Java Try-Catch Block - Assignment

Instructions:

You are provided with the `TryCatchBlock.java` class. Your task is to implement the `main()` method to demonstrate exception handling using try-catch blocks. Below are the specific tasks you need to complete, including exact exception types and handling methods.

Task 1: Handling ArithmeticException

1. **Objective**: Handle an `ArithmeticException` in the try-catch block.

2. **Details**:
  - Simulate an operation that could throw an ArithmeticException (like division by zero).

3. **Steps**:
  - Wrap the division operation `10 / 0` in a try-catch block.
  - In the `catch` block, catch the `ArithmeticException`.
  - Print the exception message using `e.getMessage()` as `System.out.println("Exception caught: " + e.getMessage());`

Task 2: Handling IOException for File Operations

1. **Objective**: Handle an `IOException` in the try-catch block when performing file operations.

2. **Details**:
  - Simulate a file operation that could throw an exception, such as trying to access a non-existent file.
  - You have to read a non-existent file with name as `"non_existent_file.txt"`, and you need to simulate an `IOException` if the file doesn't exist.

3. **Steps**:
  - Check if the file at the path `"non_existent_file.txt"` exists.
  - If it doesn't, throw an `IOException` with the message `"File not found: <path>"`.
  - In the `catch` block, catch the `IOException` and print the exception message as `System.out.println("Exception caught: " + e.getMessage());`

Final Deliverable:

1. Implement the `main()` method in the `TryCatchBlock.java` class.
2. Ensure that:
   - You **handle the `ArithmeticException`** by wrapping the division operation in a try-catch block and printing the exception message.
   - You **handle the `IOException`** by checking if the file exists and throwing an exception if the file is not found, then printing the exception message.
3. Do not include any print statements outside of the try-catch blocks for handling the exceptions.

Execution Steps to Follow:

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**

2. **To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) □ Terminal □New Terminal.**

3. **This editor Auto Saves the code.**

4. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**

5. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**

6. **To run your project use command:**

   ```
   mvn compile exec:java -Dexec.mainClass="com.yaksha.assignment.TryCatchBlock"
   ```

7. **To test your project test cases, use the command**

   ```
   mvn test
   ```

You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.