**Unit-3**

**Exception handling**

**Exception handling:** Exception-type, Uncaught Exception, Using try catch, throw, throws, finally, Throwable class and object, Exception Classes, Create own Exception Subclass.

**Exception –**

1. An exception is unexpected, unwanted, abnormal situation that occurred at the runtime of a program.
2. When an exception occurs, it disrupts the normal flow of the program.

An exception can occurs for many reasons:

1. Invalid user input
2. Device failure
3. Loss of internet connection
4. Physical limitation
5. Code errors
6. Opening an unavailable file.

Some of these exceptions are caused by user errors, others by programmer errors, and other by physical resources that have failed in some manner.

Based on these, we have 3 categories of exception, which are necessary to understand to know how exception handling works in java.

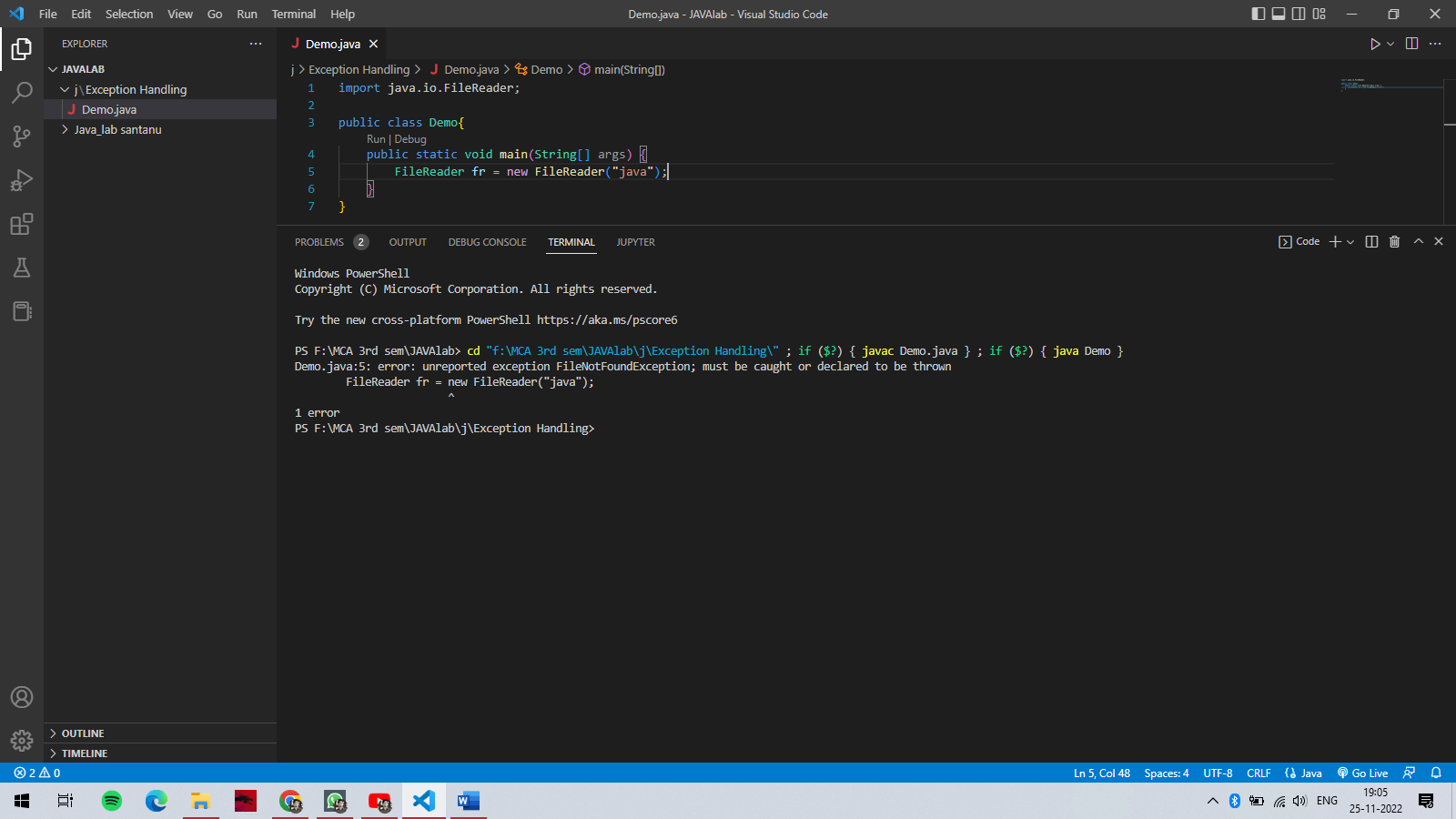
EXCEPTION

CHECKED EXCEPTION UNCHECKED EXCEPTION ERRORS

1. **Checked Exception –**
2. A checked exception is an exception that is checked by the compiler at compilation time.
3. These are also called as compile time exceptions
4. These exceptions cannot be simply ignored, the programmer should take care(handle) of these exceptions.

**e.g.,** ClassNotFoundException, FileNotFoundException, SQLException, IOException etc, are the examples of checked exception.

**Program:** When we read the file on location which does not exist then it throws FileNotFoundException;



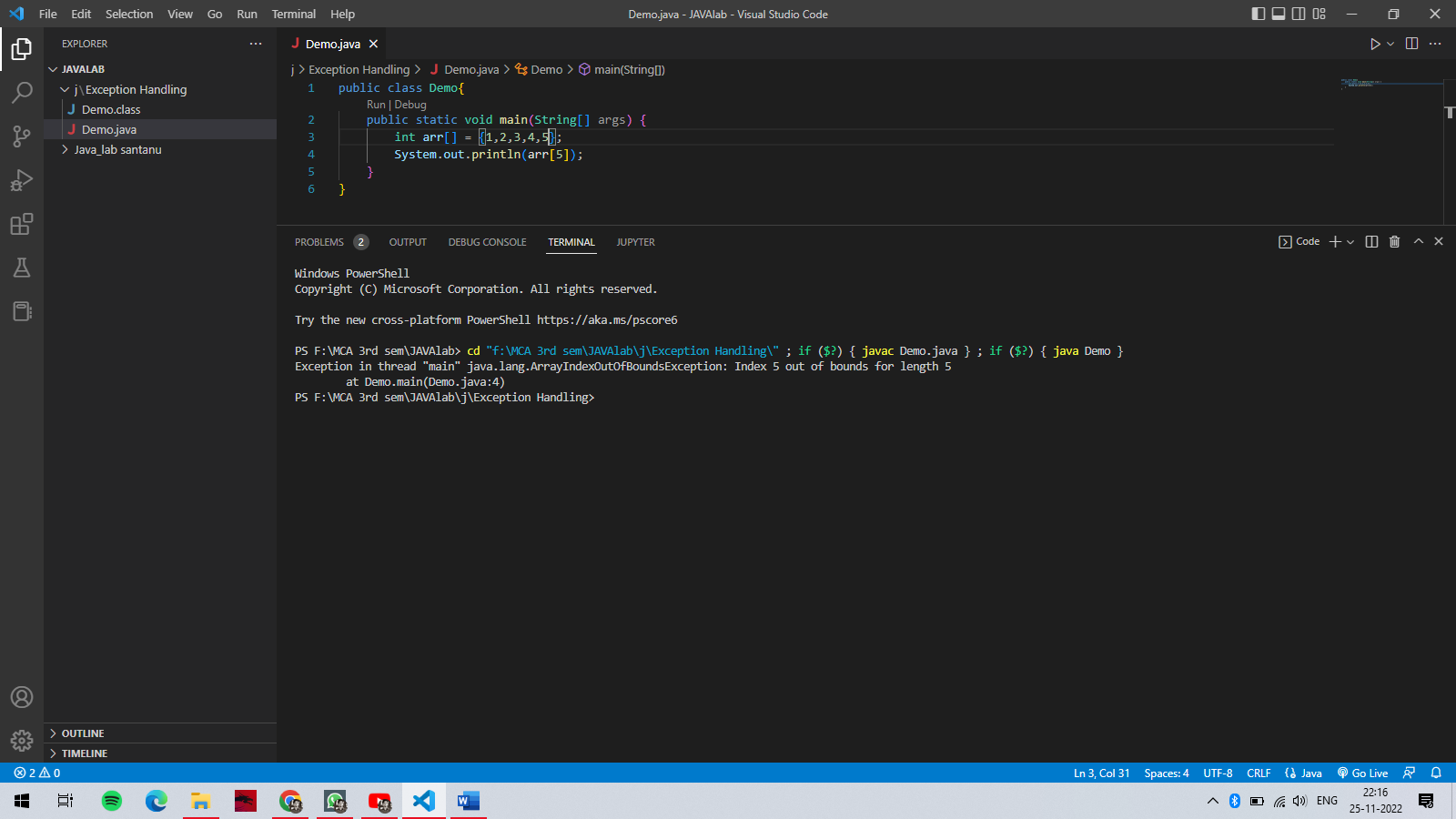
**O/P: -**

FileNotFoundException

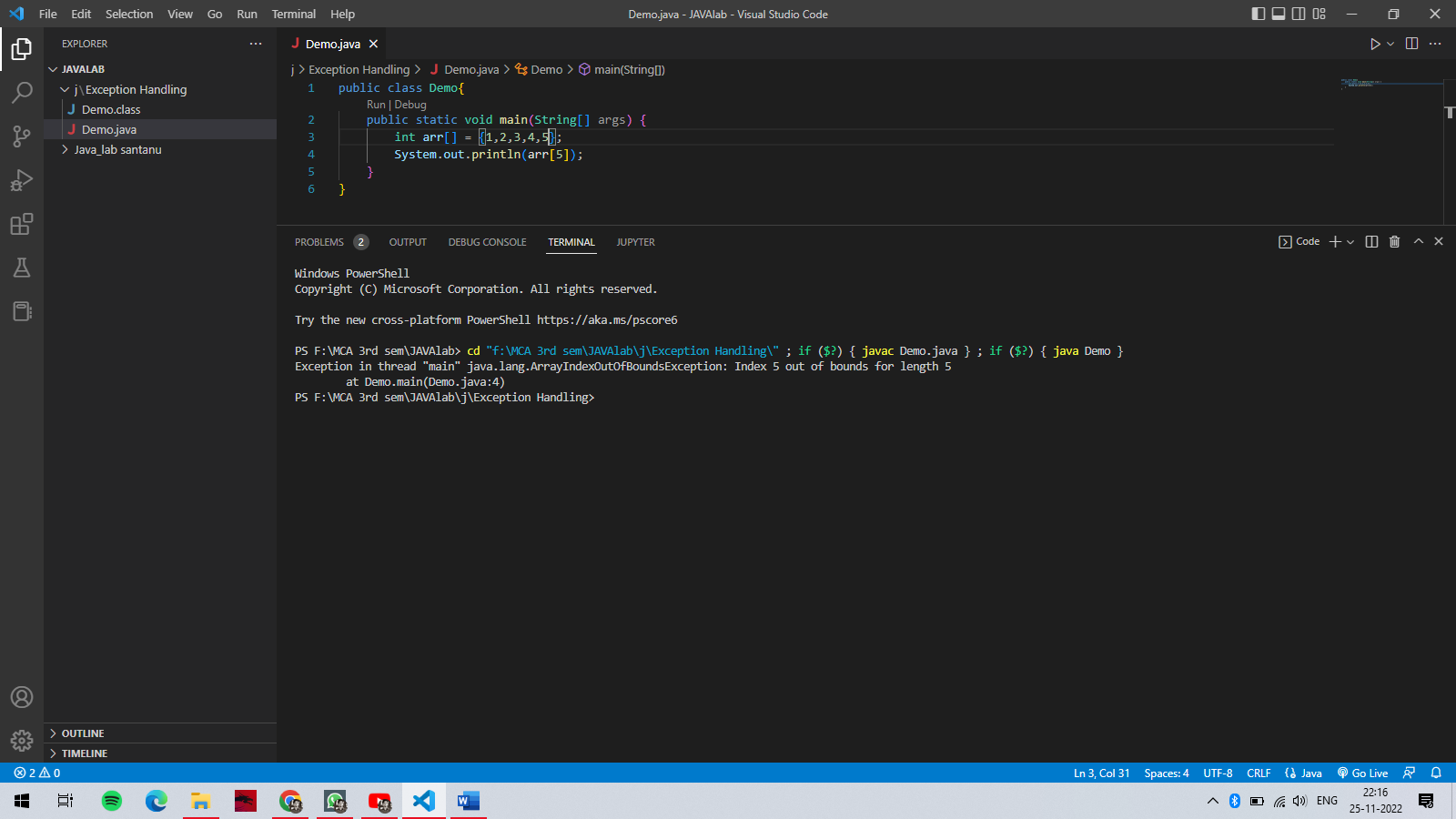
1. **Unchecked Exception –**
2. An unchecked exception is an exception that occurs at the time of execution.
3. These are also called as run-time exception.
4. These includes programming bugs such as logic errors or improper use of an API
5. Runtime exceptions are ignored at the time of compilation.

**e.g.,** ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc.

**Program:** If we have an array of size 5, and trying to access the 6th element of the array then an ArrayIndexOutOfBoundsException occurs.



**O/P: -**



1. **Errors –**
2. These are not exceptions at all, but problems that arise beyond the control of the user or the programmer.
3. Errors are typically ignored in your code because you can rarely do anything about an error.
4. If a stack overflow occurs, an error will arise. They are also ignored at the time of compilation.

**e.g.,** JVM running out of memory, memory leaks, stack overflow errors, library incompatibility, infinite recursion etc.

**Exception Handling –**

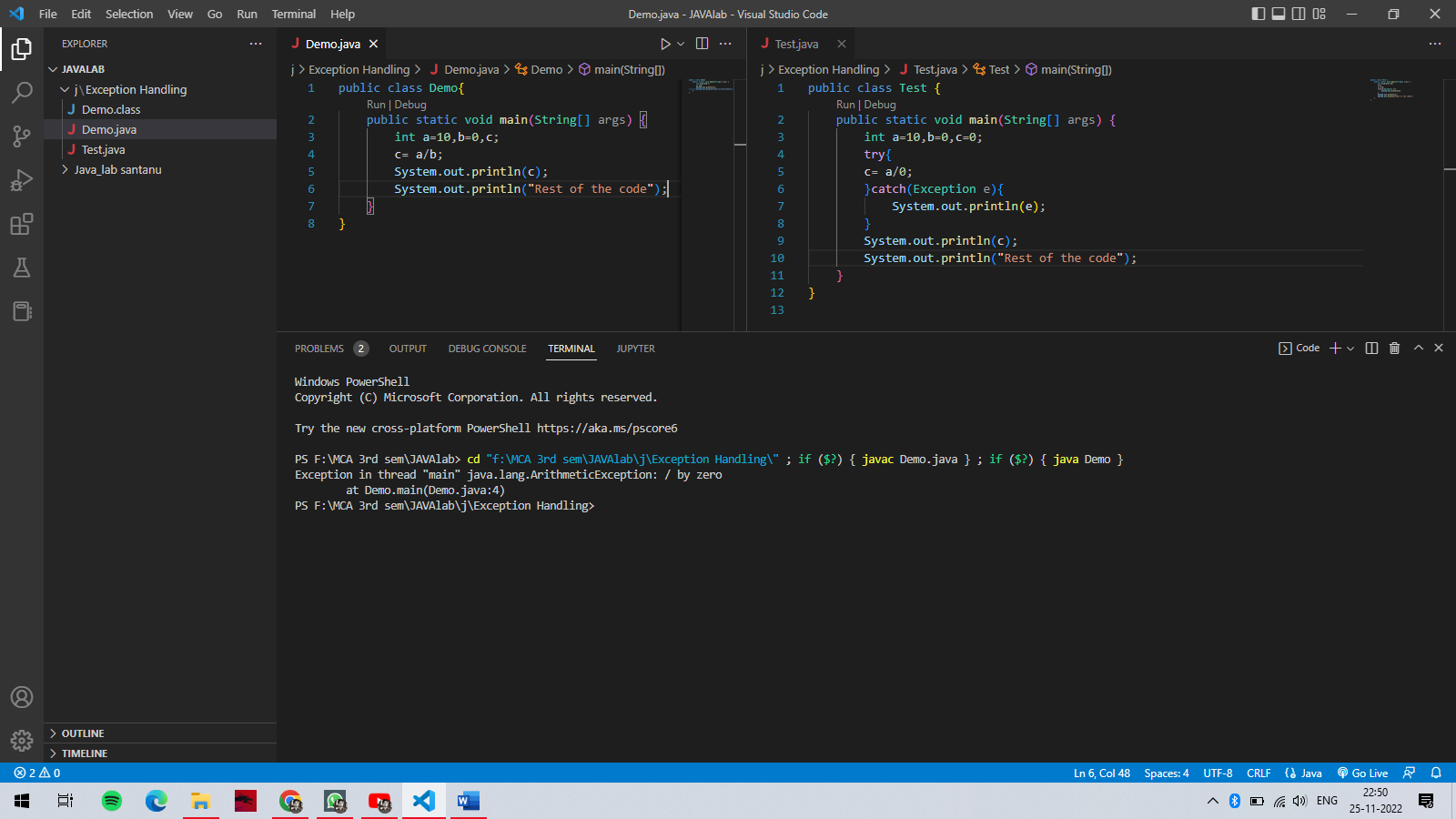
Exception handling doesn’t mean repair an error, but we have to find an alternate way to continue rest of the program normally is called exception handling.

1. Exception can be handled using try, catch and finally.
2. In a try block there can be zero or more catch block but only one finally block.
3. It is possible to handle multiple exceptions by using multiple catch block.
4. It is possible to serve multiple exceptions in a try block but at a time only one will be executed.
5. It is possible to create try block with catch block.
6. It is not possible to create catch block without try block.
7. It is possible to create try block inside try block which is called nested try block.

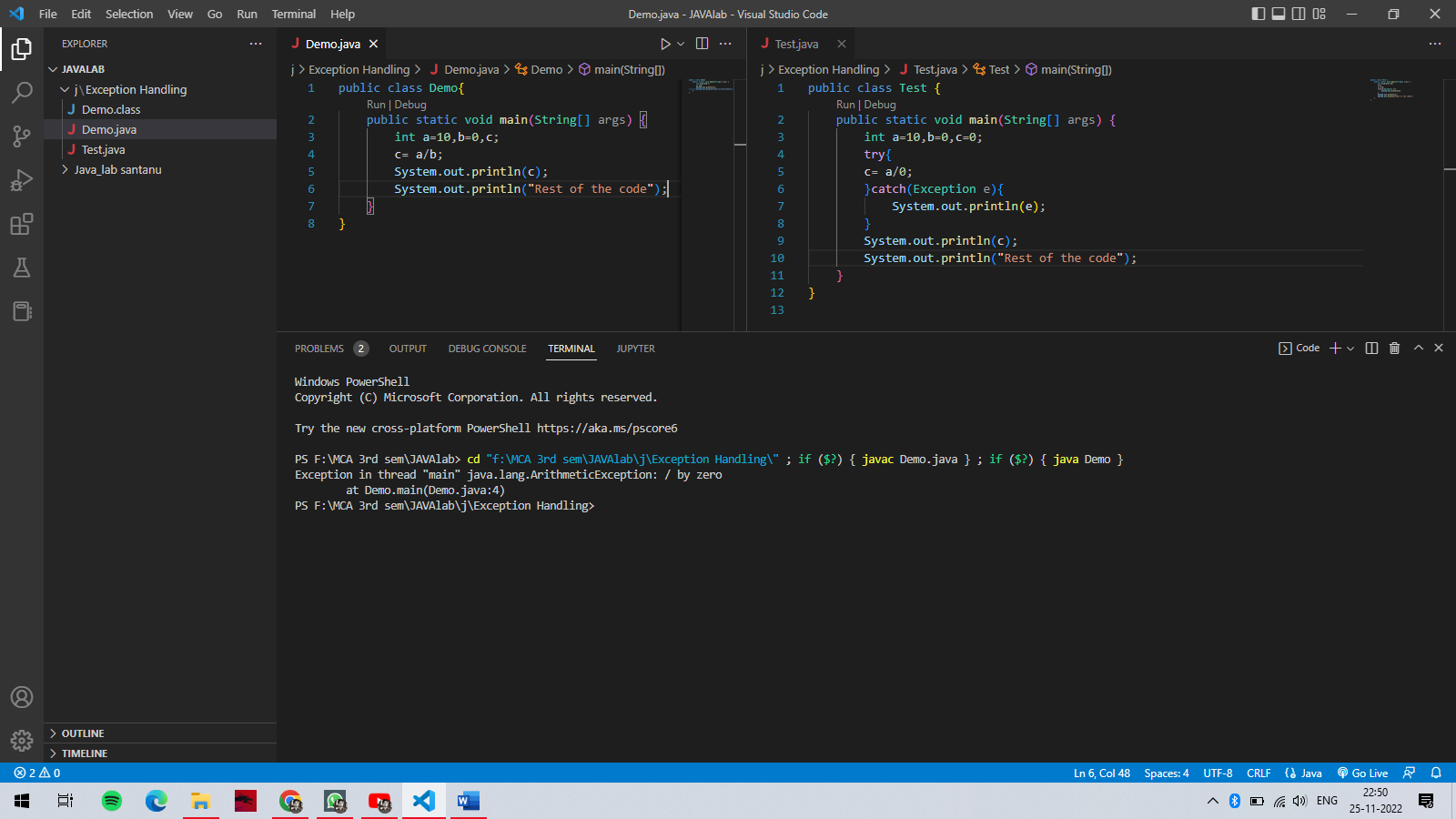
The following techniques or keywords are used to work with exception handling: -

1. try
2. catch
3. finally
4. throw
5. throws

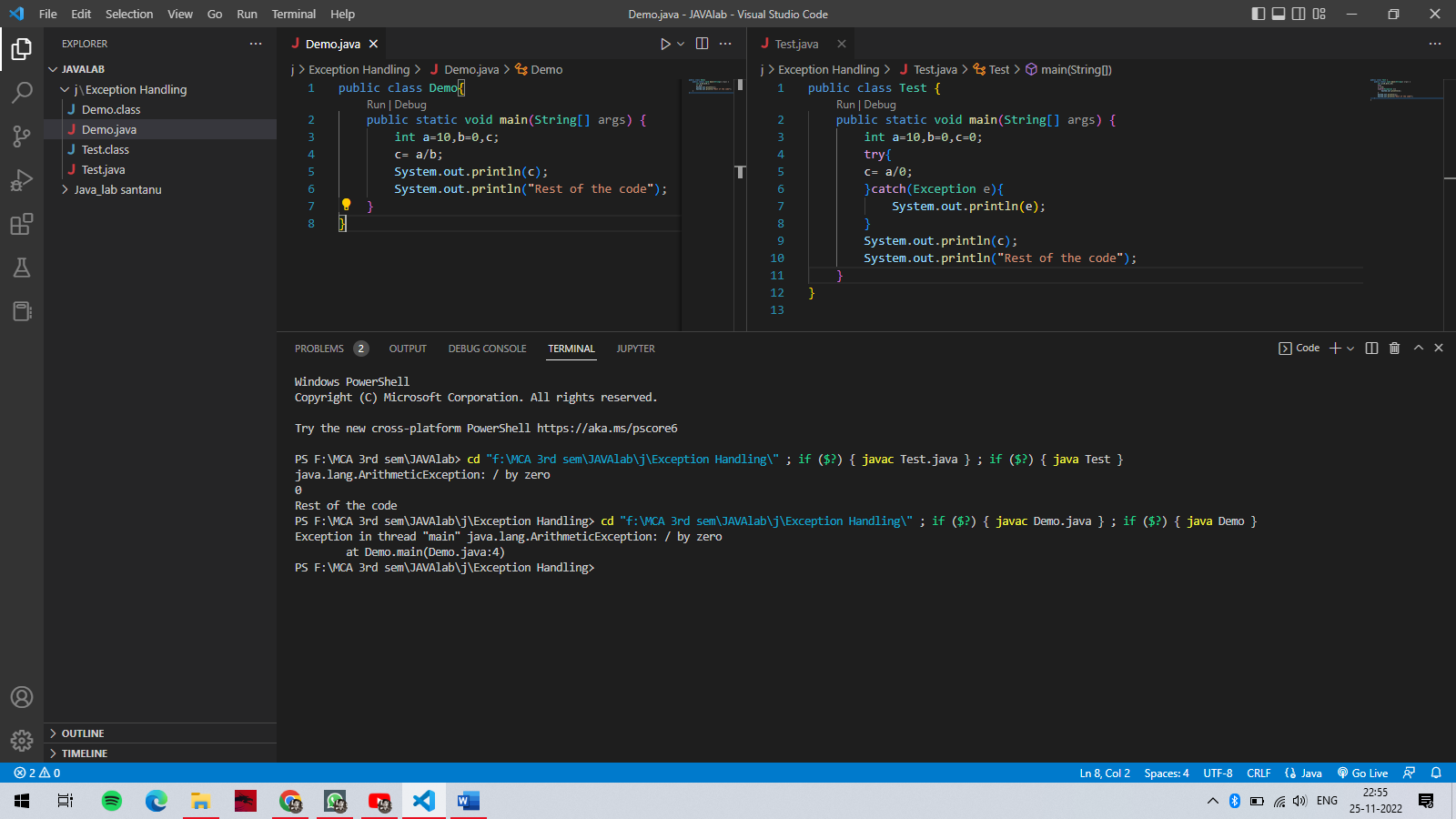
A glimpse of code without exception handling and code with exception handling are:



O/P: - 1.



O/P: - 2.



1. try block: -
2. The try keyword is used to specify a block where we should place an risky/exception code
3. Try block can’t be used alone, it must be followed by either catch or finally.
4. If an exception occurs at particular statement in the try block, the rest of the clock code will not execute. So try to avoid the code in try block that will not throw an exception.
5. catch block: -
6. The catch is used to handle the exception.
7. Catch block can’t be used alone, it must be preceded with try block
8. It can be followed by finally block later.
9. Catch block will not be executed if there is no exception

**Note: 1.** The catch block requires a parameter that shoud be of type exception.

1. Finally block: -
2. The finally block is used to execute the necessary code of the program.
3. Finally block is always executed whether an exception handled or not.

**Syntax of “try catch”**

try{

//some lines of code

}

catch(Exception e){}

**Syntax of “try finally”**

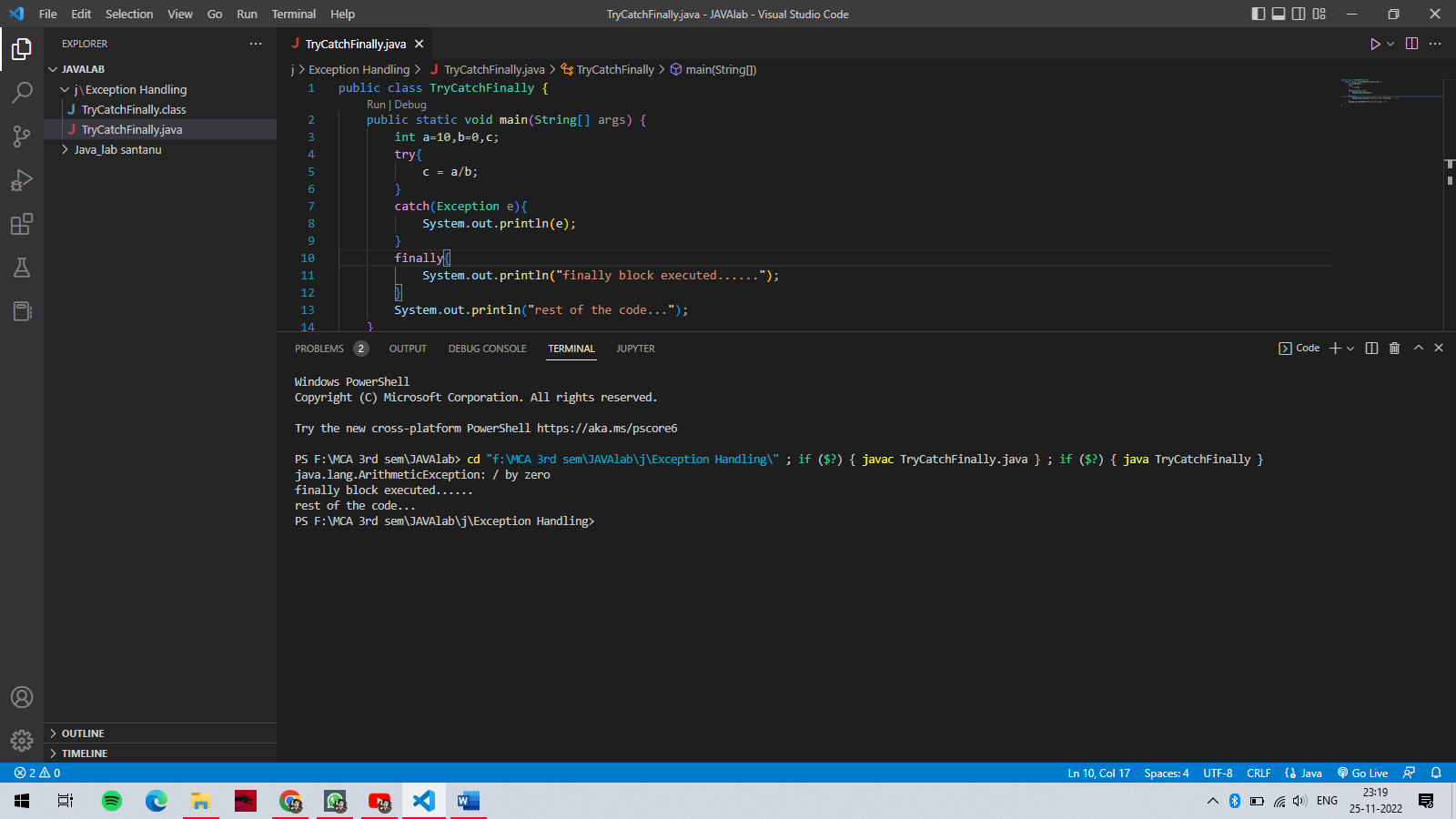
try{

//some lines of code

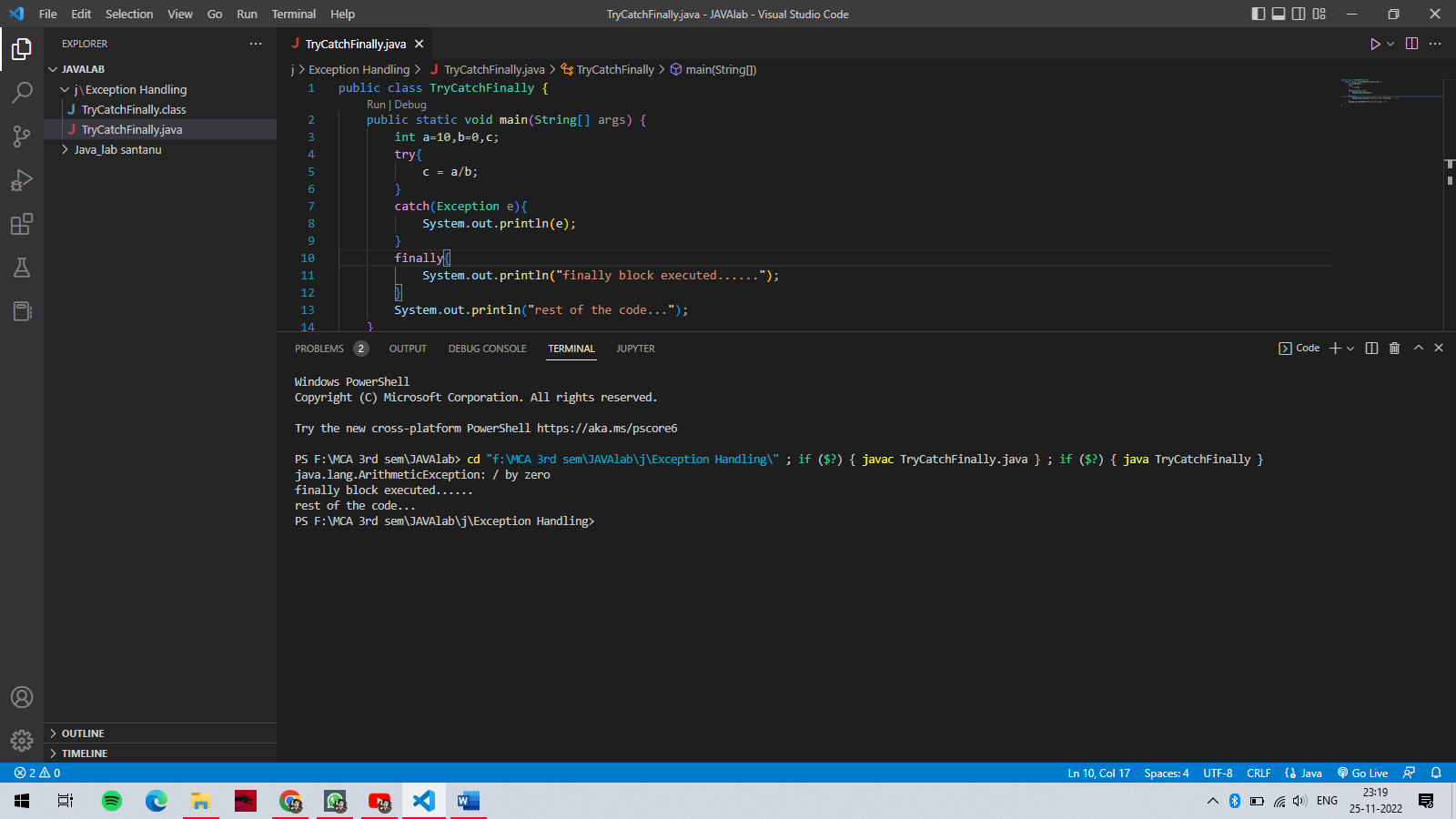
}

finally{}

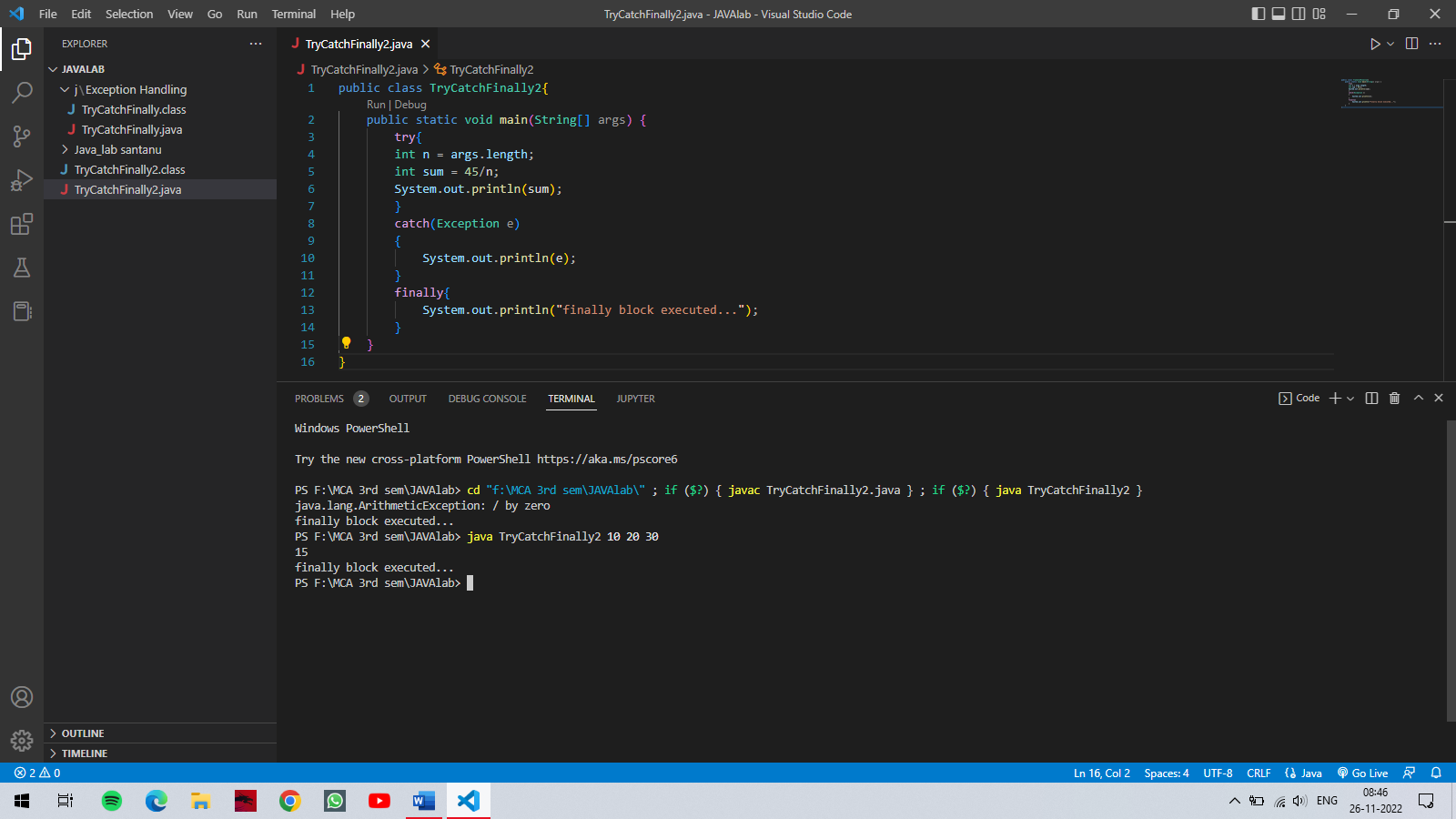
**Program:** Demonstrating a simple try catch finally block



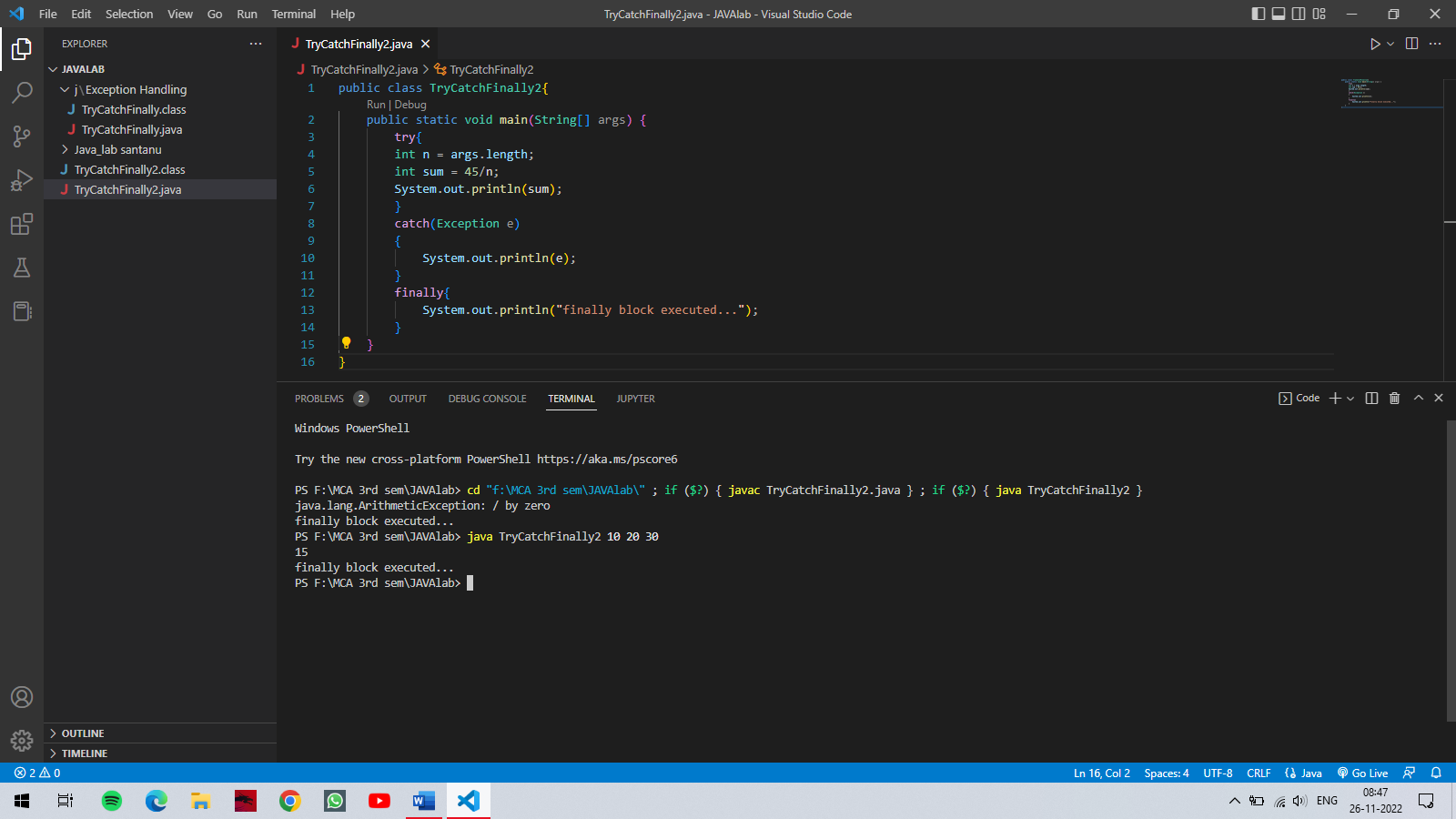
O/P: -

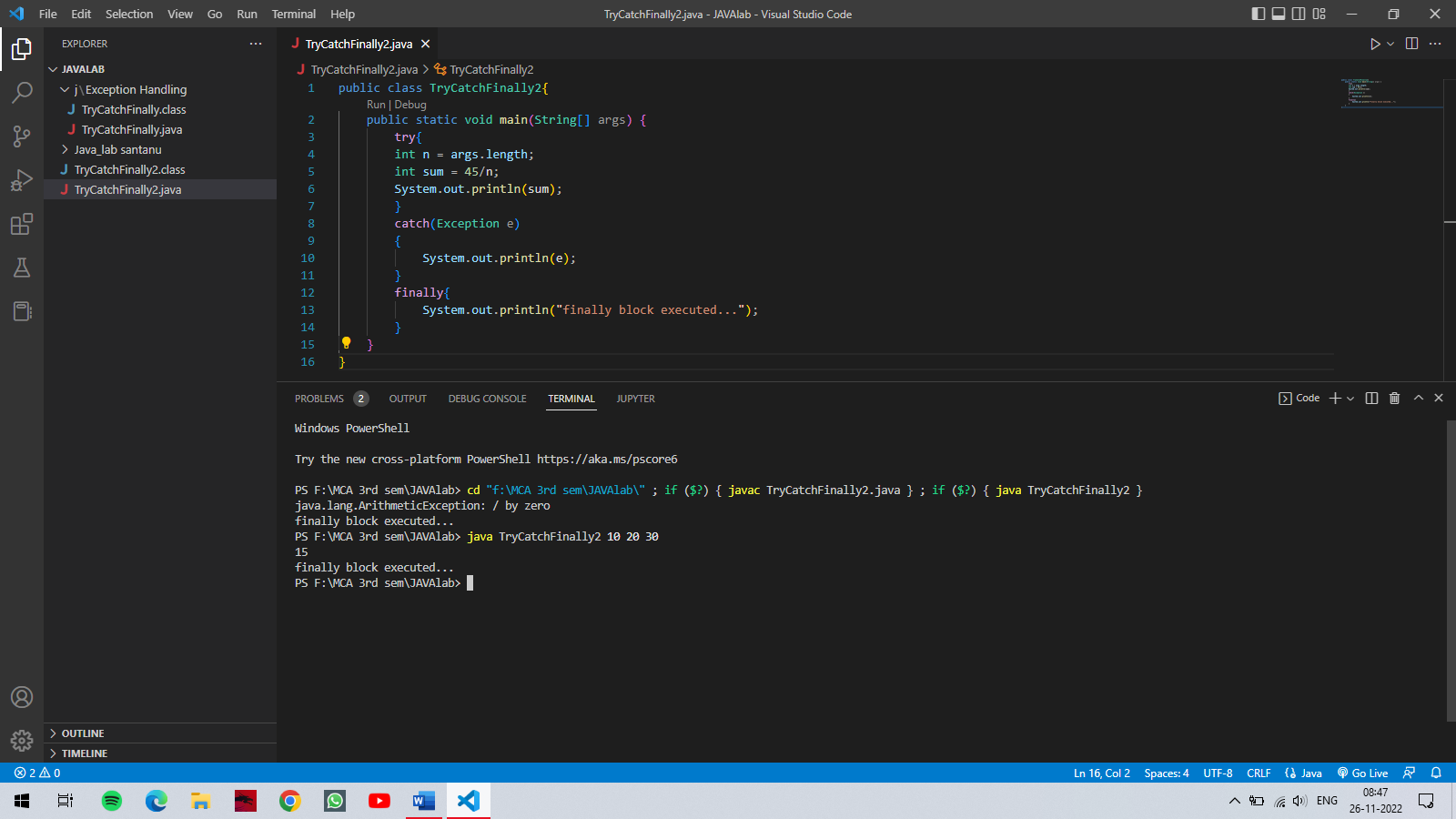


**Program:** Demonstrating a try catch finally block using command line arguments.

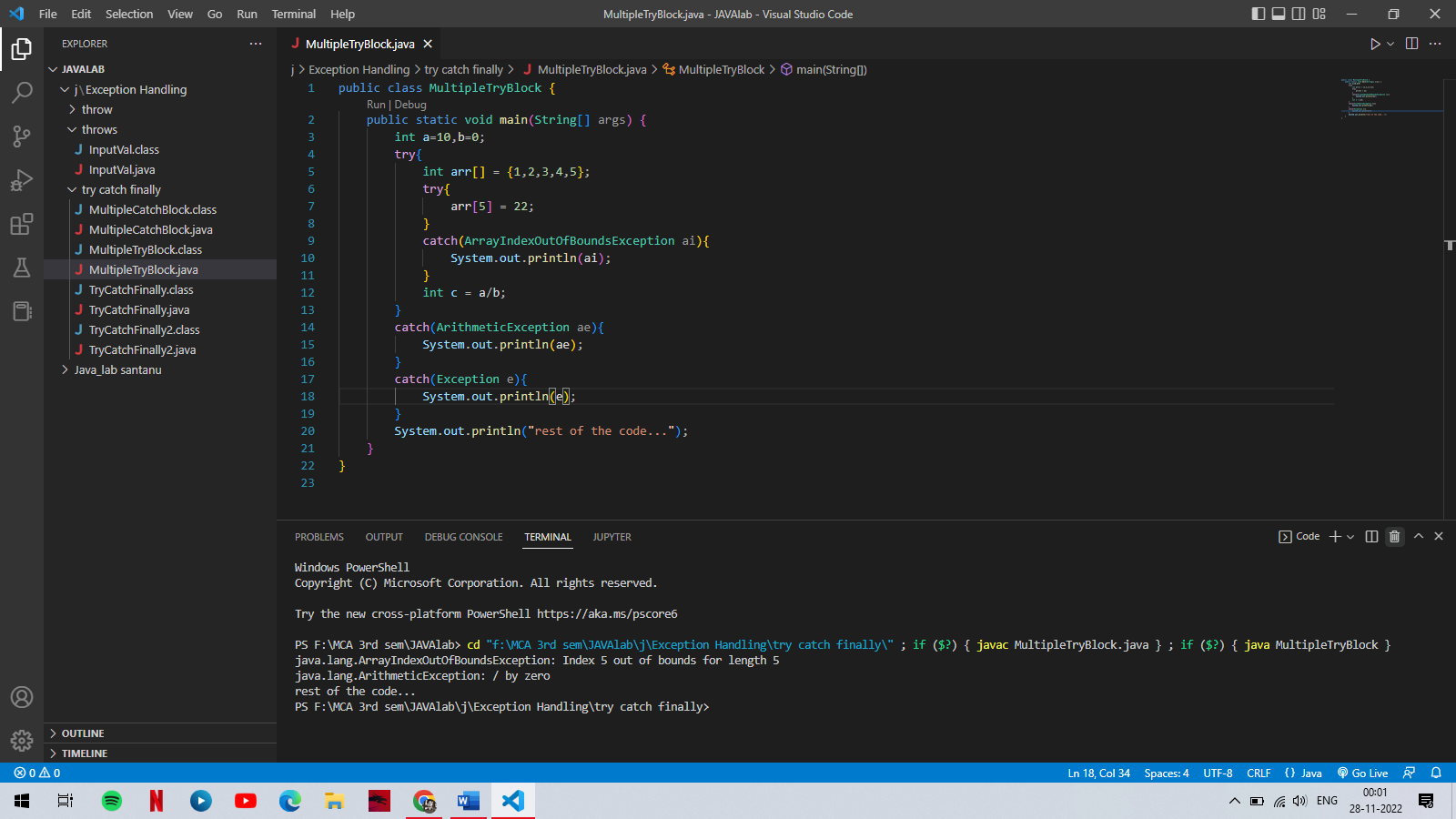


O/P: -

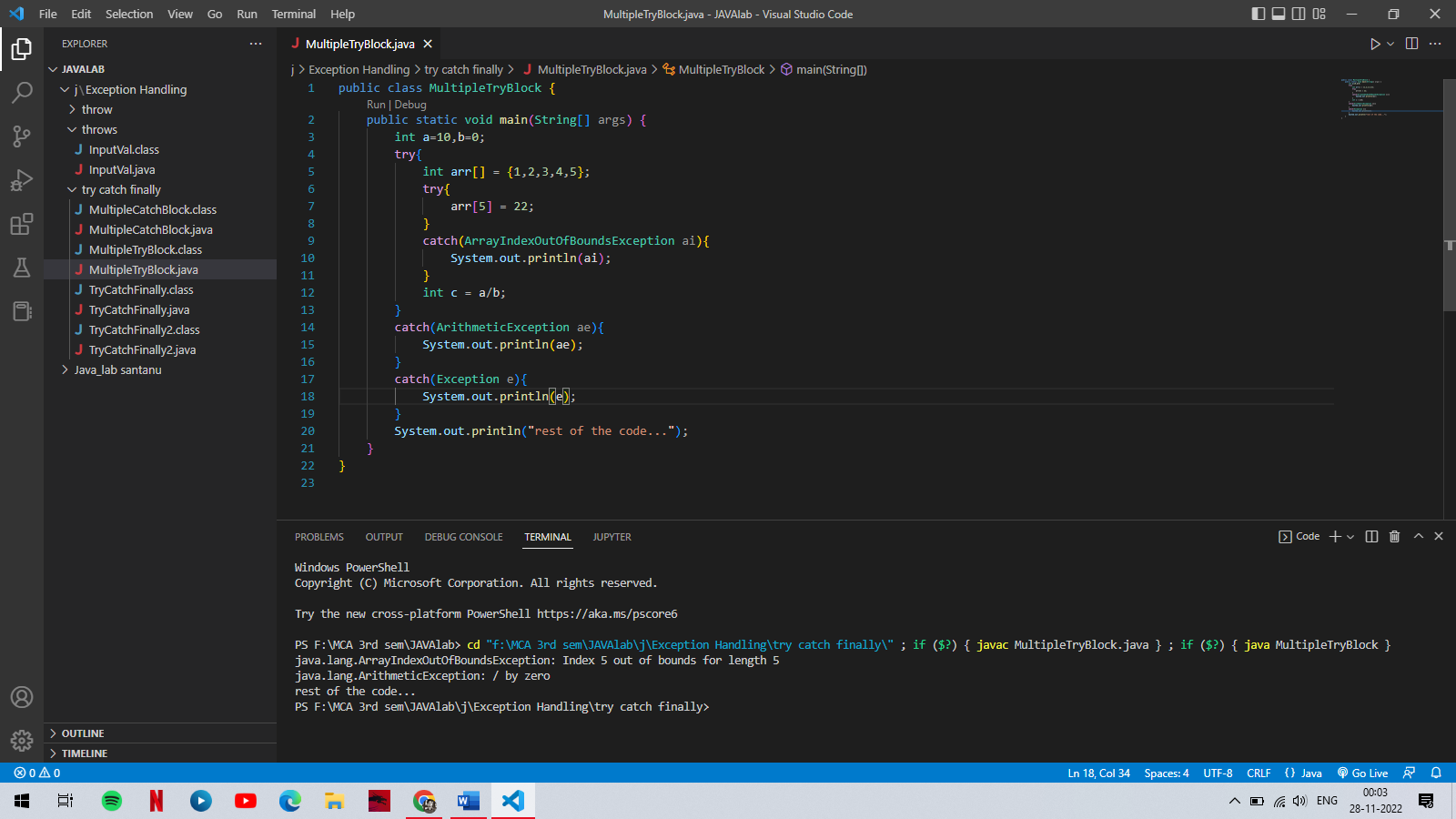




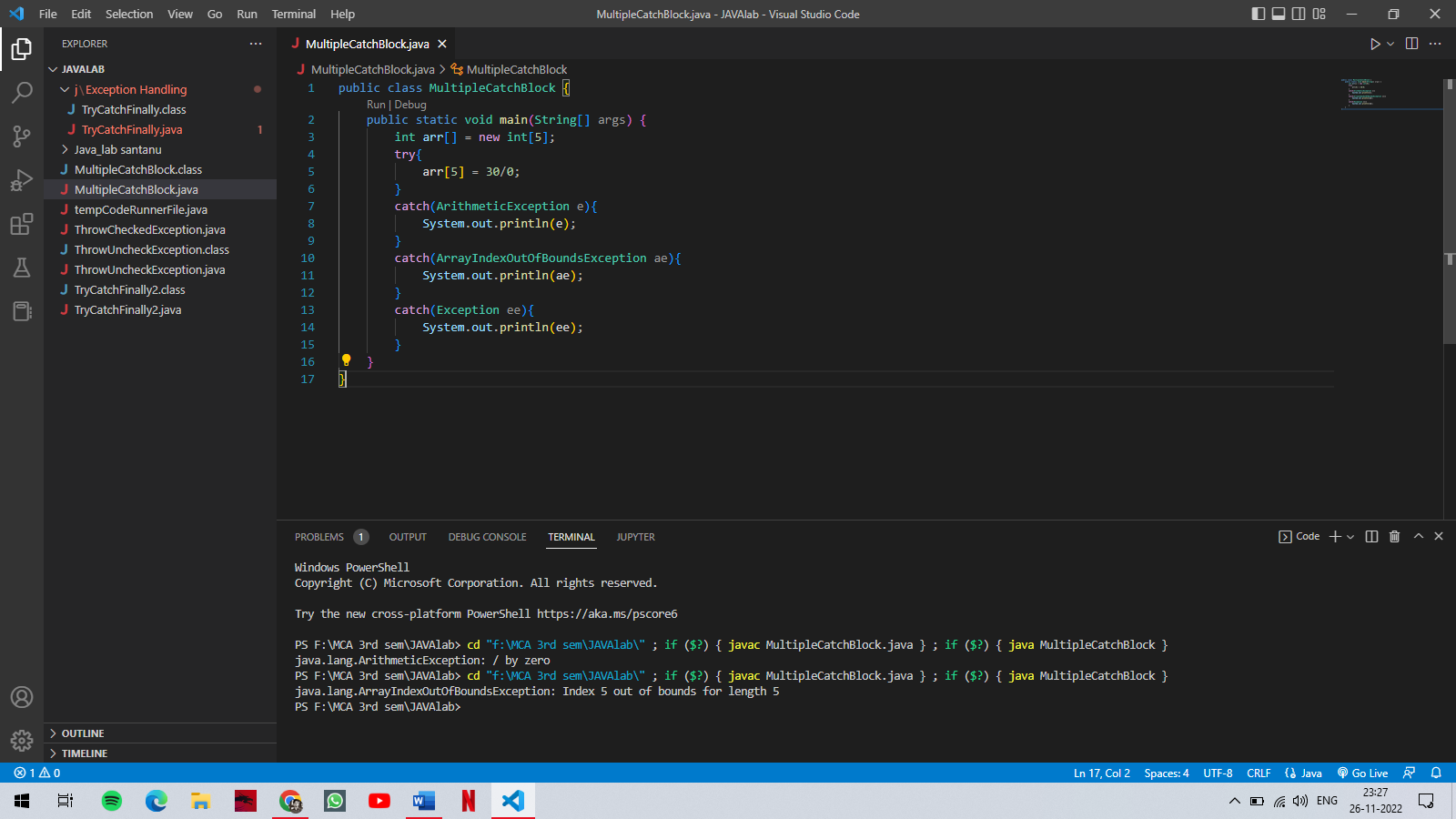
**Program:** Demonstrating a program of multiple try blocks.



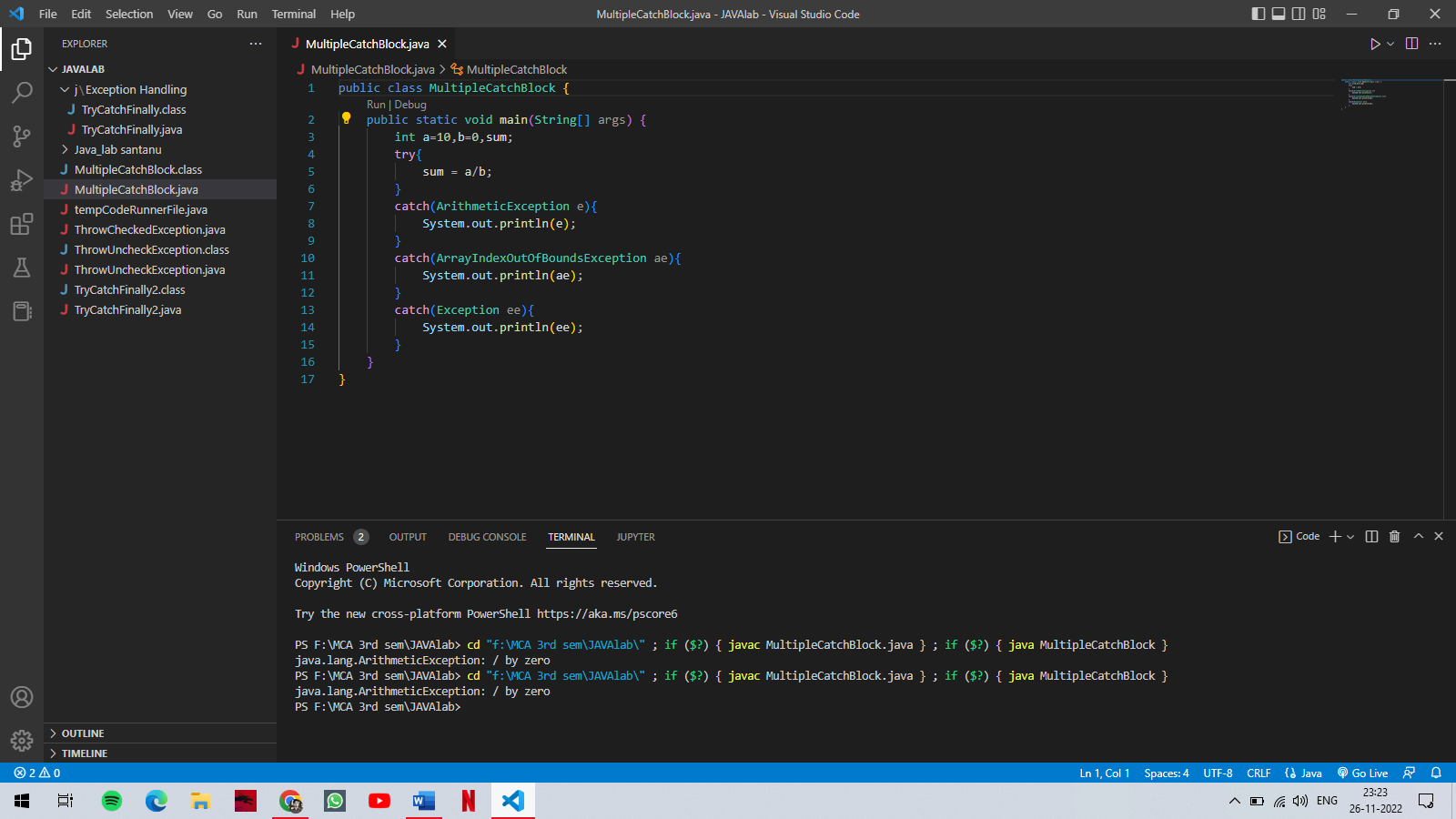
O/P: -



**Program:** Demonstrating a program of multiple catch blocks.



O/P: -



1. throw: -
2. The throw keyword is used to explicitly throw an exception by programmer.
3. We can throw either checked or unchecked exception.
4. Only object of throwable class or subclass can be thrown.

throwable class

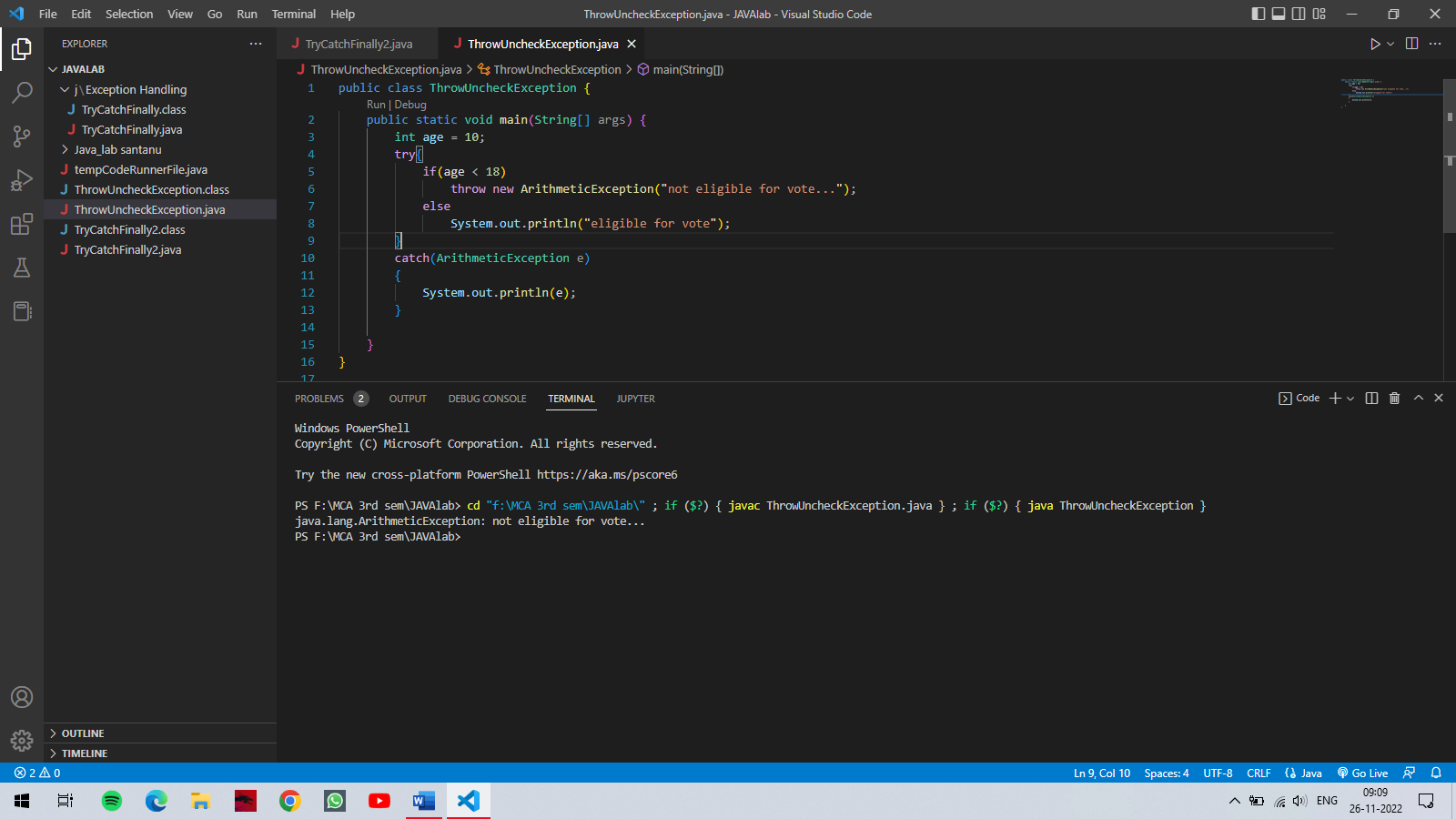
Exception Errors

1. Program execution terminates or stops on encountering throw statement, and the closest catch statement is checked for matching type exception.
2. Syntax: -

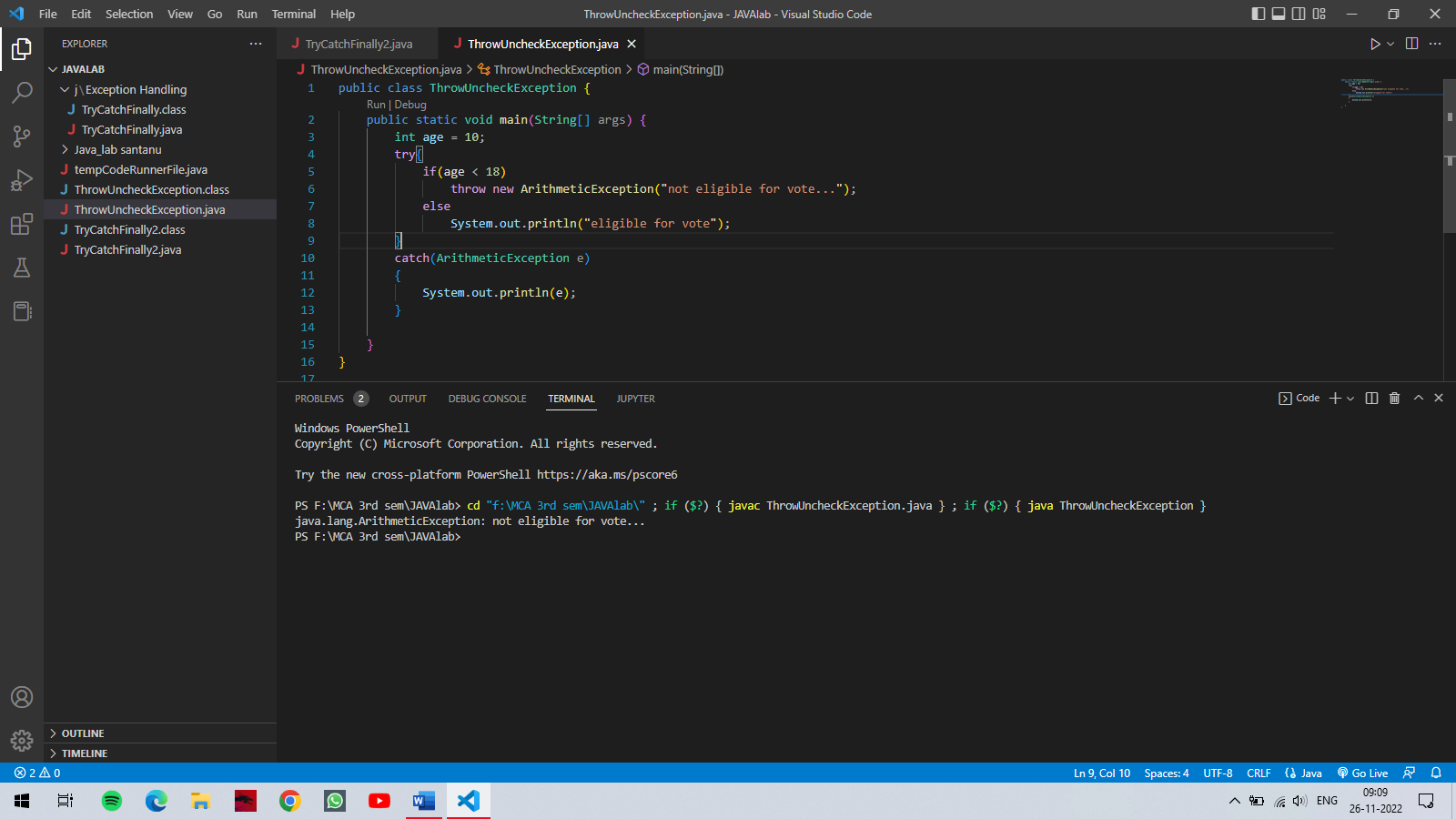
throw throwableinstance;

1. throw keyword is used to create user-defined exception.
2. To create user-defined exception extend Exception class.

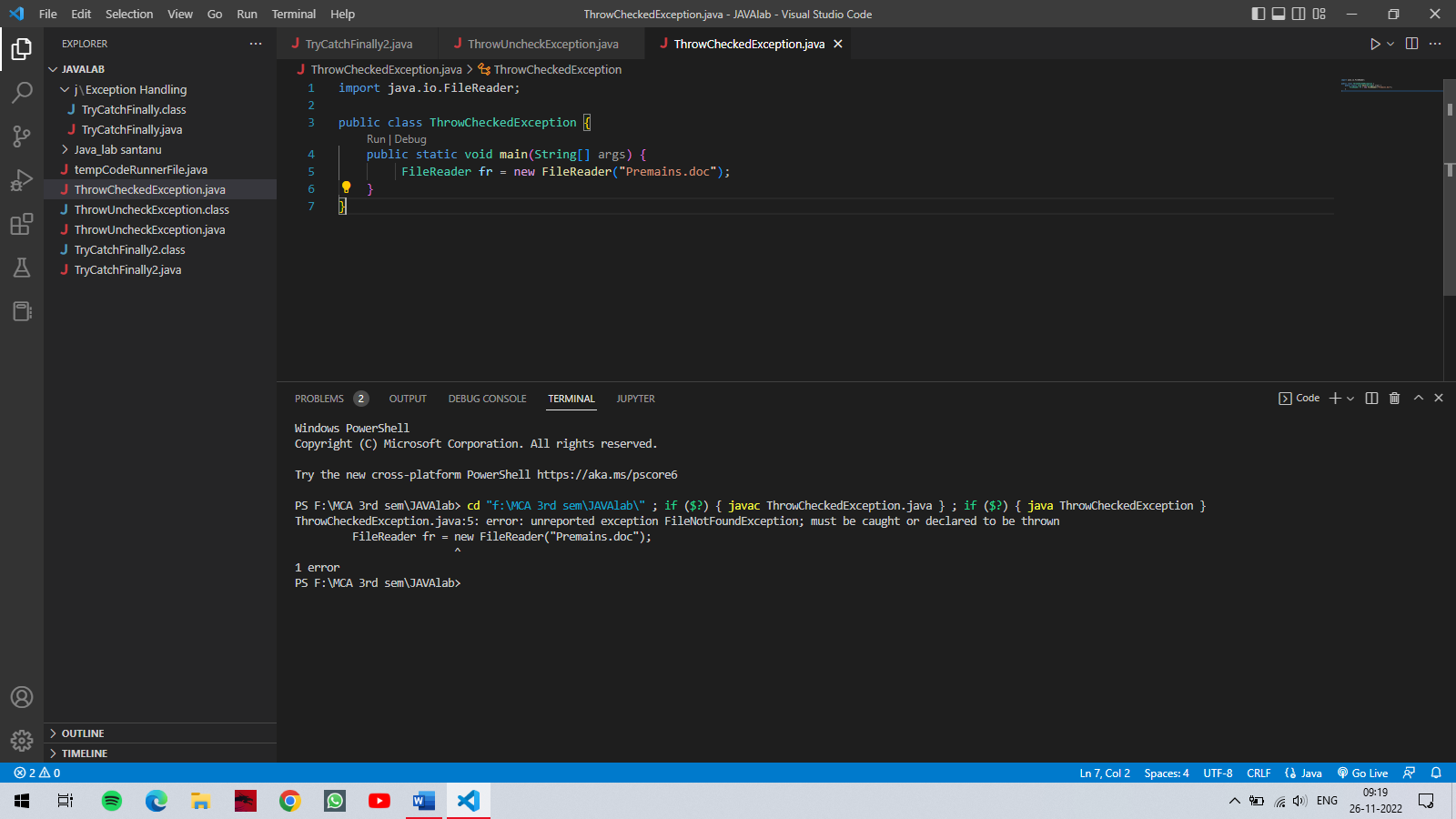
**Program:** throw unchecked exception.



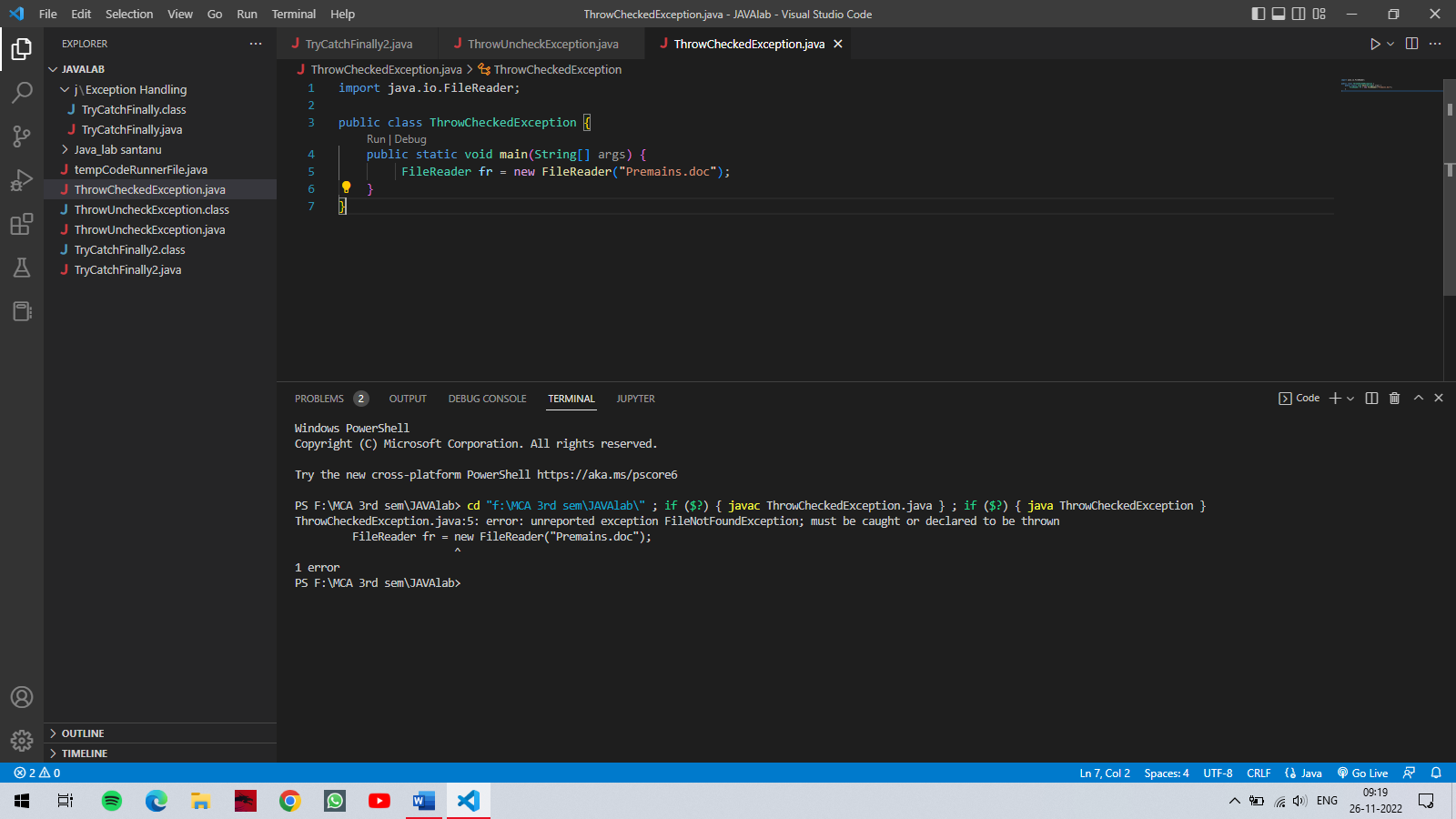
O/P: -



**Program:** throw checked exception.



O/P: -



**Steps to create user-defined exception –**

1. User defined exception can be created simply by extending Exception class. This is done as :

**class UserDefinedException extends Exception;**

1. If you do not want to store any exception details define a default constructor in your own exception class. This can be done as follows:-

**UserDefinedException()**

**{**

**}**

1. If you want to store exception details, define a parameterized constructor with string as a parameter, call superclass(Exception) constructor from this, and store variable “str”.

**UserDefinedException(String str)**

**{**

**super(str);**

**}**

1. In the last step, we need to create an object of user-defined exception class and throw it using “throw”.

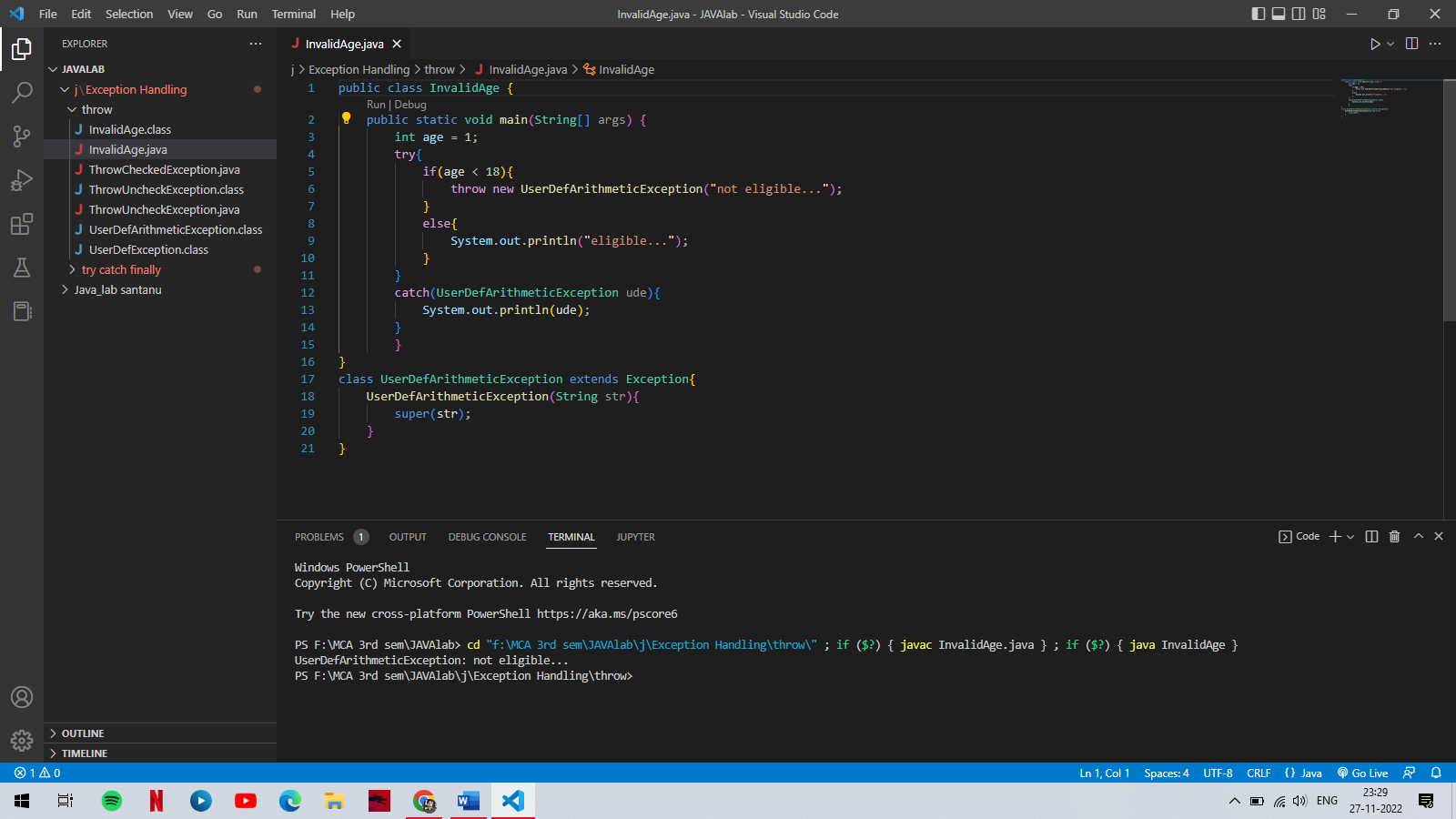
**UserDefinedException obj = new**  **UserDefinedException(“Exception details”);**

**throw obj**

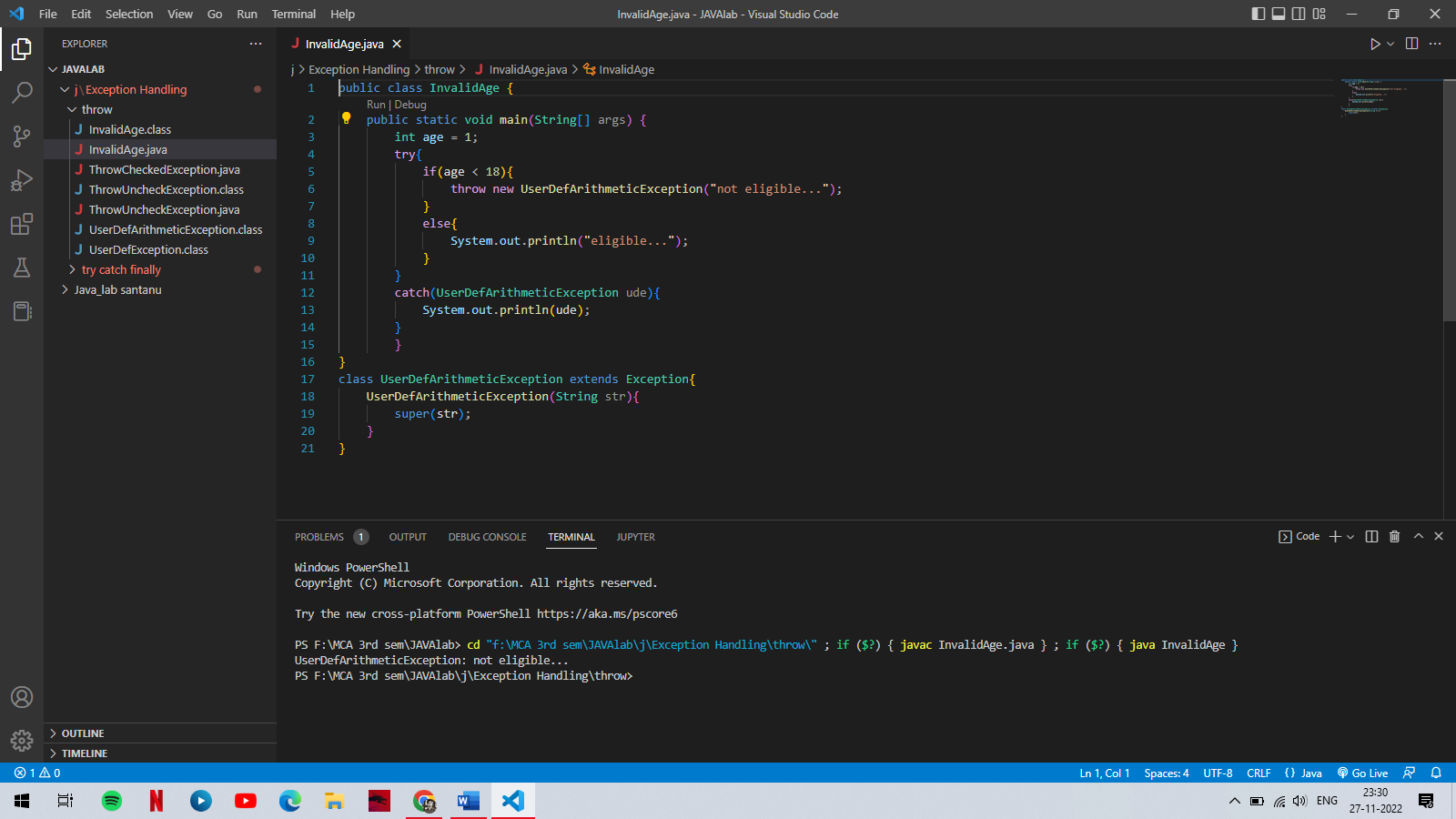
**or**

**throw new UserDefinedException(“Exception Details”);**

**Program:** throw checked exception.



O/P: -



1. Throws: -
2. A Throws clause is useful to throws an exception out of a method without handle it.
3. To throw any type of exception without handle it we can written as throws Exception.

**Syntax: -**

return\_type method\_name() throws Exception\_class\_name{

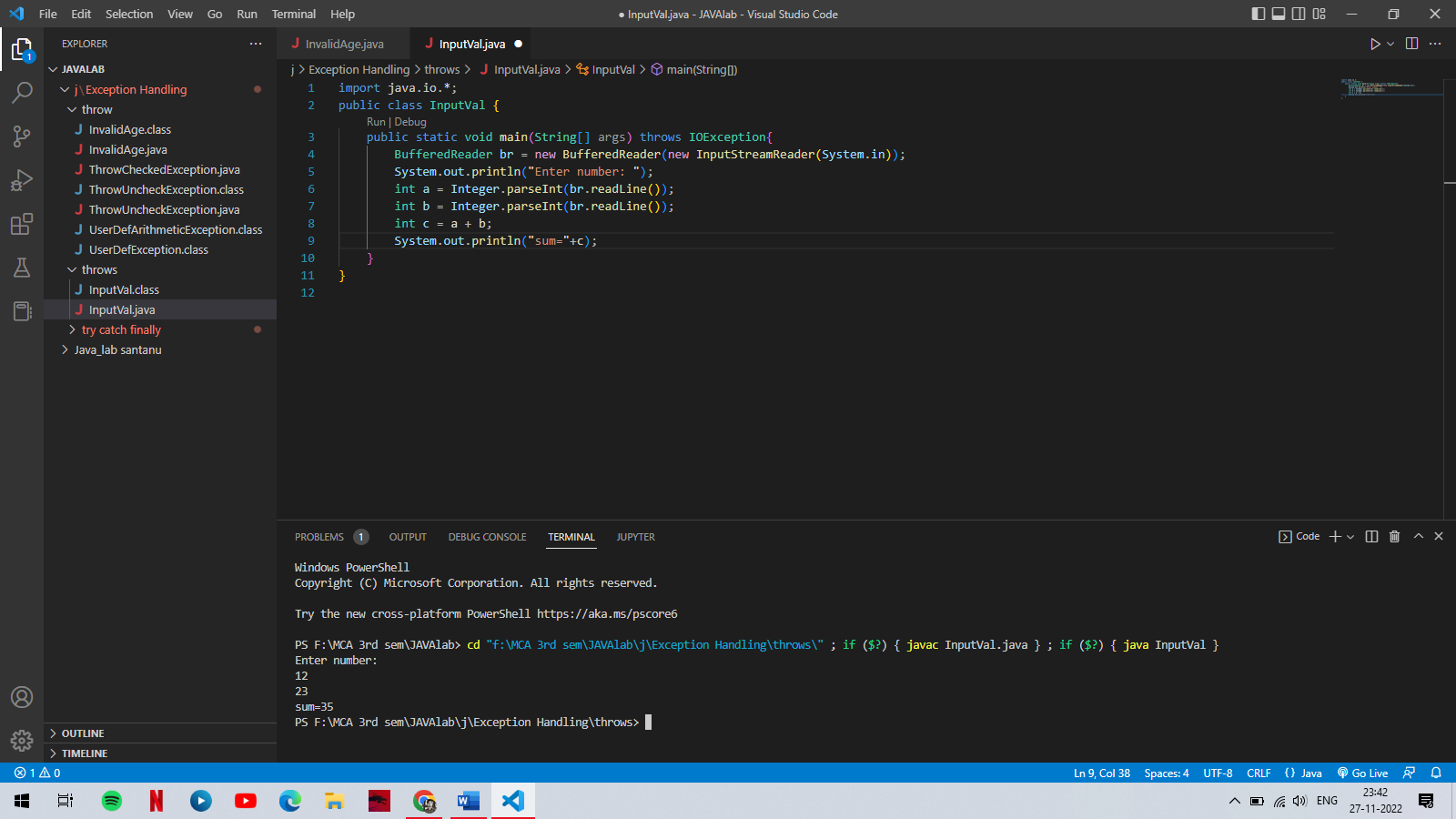
//method name

}

**Advantage of throws keyword:-**

1. Escape the exception because it is weaker exception handler.
2. Throws gives a warining message to the user, who is interested to handle the exception.

**Program:** throw checked exception.



O/P: -

