**Practical No. 23, 24 and 25: Develop a program for implementation of try, catch and finally block.**

1. **Practical Significance:**

Managing errors and Exception handing helps to detect exceptional conditions in a program and fix the exceptions as and when they occur. Student will be able to handle different types of exceptions using try, catch and finally blocks.

1. **Relevant Course Outcome:**

Implement Exception Handling.

1. **Practical Outcome:**

Develop a program for implementation of try, catch and finally block.

1. **Minimum Theoretical Background:**

**Errors:**

Errors are the mistakes that can make a program go wrong.

Error may produce a wrong results or abruptly terminates the execution of program or may cause the system to crash.

Detecting and managing the errors is very important during the program execution.

**Types of Errors:**

1. Compile-time errors
2. Run-time errors

**Exception Handling Tasks:**

1. Find the Problem (Hit the exception)
2. Inform that an error has occurred (Throw the exception)
3. Receive the error information (Catch the exception)
4. Take corrective action (Handle the exception)

**Common Java Exceptions:**

|  |  |  |
| --- | --- | --- |
| **Sr.**  **No** | **Exception Type** | **Causes of Exception** |
| 1 | ArithmeticException | Caused by math errors such as division by zero |
| 2 | ArrayIndexOutOfBoundsException | Caused by bad array indexes |
| 3 | ArrayStoreException | Caoused when a program tries to store the wrong type of data in an array |
| 4 | FileNotFoundException | Caused by an attempt to access a nonexistent file |
| 5 | IOException | Caused by general I/O failures, such as inability to read from a file |
| 6 | NullPointerException | Caused by referencing a null object |
| 7 | NumberFormatException | Caused when a conversion between string and number fails |
| 8 | OutOfMemoryException | Caused when there is not enough memory to allocate a new object |
| 9 | SecurityException | Caused when an applet tries to perform an action not allowed by the browser’s security setting. |
| 10 | StackOverflowException | Caused when the system runs out of stack space |
| 11 | StringIndexOutOfBoundsException | Caused when a program attempts to access a nonexistent character position in a string. |

**Categories of Exceptions:**

1. **Checked Exceptions**

Checked exceptions are explicitly handled in the code itself using try catch blocks.

These are extended from the java.lang.Exception class

1. **Unchecked Exceptions**

Unchecked exceptions are not necessarily handled in the program code, instead the JVM handles such exceptions.

These are extended from the java.lang.Runtime.Exception class.

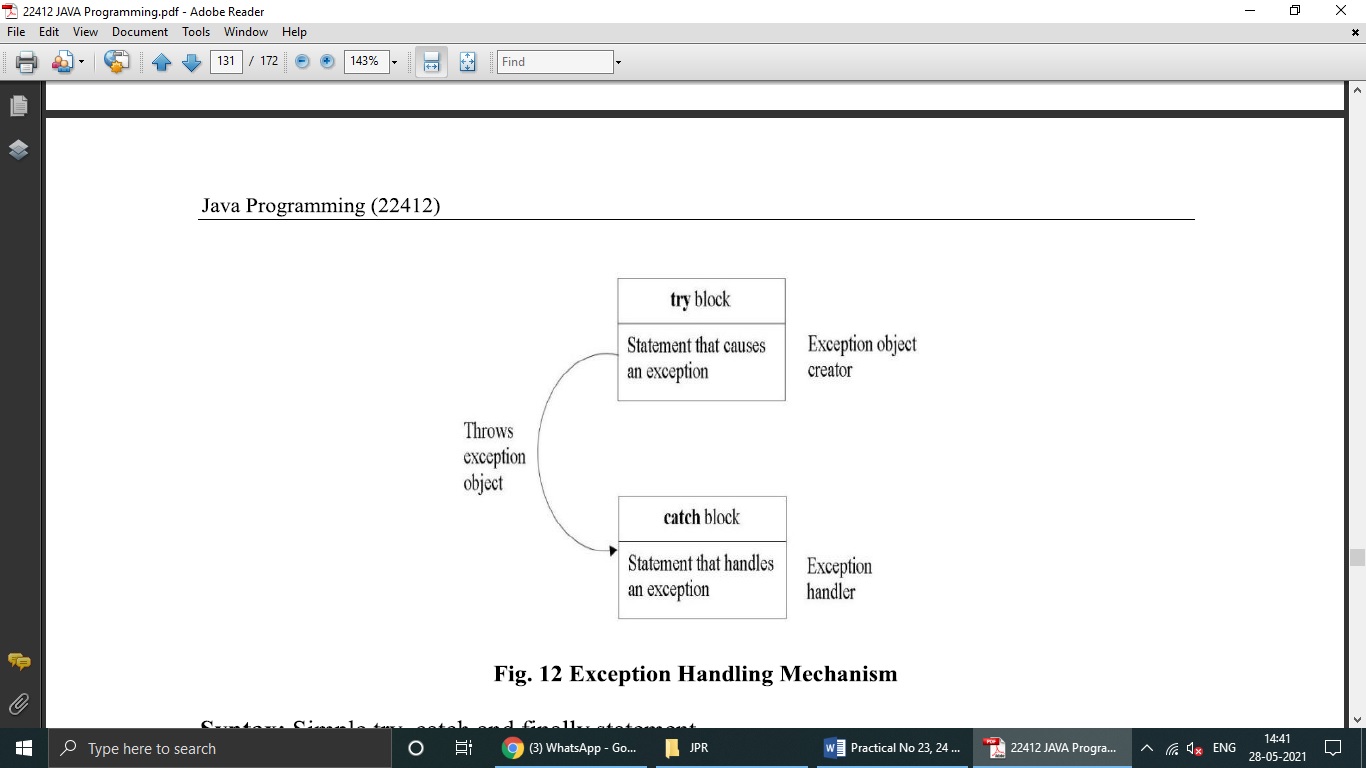
**Exception Handling Code:**

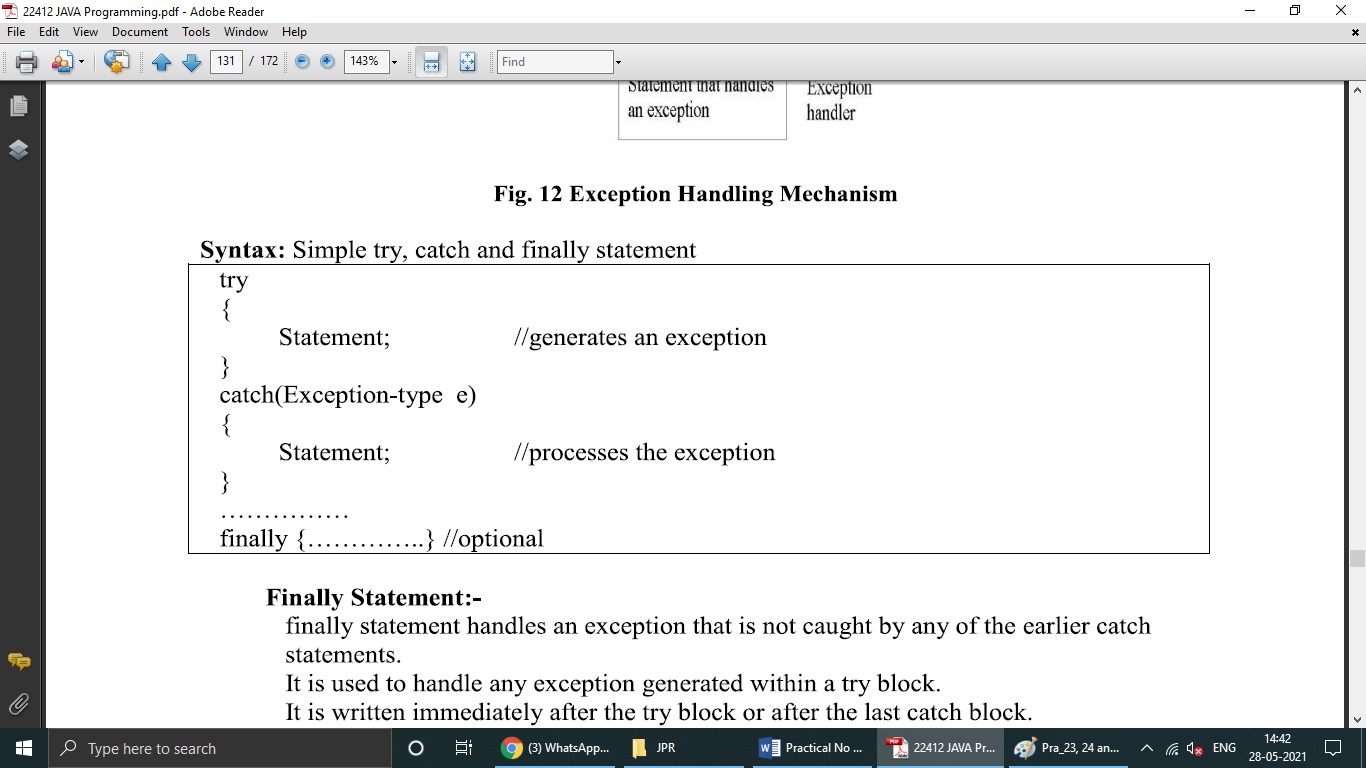
**Try:**

A keyword ‘try’ is used for a block of code that causes an error condition and ‘throw’ an exception.

**Catch:**

A keyword ‘catch’ is used for a block of code that ‘catches; the exception thrown by the ‘try’ block and handles it properly.





**Finally Statement:**

Finally statement handles an exception that is not caught by any of the earlier catch statements.

It is used to handle any exception generated within a try block.

It is written immediately after the try block or after the last catch block.

1. **Program Code:**

public class ExcepTest

{

public static void main(String[] args)

{

int a[] = new int[12];

try

{

System.out.println("Access elements three :"+a[3]);

}

catch (ArrayIndexOutOfBoundsException e)

{

System.out.println("Exception thrown"+e);

}

finally

{

System.out.println("First elements value:"+a[0]);

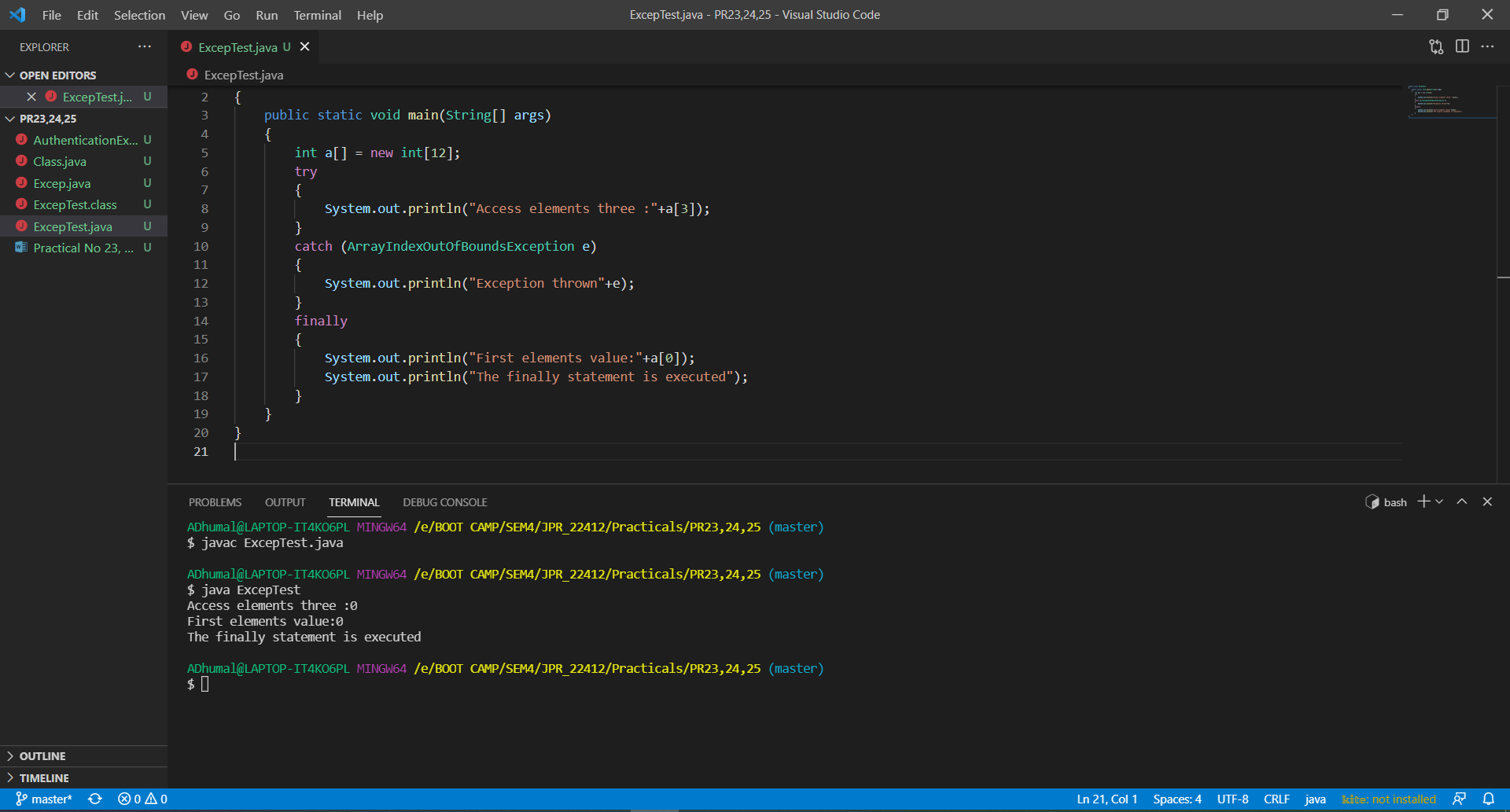
System.out.println("The finally statement is executed");

}

}

}

1. **Result:**



1. **Practical Related Questions:**
2. **How exception is thrown by main method?**

When exception is thrown by main() method, Java Runtime terminates the program and print the exception message and stack trace in system console. The throws clause only states that the method throws a checked FileNotFoundException and the calling method should catch or rethrow it.

1. **Differentiate between error and exception in java.**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Error** | **Exception** |
| 1 | Classified as an unchecked type | Classified as checked and unchecked |
| 2 | It belongs to java.lang.error | It belongs to java.lang.Exception |
| 3 | It is irrecoverable | It is recoverable |
| 4 | It can't be occur at compile time | It can occur at run time compile time both |
| 5 | OutOfMemoryError ,IOError | NullPointerException , SqlException |

1. **Can we throw exception manually? Illustrate with sample program.**

You can throw a user defined exception or, a predefined exception explicitly using the throw keyword. ... To throw an exception explicitly you need to instantiate the class of it and throw its object using the throw keyword.

1. **Explain the use of finally block?**

The finally block in java is used to put important codes such as clean up code e.g. closing the file or closing the connection. The finally block executes whether exception rise or not and whether exception handled or not. A finally contains all the crucial statements regardless of the exception occurs or not.

1. **Exercise:**
2. **The program calculates sum of two numbers inputted as command-line arguments. When will it give exception?**

public class Excep

{

public static void main(String[] args)

{

try

{

int n = Integer.parseInt(args[0]);

int n1 = Integer.parseInt(args[1]);

int n2 = n + n1;

System.out.println(" Sum is : " + n2);

}

catch (NumberFormatException ex)

{

System.out.println(ex);

}

finally

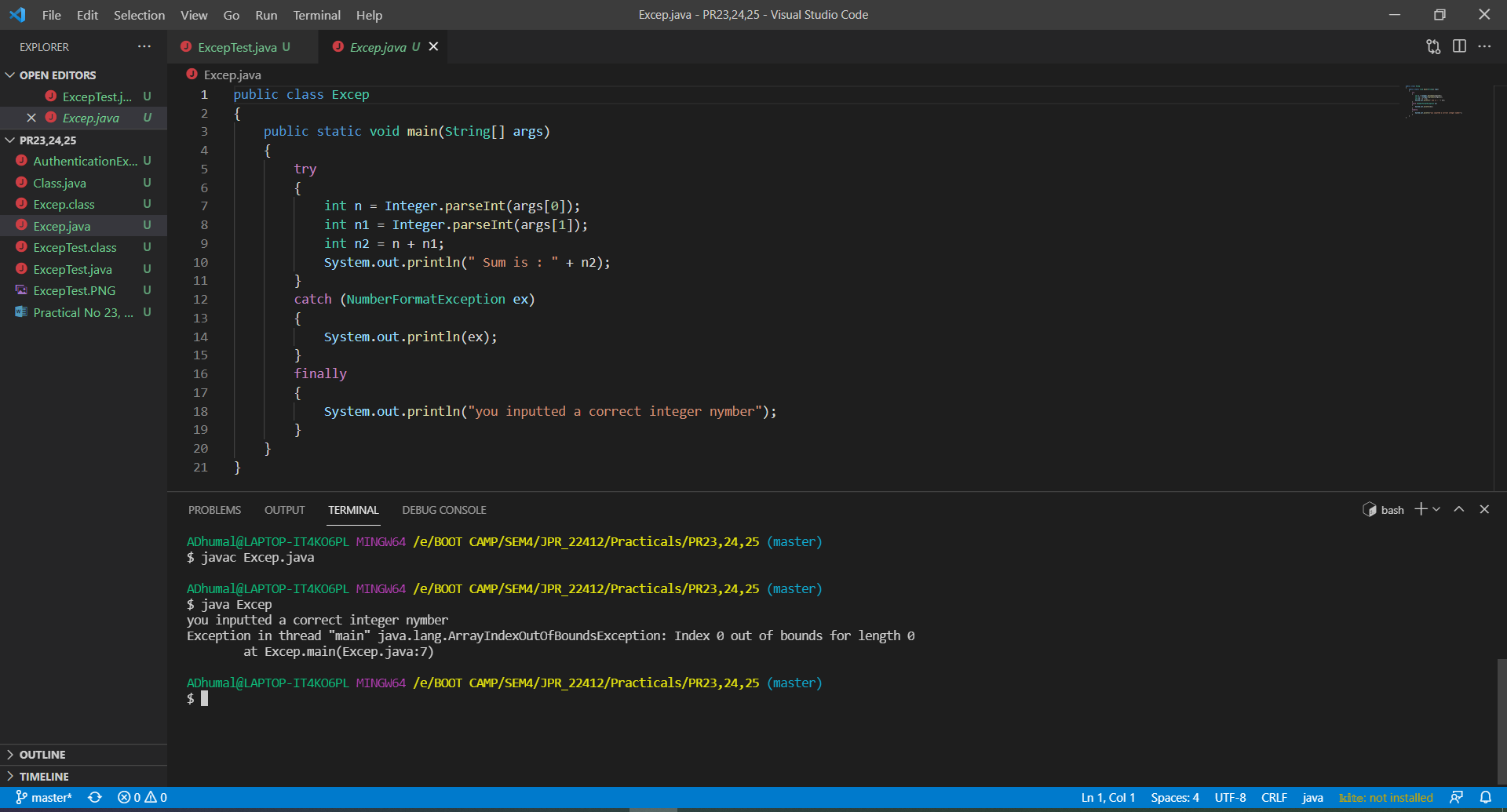
{

System.out.println("you inputted a correct integer nymber");

}

}

}



1. **Develop a program to accept a password from the user and throw “Authentication Failure” exception if the password is incorrect.**

import java.util.\*;

class AuthenticationException extends Exception

{

public AuthenticationException(String message)

{

super(message);

}

}

public class AuthenticationExDemo

{

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

String pwd;

try

{

System.out.println("Enter Password ::");

pwd = sc.nextLine();

if(!pwd.equals("123"))

throw new AuthenticationException("Incorrect password \n Type correct password");

else

System.out.println("Welcome User !!!");

}

catch (AuthenticationException a)

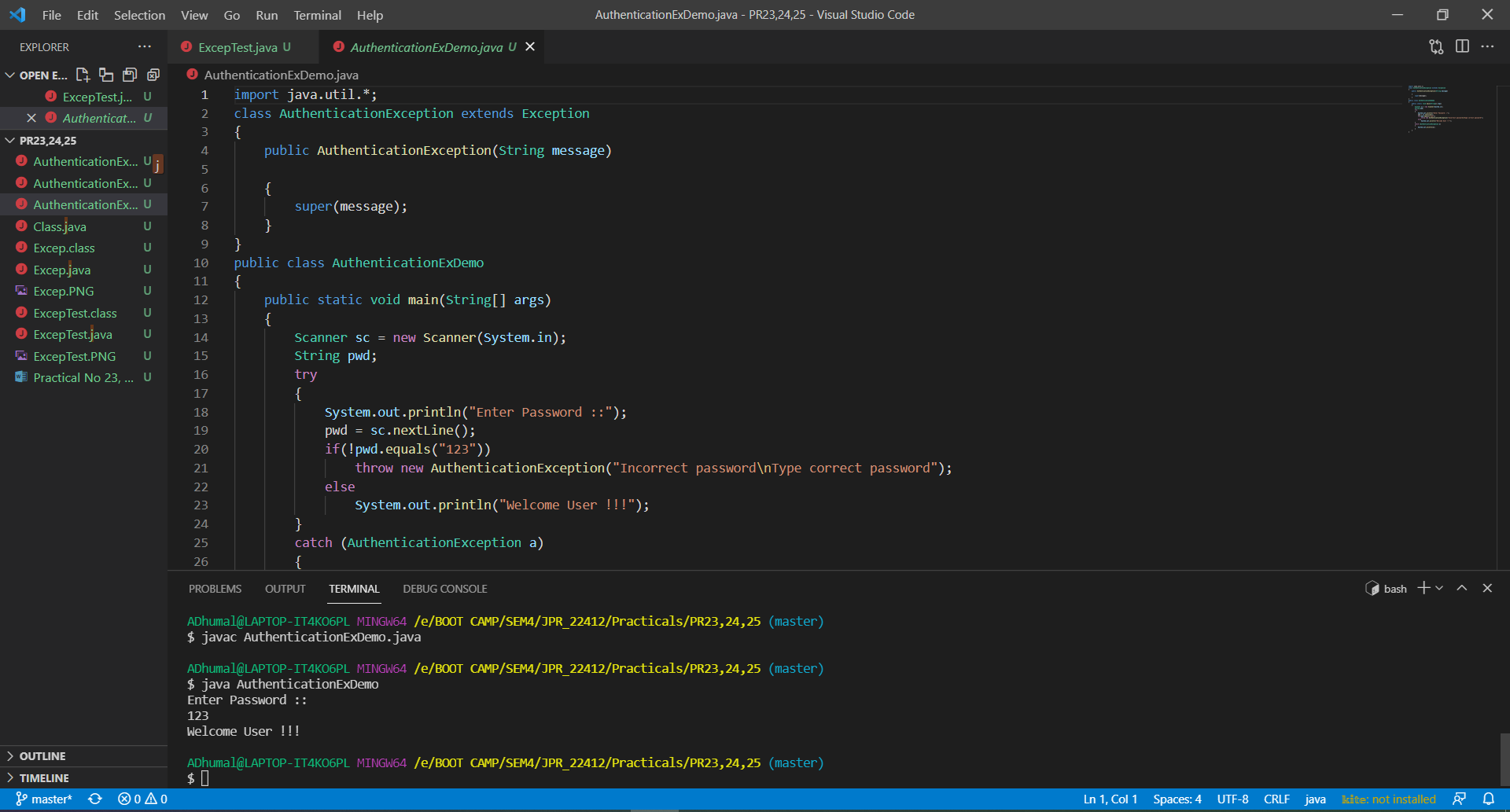
{

System.out.println(a);

}

}

}



1. **Write the exception thrown by the following code block?**

Integer[][] ints = {{1,2,3}, {null},{7,8,9}};

System.out.println(“value=”+ints[1][1].intValue());

public class Class

{

public static void main(String[] args)

{

Integer[][] ints = {{1,2,3}, {null},{7,8,9}};

System.out.println("value="+ints[1][1].intValue());

}

}

