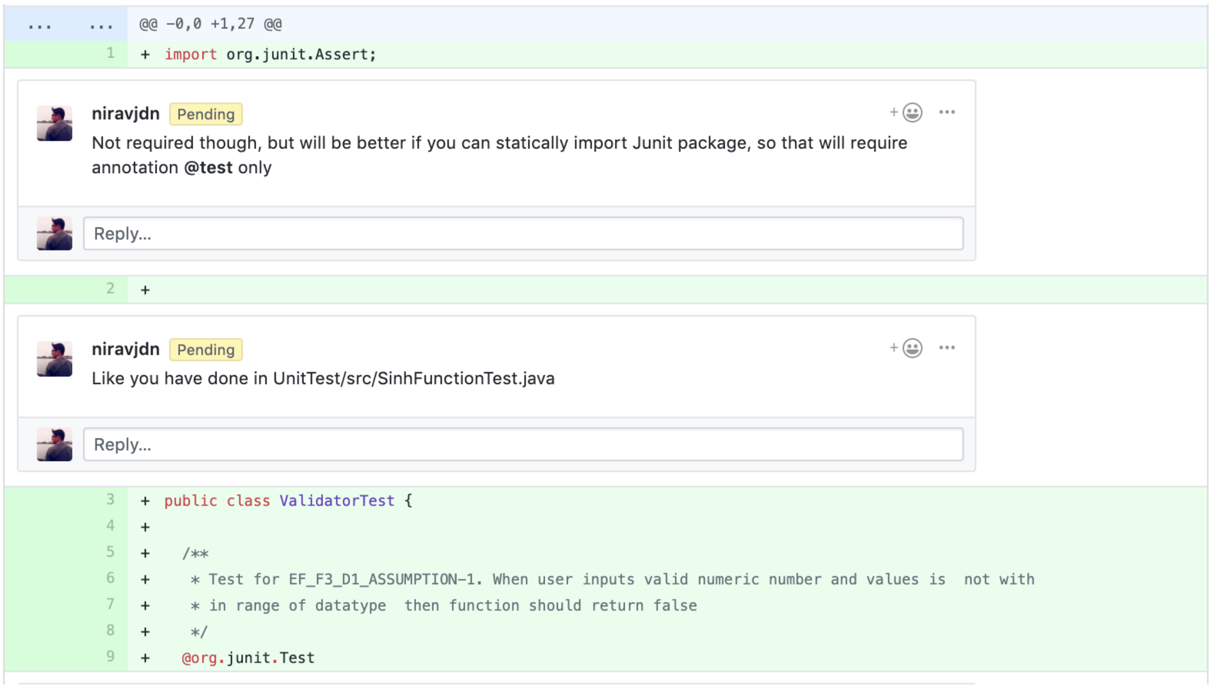
```
@@ -0,0 +1,24 @@
+ class ValidationUtil {
+
+   static boolean validateInputRange(double userInput) {
+     boolean result = true;
+     if (userInput > Double.MAX_VALUE) {
+       result = false;
+     } else if (userInput < -Double.MAX_VALUE) {
```

niravjdn Pending
This two conditions can be combined with one if statement with || condition

Reply...

```
+       result = false;
+     }
+     return result;
+   }
+
+   static boolean validateOutputRange(double output) {
+     boolean result = true;
+     if (output == Double.POSITIVE_INFINITY) {
+       System.out.println(UserInterfaceMessages.ERROR_MESSAGE_RANGE_OUTPUT_MAX);
+       result = false;
+     } else if (output == Double.NEGATIVE_INFINITY) {
+       System.out.println(UserInterfaceMessages.ERROR_MESSAGE_RANGE_OUTPUT_MIN);
+       result = false;
+     }
+     return result;
+   }
+ }
```

4. For the test case, proper import can help in not writing the respective package name while using class belonging to that package name. So instead of @org Junit.Test, it can be written as @Test only.



```
...   ...  @@ -0,0 +1,27 @@
+ import org.junit.Assert;
```

niravjdn Pending
Not required though, but will be better if you can statically import Junit package, so that will require annotation **@test** only

Reply...

```
+
```

niravjdn Pending
Like you have done in UnitTest/src/SinhFunctionTest.java

Reply...

```
+ public class ValidatorTest {
+
+   /**
+    * Test for EF_F3_D1_ASSUMPTION-1. When user inputs valid numeric number and values is  not with
+    * in range of datatype  then function should return false
+    */
+   @org.junit.Test
```

5. Not necessary, but as a standard practice of Junit test cases, Assert class from junit package can be imported as static. So it requires just the use of assertEquals instead of Assert.assertEquals().

Automatic Code Review

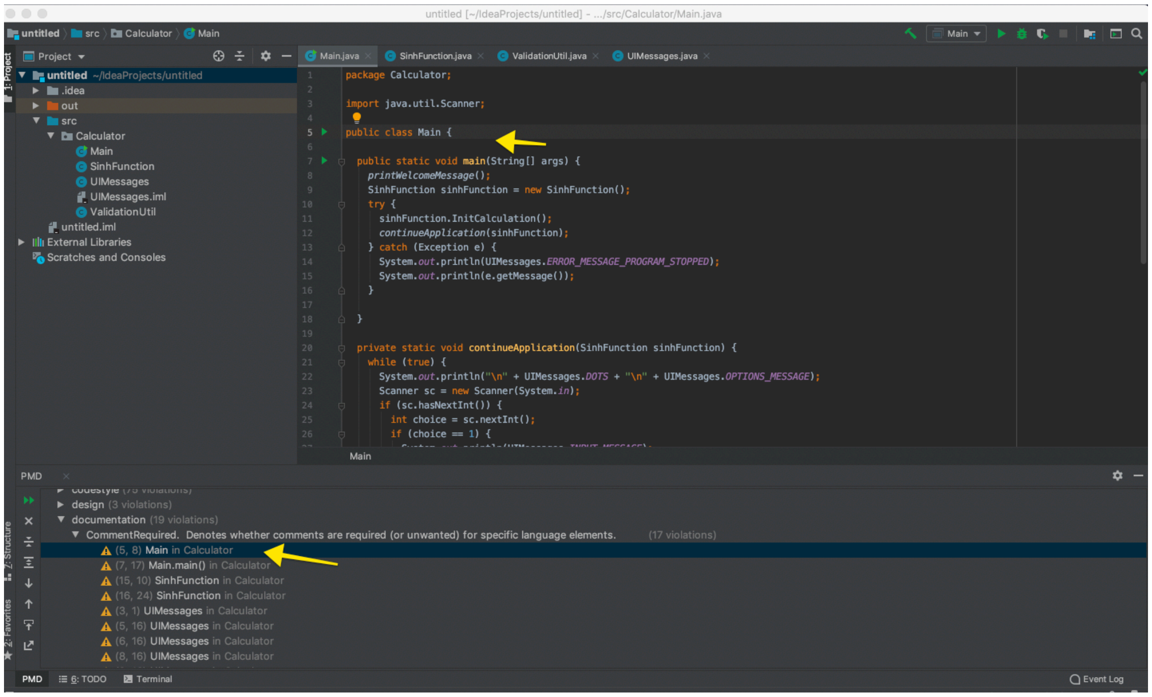
For the automatic code review, the Intellij plugin PMD was used as a tool. As developer had developed project in IntelliJ, I had to use IntelliJ in order to mimic the same development environment.

The Installation and usage process of PMD is simpler, It needs to be installed from IntelliJ Market place, and then run it on target project.

It analyses the entire source code for number of different constrains such as Best Practices, Code Style, Design, Documentation, Error Prone Code, Performance and many other.

The important possible and unique improvements shown by the tool are listed below.

• It showed that at some methods, comments are missing. Comments are crucial part in program, which helps maintainer to modify the code whenever there is necessity. Refer the figure to have a idea, how the plugin shows the suggestions.

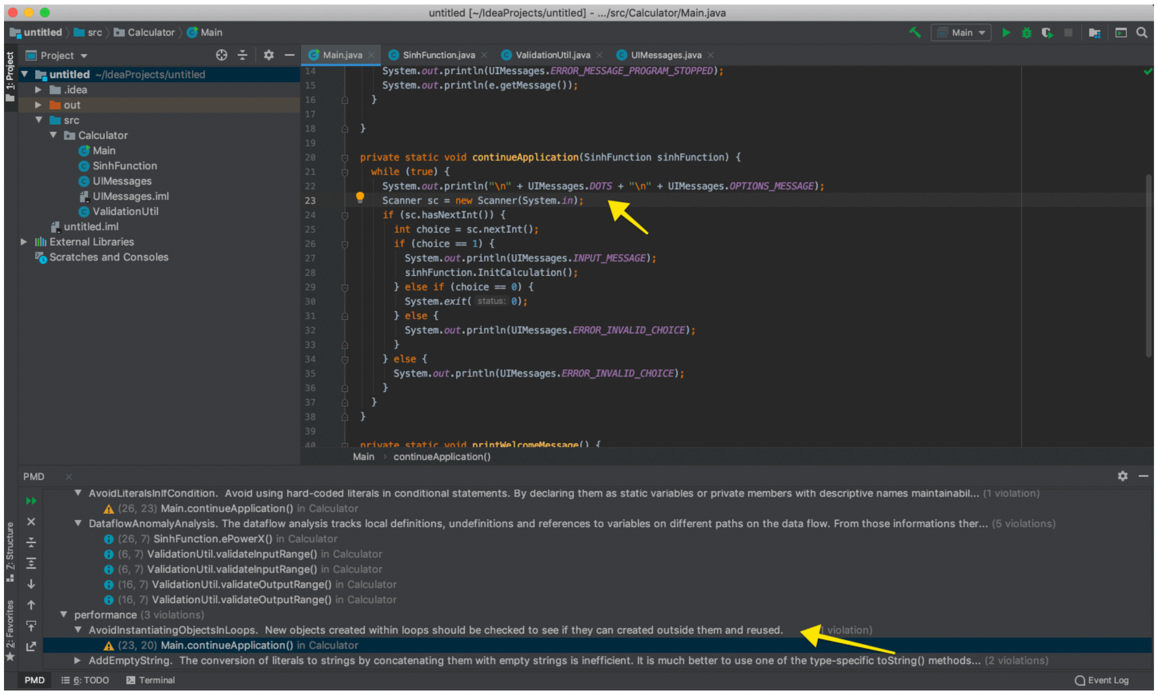


```
package Calculator;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        printWelcomeMessage();
        SinhFunction sinhFunction = new SinhFunction();
        try {
            sinhFunction.InitCalculation();
            continueApplication(sinhFunction);
        } catch (Exception e) {
            System.out.println(UIMessages.ERROR_MESSAGE_PROGRAM_STOPPED);
            System.out.println(e.getMessage());
        }

    }

    private static void continueApplication(SinhFunction sinhFunction) {
        while (true) {
            System.out.println("\n" + UIMessages.DOTS + "\n" + UIMessages.OPTIONS_MESSAGE);
            Scanner sc = new Scanner(System.in);
            if (sc.hasNextInt()) {
                int choice = sc.nextInt();
                if (choice == 1) {
```

```
design (3 violations)
documentation (19 violations)
  CommentRequired.  Denotes whether comments are required (or unwanted) for specific language elements.   (17 violations)
    (5, 8) Main in Calculator
    (7, 17) Main.main() in Calculator
    (15, 10) SinhFunction in Calculator
    (16, 24) SinhFunction in Calculator
    (3, 1) UIMessages in Calculator
    (5, 16) UIMessages in Calculator
    (6, 16) UIMessages in Calculator
    (8, 16) UIMessages in Calculator
```

Figure 1.1: Comment missing at some places

• It showed that at developer should not created instances inside the loop. Here in this case, Scanner is not crated globally and instead created new object every time for new calculation, which is very bad practice considering the memory management.

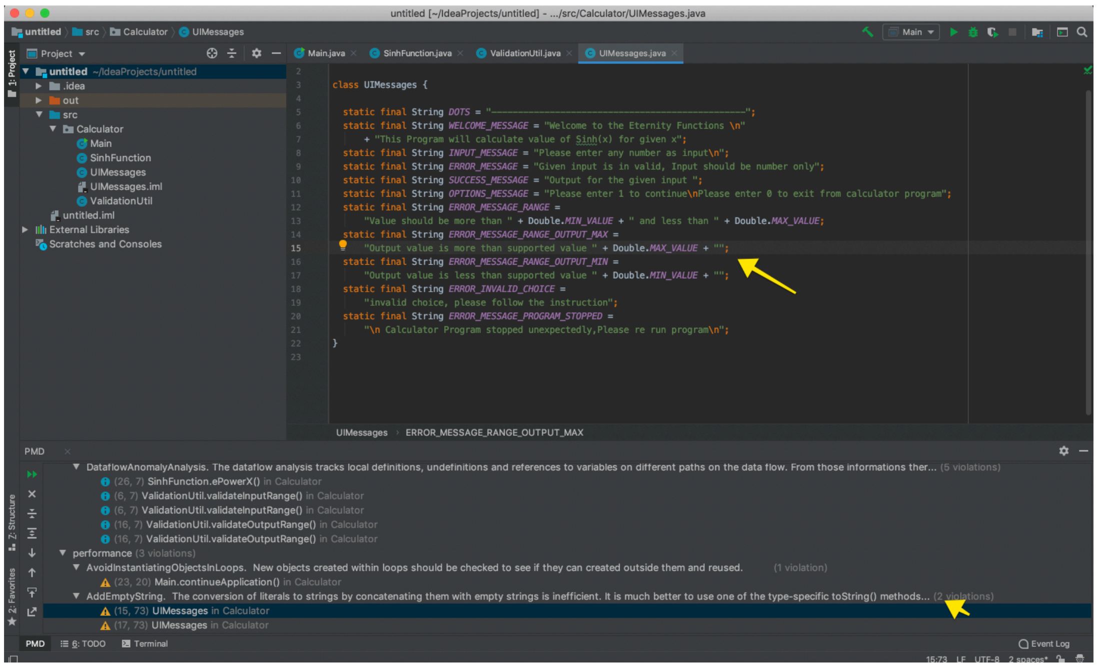


```
            System.out.println(UIMessages.ERROR_MESSAGE_PROGRAM_STOPPED);
            System.out.println(e.getMessage());
        }

    }

    private static void continueApplication(SinhFunction sinhFunction) {
        while (true) {
            System.out.println("\n" + UIMessages.DOTS + "\n" + UIMessages.OPTIONS_MESSAGE);
            Scanner sc = new Scanner(System.in);
            if (sc.hasNextInt()) {
                int choice = sc.nextInt();
                if (choice == 1) {
                    System.out.println(UIMessages.INPUT_MESSAGE);
                    sinhFunction.InitCalculation();
                } else if (choice == 0) {
                    System.exit( status: 0);
                } else {
                    System.out.println(UIMessages.ERROR_INVALID_CHOICE);
                }
            } else {
                System.out.println(UIMessages.ERROR_INVALID_CHOICE);
            }
        }
    }
```

```
  AvoidLiteralsInIfCondition.  Avoid using hard-coded literals in conditional statements. By declaring them as static variables or private members with descriptive names maintainabil... (1 violation)
    (26, 23) Main.continueApplication() in Calculator
  DataflowAnomalyAnalysis. The dataflow analysis tracks local definitions, undefinitions and references to variables on different paths on the data flow. From those informations ther... (5 violations)
    (26, 7) SinhFunction.ePowerX() in Calculator
    (6, 7) ValidationUtil.validateInputRange() in Calculator
    (6, 7) ValidationUtil.validateInputRange() in Calculator
    (16, 7) ValidationUtil.validateOutputRange() in Calculator
    (16, 7) ValidationUtil.validateOutputRange() in Calculator
performance (3 violations)
  AvoidInstantiatingObjectsInLoops.  New objects created within loops should be checked to see if they can created outside them and reused.   (1 violation)
    (23, 20) Main.continueApplication() in Calculator
  AddEmptyString.  The conversion of literals to strings by concatenating them with empty strings is inefficient. It is much better to use one of the type-specific toString() methods... (2 violations)
```

Figure 1.2: Creating Instance inside loop is bad practice and impediment to performance

• It showed string should not be concatenated with empty string. This practice does not add any value to the program and is unnecessary computation.

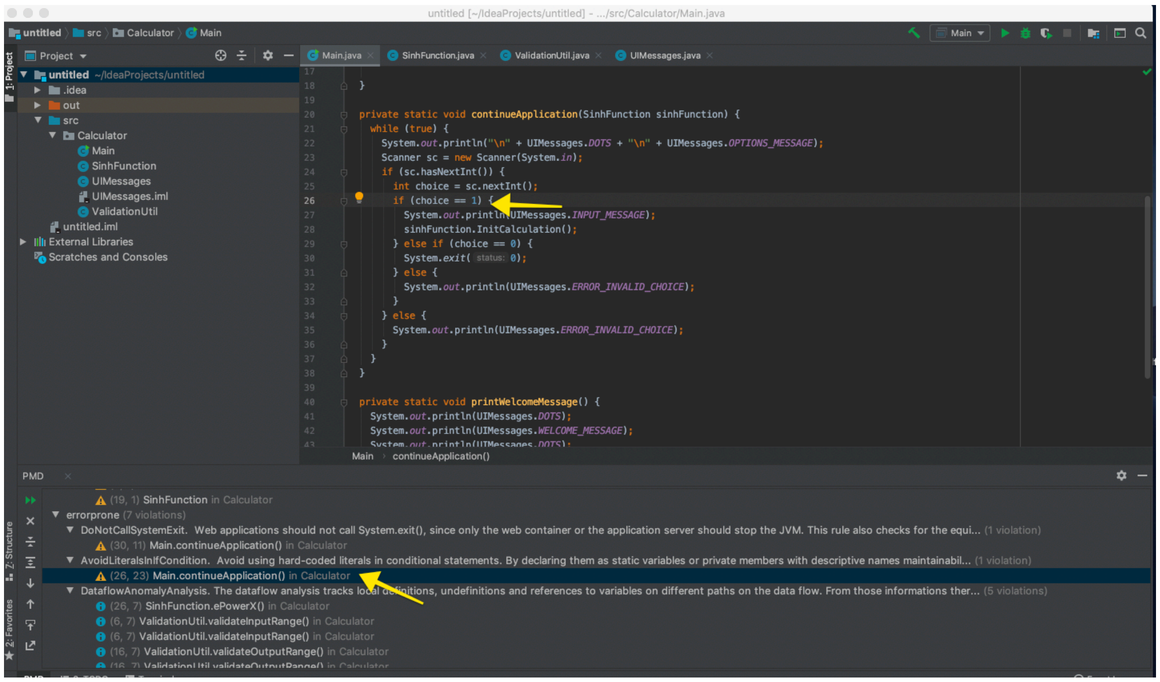

untitled [~/IdeaProjects/untitled] - .../src/Calculator/UIMessages.java
Main.java
SinhFunction.java
ValidationUtil.java
UIMessages.java
UIMessages › ERROR_MESSAGE_RANGE_OUTPUT_MAX
PMD
6: TODO
Terminal
Event Log

```
class UIMessages {

  static final String DOTS = "-----------------------------------------------";
  static final String WELCOME_MESSAGE = "Welcome to the Eternity Functions \n"
      + "This Program will calculate value of Sinh(x) for given x";
  static final String INPUT_MESSAGE = "Please enter any number as input\n";
  static final String ERROR_MESSAGE = "Given input is in valid, Input should be number only";
  static final String SUCCESS_MESSAGE = "Output for the given input ";
  static final String OPTIONS_MESSAGE = "Please enter 1 to continue\nPlease enter 0 to exit from calculator program";
  static final String ERROR_MESSAGE_RANGE =
      "Value should be more than " + Double.MIN_VALUE + " and less than " + Double.MAX_VALUE;
  static final String ERROR_MESSAGE_RANGE_OUTPUT_MAX =
      "Output value is more than supported value " + Double.MAX_VALUE + "";
  static final String ERROR_MESSAGE_RANGE_OUTPUT_MIN =
      "Output value is less than supported value " + Double.MIN_VALUE + "";
  static final String ERROR_INVALID_CHOICE =
      "invalid choice, please follow the instruction";
  static final String ERROR_MESSAGE_PROGRAM_STOPPED =
      "\n Calculator Program stopped unexpectedly,Please re run program\n";
}
```

```
▼ DataflowAnomalyAnalysis. The dataflow analysis tracks local definitions, undefinitions and references to variables on different paths on the data flow. From those informations ther... (5 violations)
    (26, 7) SinhFunction.ePowerX() in Calculator
    (6, 7) ValidationUtil.validateInputRange() in Calculator
    (6, 7) ValidationUtil.validateInputRange() in Calculator
    (16, 7) ValidationUtil.validateOutputRange() in Calculator
    (16, 7) ValidationUtil.validateOutputRange() in Calculator
▼ performance (3 violations)
  ▼ AvoidInstantiatingObjectsInLoops.  New objects created within loops should be checked to see if they can created outside them and reused.    (1 violation)
      (23, 20) Main.continueApplication() in Calculator
  ▼ AddEmptyString.  The conversion of literals to strings by concatenating them with empty strings is inefficient. It is much better to use one of the type-specific toString() methods... (2 violations)
      (15, 73) UIMessages in Calculator
      (17, 73) UIMessages in Calculator
```

Figure 1.3: Concatenation of empty string to string literal

• It showed string literal should not be used on fly, instead should be declared as variable, either static final or static.

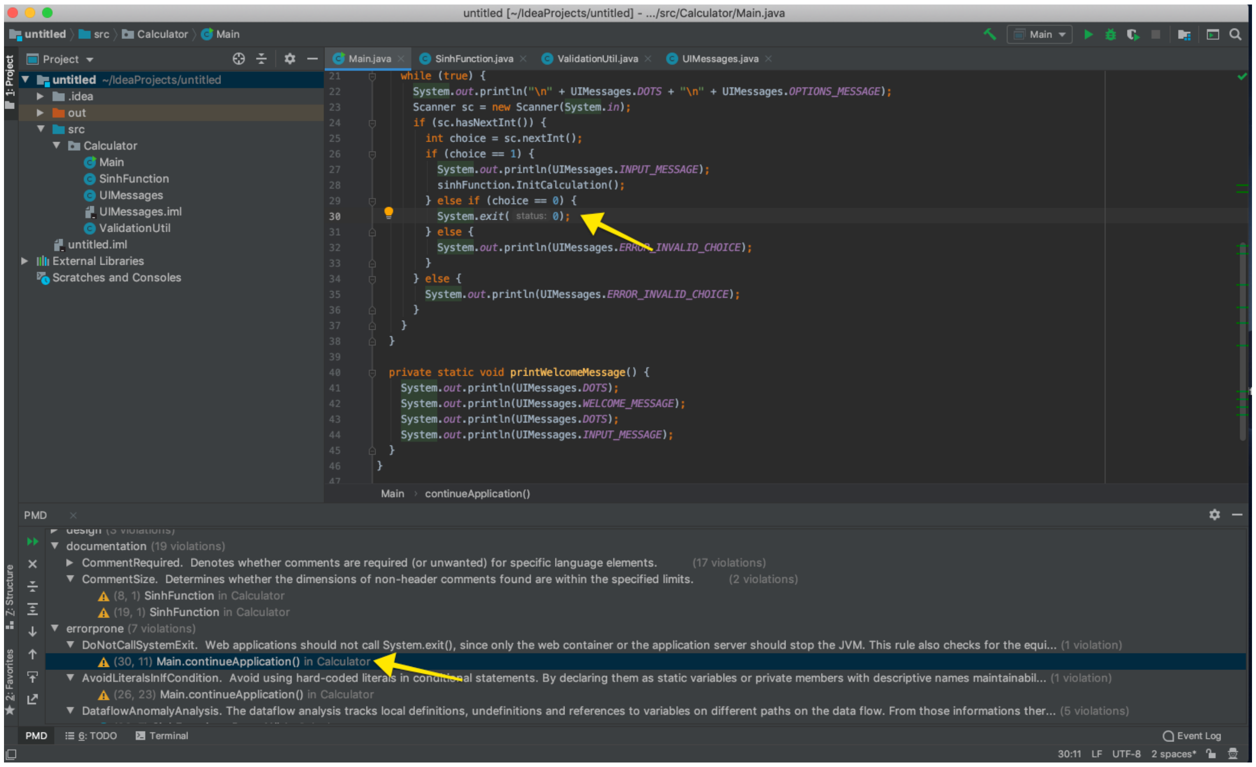

untitled [~/IdeaProjects/untitled] - .../src/Calculator/Main.java
Main.java
SinhFunction.java
ValidationUtil.java
UIMessages.java
Main › continueApplication()
PMD

```
  }

  private static void continueApplication(SinhFunction sinhFunction) {
    while (true) {
      System.out.println("\n" + UIMessages.DOTS + "\n" + UIMessages.OPTIONS_MESSAGE);
      Scanner sc = new Scanner(System.in);
      if (sc.hasNextInt()) {
        int choice = sc.nextInt();
        if (choice == 1) {
          System.out.println(UIMessages.INPUT_MESSAGE);
          sinhFunction.InitCalculation();
        } else if (choice == 0) {
          System.exit( status: 0);
        } else {
          System.out.println(UIMessages.ERROR_INVALID_CHOICE);
        }
      } else {
        System.out.println(UIMessages.ERROR_INVALID_CHOICE);
      }
    }
  }

  private static void printWelcomeMessage() {
    System.out.println(UIMessages.DOTS);
    System.out.println(UIMessages.WELCOME_MESSAGE);
    System.out.println(UIMessages.DOTS);
```

```
      (19, 1) SinhFunction in Calculator
▼ errorprone (7 violations)
  ▼ DoNotCallSystemExit.  Web applications should not call System.exit(), since only the web container or the application server should stop the JVM. This rule also checks for the equi... (1 violation)
      (30, 11) Main.continueApplication() in Calculator
  ▼ AvoidLiteralsInIfCondition.  Avoid using hard-coded literals in conditional statements. By declaring them as static variables or private members with descriptive names maintainabil... (1 violation)
      (26, 23) Main.continueApplication() in Calculator
  ▼ DataflowAnomalyAnalysis. The dataflow analysis tracks local definitions, undefinitions and references to variables on different paths on the data flow. From those informations ther... (5 violations)
      (26, 7) SinhFunction.ePowerX() in Calculator
      (6, 7) ValidationUtil.validateInputRange() in Calculator
      (6, 7) ValidationUtil.validateInputRange() in Calculator
      (16, 7) ValidationUtil.validateOutputRange() in Calculator
```

Figure 1.4: Static String Literals 4

• It showed System.exit() should not be used to stop JVM and should be used by only the Web container and Application Server.



untitled [~/IdeaProjects/untitled] - .../src/Calculator/Main.java

```java
    while (true) {
      System.out.println("\n" + UIMessages.DOTS + "\n" + UIMessages.OPTIONS_MESSAGE);
      Scanner sc = new Scanner(System.in);
      if (sc.hasNextInt()) {
        int choice = sc.nextInt();
        if (choice == 1) {
          System.out.println(UIMessages.INPUT_MESSAGE);
          sinhFunction.InitCalculation();
        } else if (choice == 0) {
          System.exit( status: 0);
        } else {
          System.out.println(UIMessages.ERROR_INVALID_CHOICE);
        }
      } else {
        System.out.println(UIMessages.ERROR_INVALID_CHOICE);
      }
    }
  }

  private static void printWelcomeMessage() {
    System.out.println(UIMessages.DOTS);
    System.out.println(UIMessages.WELCOME_MESSAGE);
    System.out.println(UIMessages.DOTS);
    System.out.println(UIMessages.INPUT_MESSAGE);
  }
}
```

```
design (3 violations)
documentation (19 violations)
  CommentRequired.  Denotes whether comments are required (or unwanted) for specific language elements.   (17 violations)
  CommentSize.  Determines whether the dimensions of non-header comments found are within the specified limits.   (2 violations)
    (8, 1) SinhFunction in Calculator
    (19, 1) SinhFunction in Calculator
errorprone (7 violations)
  DoNotCallSystemExit.  Web applications should not call System.exit(), since only the web container or the application server should stop the JVM. This rule also checks for the equi... (1 violation)
    (30, 11) Main.continueApplication() in Calculator
  AvoidLiteralsInIfCondition.  Avoid using hard-coded literals in conditional statements. By declaring them as static variables or private members with descriptive names maintainabil... (1 violation)
    (26, 23) Main.continueApplication() in Calculator
  DataflowAnomalyAnalysis. The dataflow analysis tracks local definitions, undefinitions and references to variables on different paths on the data flow. From those informations ther... (5 violations)
```

Figure 1.5: System.Exit() should not be used