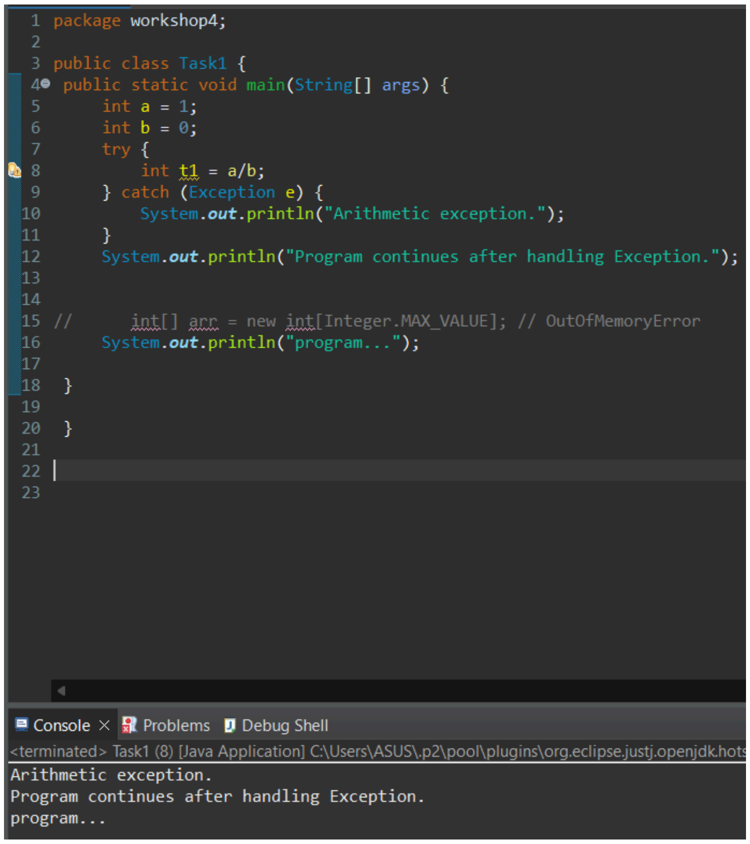
Workshop 4

Exception Handling

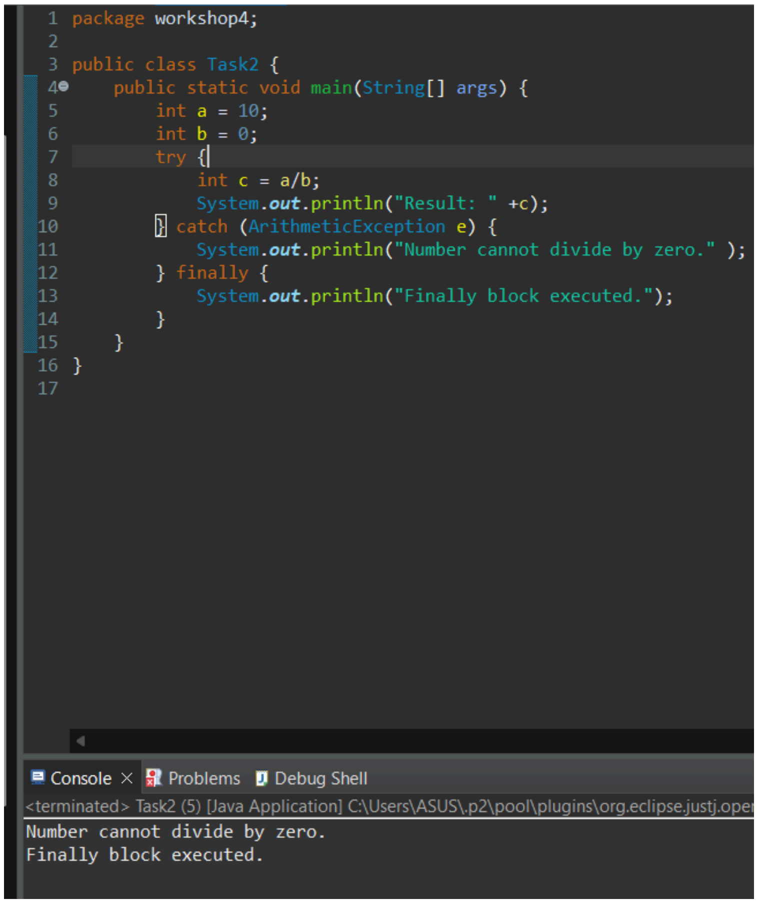
1. Write a Java program to demonstrate the difference between an Exception and an Error.



```java
package workshop4;

public class Task1 {
public static void main(String[] args) {
    int a = 1;
    int b = 0;
    try {
        int t1 = a/b;
    } catch (Exception e) {
        System.out.println("Arithmetic exception.");
    }
    System.out.println("Program continues after handling Exception.");


//      int[] arr = new int[Integer.MAX_VALUE]; // OutOfMemoryError
    System.out.println("program...");

 }

 }


```

```
<terminated> Task1 (8) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hots
Arithmetic exception.
Program continues after handling Exception.
program...
```

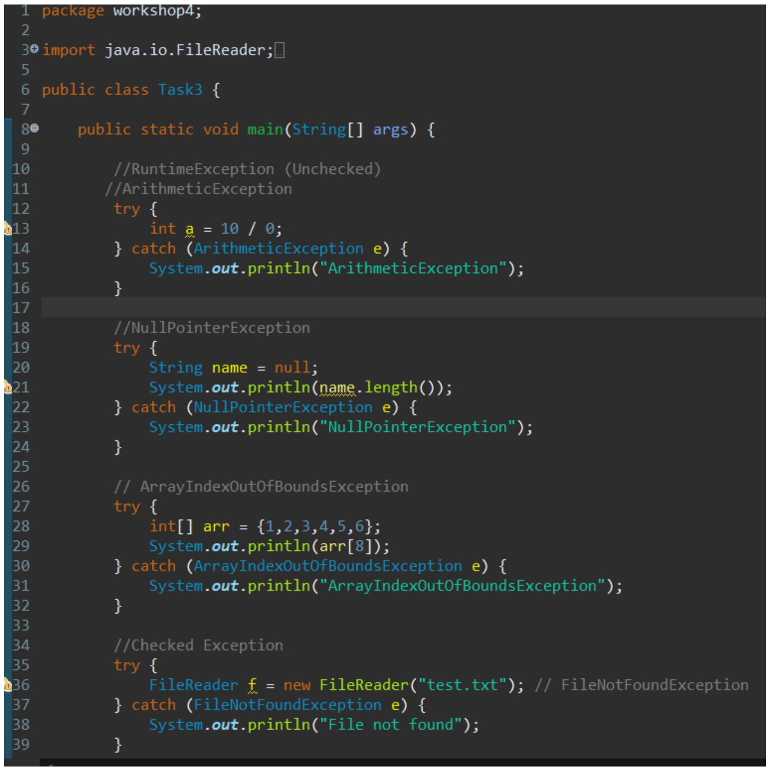
2. Create a program that demonstrates ArithmeticException by dividing a number by zero and handle it using try-catch.



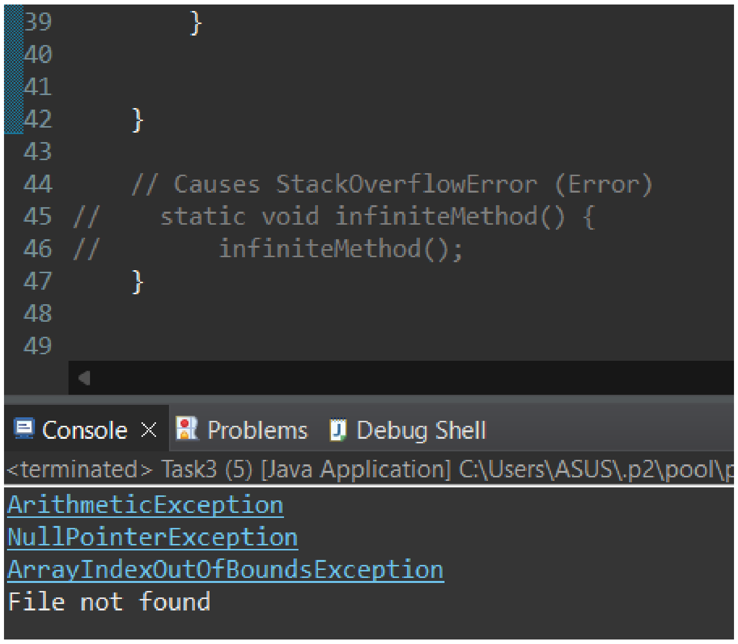
```java
package workshop4;

public class Task2 {
    public static void main(String[] args) {
        int a = 10;
        int b = 0;
        try {
            int c = a/b;
            System.out.println("Result: " +c);
        } catch (ArithmeticException e) {
            System.out.println("Number cannot divide by zero." );
        } finally {
            System.out.println("Finally block executed.");
        }
    }
}
```

```
<terminated> Task2 (5) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.oper
Number cannot divide by zero.
Finally block executed.
```

3. Write a Java program to show the complete Exception Hierarchy using comments and examples.



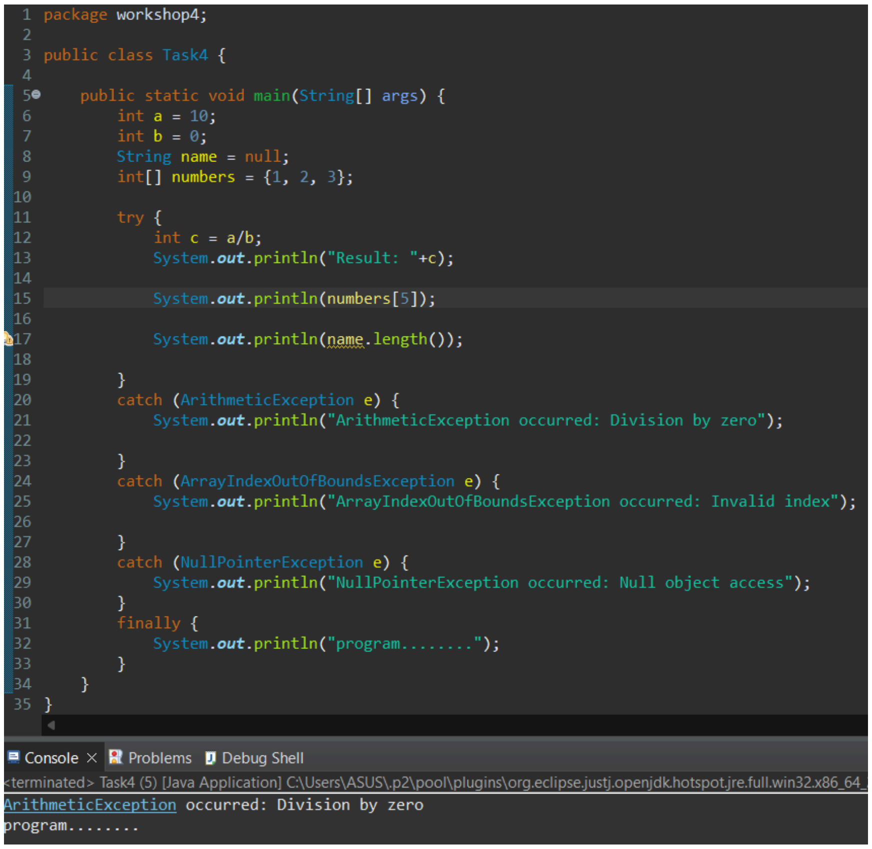
```java
package workshop4;

import java.io.FileReader;

public class Task3 {

    public static void main(String[] args) {

        //RuntimeException (Unchecked)
       //ArithmeticException
        try {
            int a = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("ArithmeticException");
        }

        //NullPointerException
        try {
            String name = null;
            System.out.println(name.length());
        } catch (NullPointerException e) {
            System.out.println("NullPointerException");
        }

        // ArrayIndexOutOfBoundsException
        try {
            int[] arr = {1,2,3,4,5,6};
            System.out.println(arr[8]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("ArrayIndexOutOfBoundsException");
        }

        //Checked Exception
        try {
            FileReader f = new FileReader("test.txt"); // FileNotFoundException
        } catch (FileNotFoundException e) {
            System.out.println("File not found");
        }
```



```java
        }


    }

    // Causes StackOverflowError (Error)
//     static void infiniteMethod() {
//         infiniteMethod();
    }


```

```
<terminated> Task3 (5) [Java Application] C:\Users\ASUS\.p2\pool\p
ArithmeticException
NullPointerException
ArrayIndexOutOfBoundsException
File not found
```

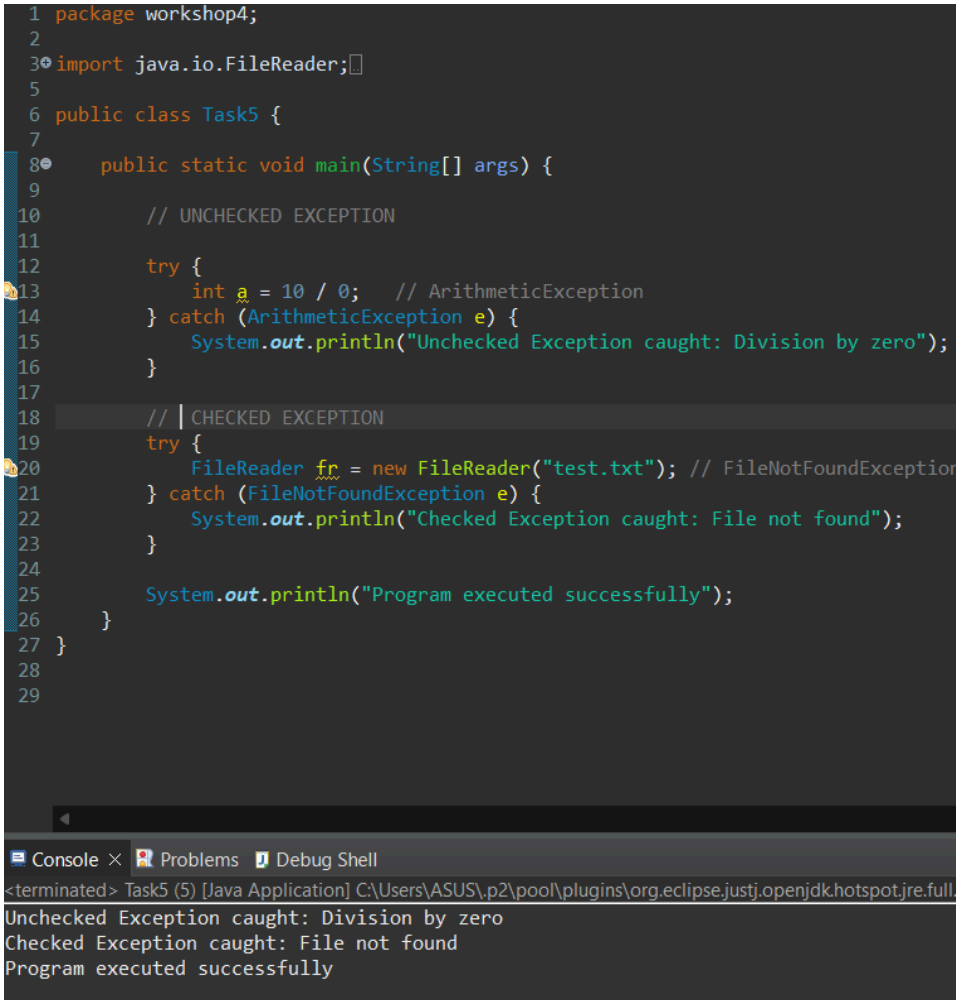
4. Develop a program to handle multiple types of exceptions (ArithmeticException, ArrayIndexOutOfBoundsException, and NullPointerException) in a single try block.



```java
package workshop4;

public class Task4 {

    public static void main(String[] args) {
        int a = 10;
        int b = 0;
        String name = null;
        int[] numbers = {1, 2, 3};

        try {
            int c = a/b;
            System.out.println("Result: "+c);

            System.out.println(numbers[5]);

            System.out.println(name.length());

        }
        catch (ArithmeticException e) {
            System.out.println("ArithmeticException occurred: Division by zero");

        }
        catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("ArrayIndexOutOfBoundsException occurred: Invalid index");

        }
        catch (NullPointerException e) {
            System.out.println("NullPointerException occurred: Null object access");
        }
        finally {
            System.out.println("program........");
        }
    }
}
```

```
<terminated> Task4 (5) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_
ArithmeticException occurred: Division by zero
program........
```

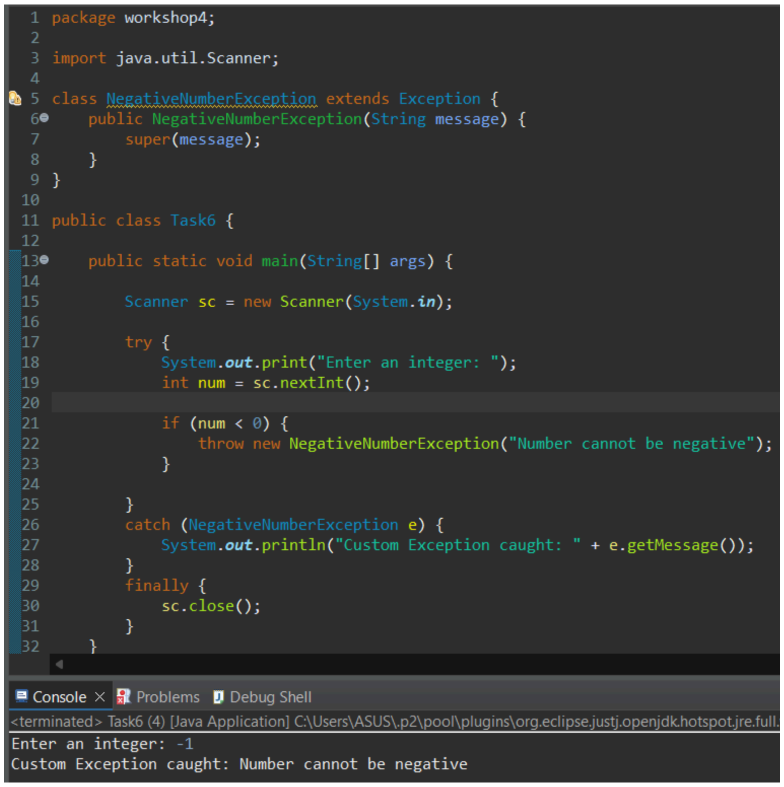
5. Write a program that demonstrates checked and unchecked exceptions with suitable examples.



```java
package workshop4;

import java.io.FileReader;

public class Task5 {

    public static void main(String[] args) {

        // UNCHECKED EXCEPTION

        try {
            int a = 10 / 0;    // ArithmeticException
        } catch (ArithmeticException e) {
            System.out.println("Unchecked Exception caught: Division by zero");
        }

        // CHECKED EXCEPTION
        try {
            FileReader fr = new FileReader("test.txt"); // FileNotFoundExceptio
        } catch (FileNotFoundException e) {
            System.out.println("Checked Exception caught: File not found");
        }

        System.out.println("Program executed successfully");
    }
}
```

```
<terminated> Task5 (5) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.
Unchecked Exception caught: Division by zero
Checked Exception caught: File not found
Program executed successfully
```

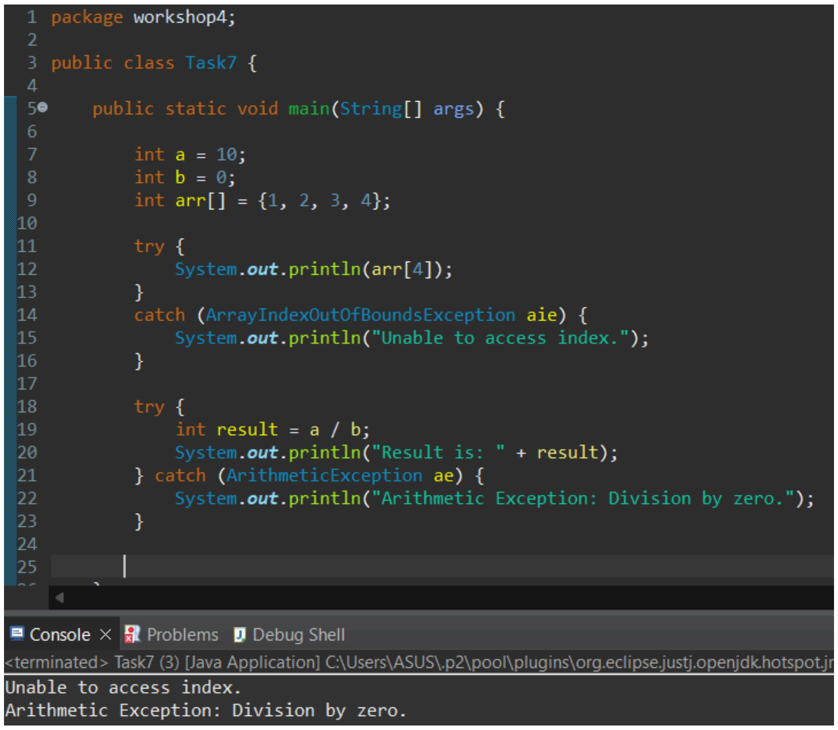
6. Create a program that reads an integer from the user and throws a custom exception if the number is negative.



```java
package workshop4;

import java.util.Scanner;

class NegativeNumberException extends Exception {
    public NegativeNumberException(String message) {
        super(message);
    }
}

public class Task6 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        try {
            System.out.print("Enter an integer: ");
            int num = sc.nextInt();

            if (num < 0) {
                throw new NegativeNumberException("Number cannot be negative");
            }

        }
        catch (NegativeNumberException e) {
            System.out.println("Custom Exception caught: " + e.getMessage());
        }
        finally {
            sc.close();
        }
    }
```

```
<terminated> Task6 (4) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.
Enter an integer: -1
Custom Exception caught: Number cannot be negative
```

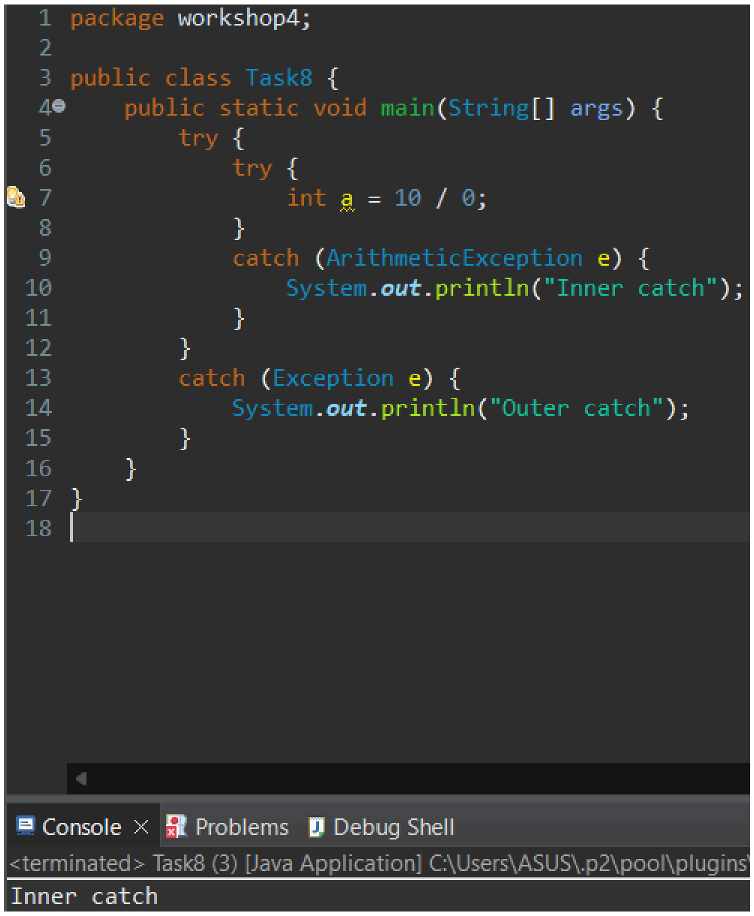
7. Write a Java program using multiple catch blocks to handle different exceptions separately.



```java
package workshop4;

public class Task7 {

    public static void main(String[] args) {

        int a = 10;
        int b = 0;
        int arr[] = {1, 2, 3, 4};

        try {
            System.out.println(arr[4]);
        }
        catch (ArrayIndexOutOfBoundsException aie) {
            System.out.println("Unable to access index.");
        }

        try {
            int result = a / b;
            System.out.println("Result is: " + result);
        } catch (ArithmeticException ae) {
            System.out.println("Arithmetic Exception: Division by zero.");
        }

```

```
<terminated> Task7 (3) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jr
Unable to access index.
Arithmetic Exception: Division by zero.
```

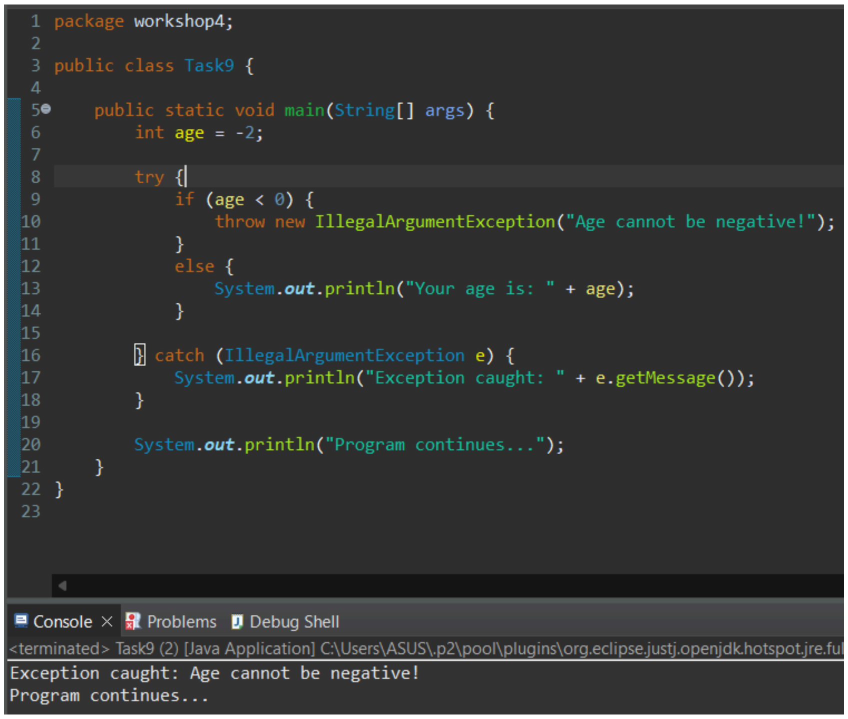
8. Create a program demonstrating nested try-catch blocks.



```java
package workshop4;

public class Task8 {
    public static void main(String[] args) {
        try {
            try {
                int a = 10 / 0;
            }
            catch (ArithmeticException e) {
                System.out.println("Inner catch");
            }
        }
        catch (Exception e) {
            System.out.println("Outer catch");
        }
    }
}

```

```
<terminated> Task8 (3) [Java Application] C:\Users\ASUS\.p2\pool\plugins\
Inner catch
```

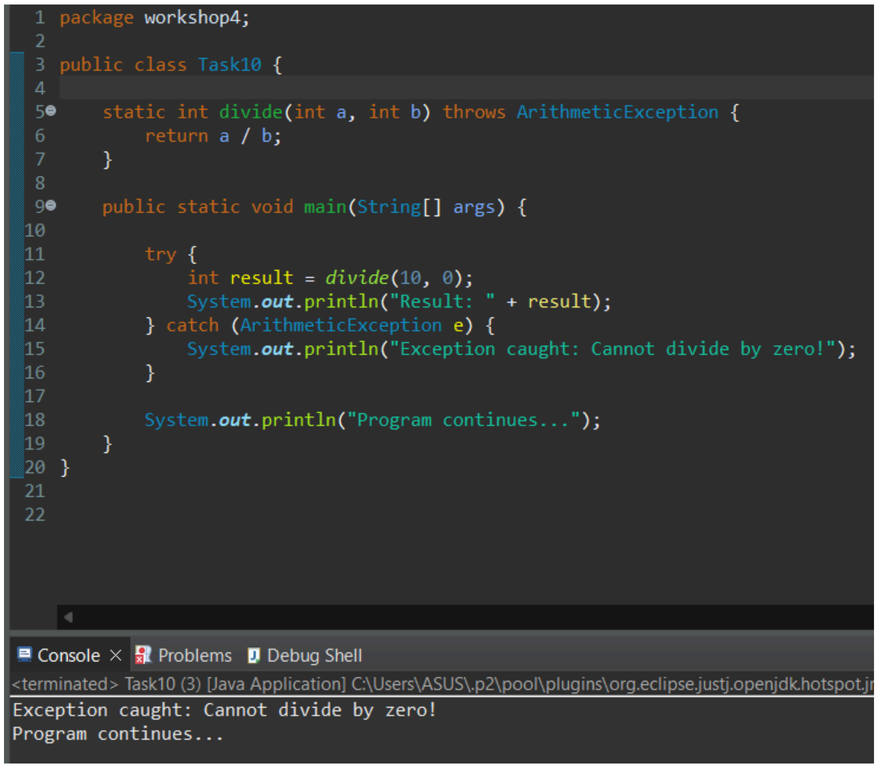
9. Write a Java program that uses the throw keyword to manually throw an exception when invalid input is entered.



```java
package workshop4;

public class Task9 {

    public static void main(String[] args) {
        int age = -2;

        try {
            if (age < 0) {
                throw new IllegalArgumentException("Age cannot be negative!");
            }
            else {
                System.out.println("Your age is: " + age);
            }

        } catch (IllegalArgumentException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }

        System.out.println("Program continues...");
    }
}

```

```
<terminated> Task9 (2) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.ful
Exception caught: Age cannot be negative!
Program continues...
```

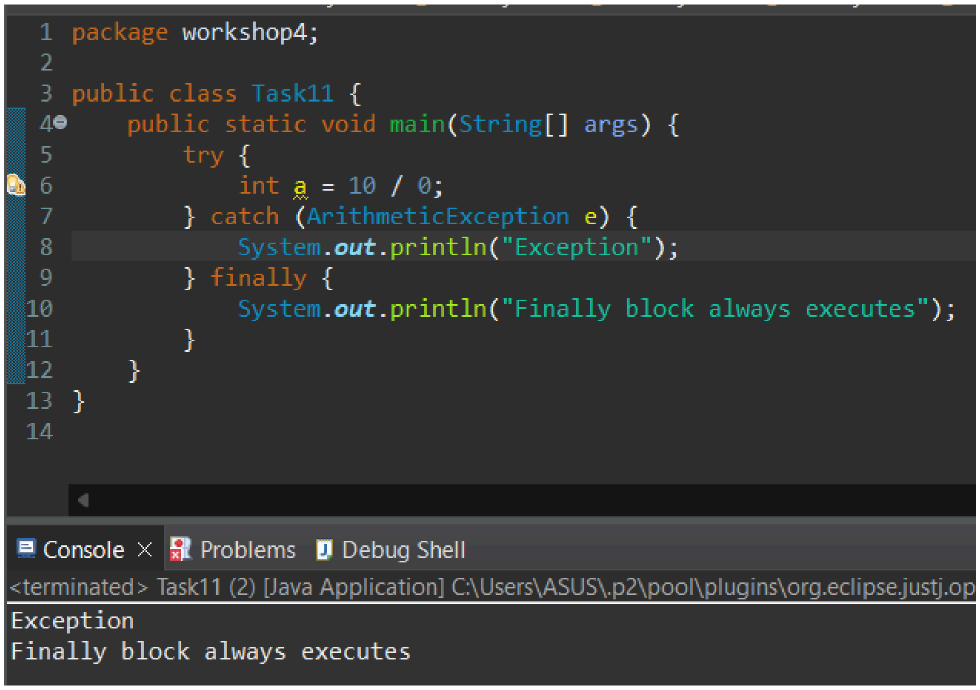
10. Develop a Java program that uses the throws keyword in a method and handle the exception in the calling method.



```java
package workshop4;

public class Task10 {

    static int divide(int a, int b) throws ArithmeticException {
        return a / b;
    }

    public static void main(String[] args) {

        try {
            int result = divide(10, 0);
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Exception caught: Cannot divide by zero!");
        }

        System.out.println("Program continues...");
    }
}

```

```
<terminated> Task10 (3) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jr
Exception caught: Cannot divide by zero!
Program continues...
```

11. Write a Java program to demonstrate the use of the finally block in exception handling.

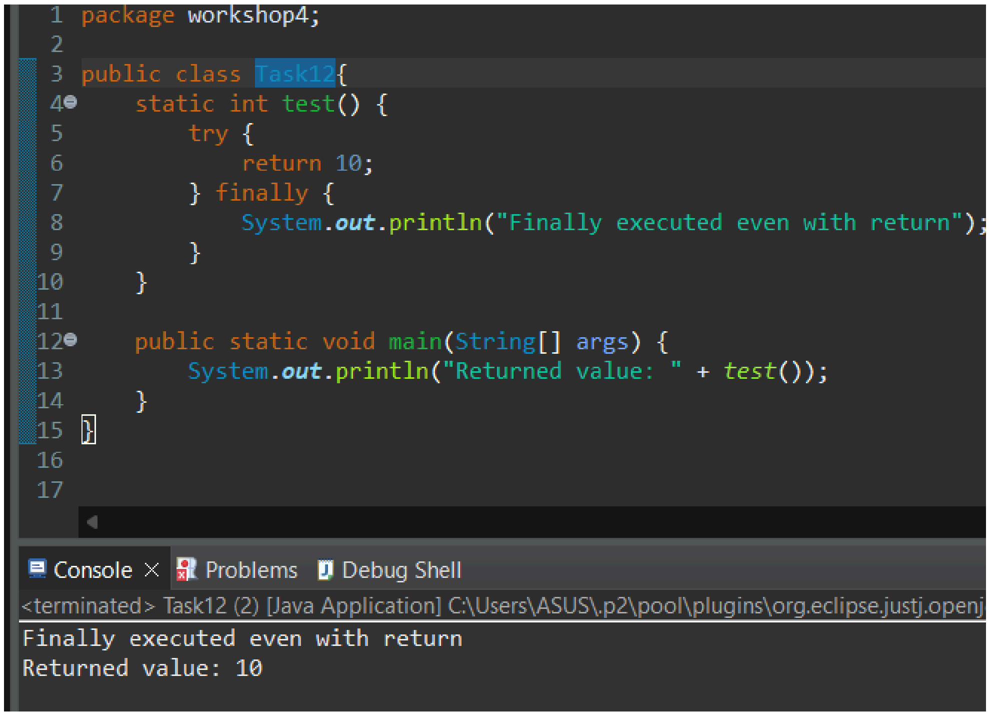


```java
package workshop4;

public class Task11 {
    public static void main(String[] args) {
        try {
            int a = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Exception");
        } finally {
            System.out.println("Finally block always executes");
        }
    }
}

```

Console
Problems
Debug Shell
<terminated> Task11 (2) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.op

```
Exception
Finally block always executes
```

12. Create a program to show that the finally block executes even when return is used inside a try or catch block.

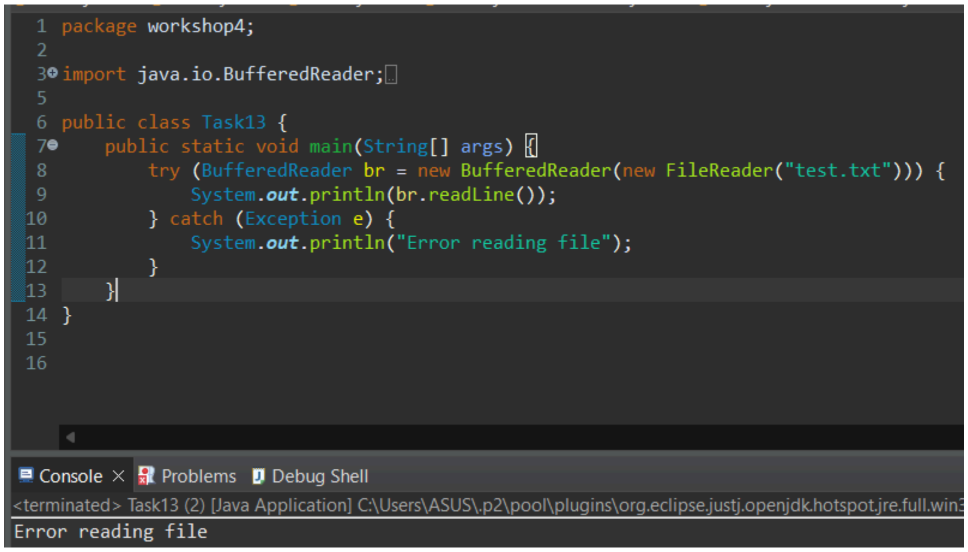


```java
package workshop4;

public class Task12{
    static int test() {
        try {
            return 10;
        } finally {
            System.out.println("Finally executed even with return");
        }
    }

    public static void main(String[] args) {
        System.out.println("Returned value: " + test());
    }
}


```

Console
Problems
Debug Shell
<terminated> Task12 (2) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openj

```
Finally executed even with return
Returned value: 10
```

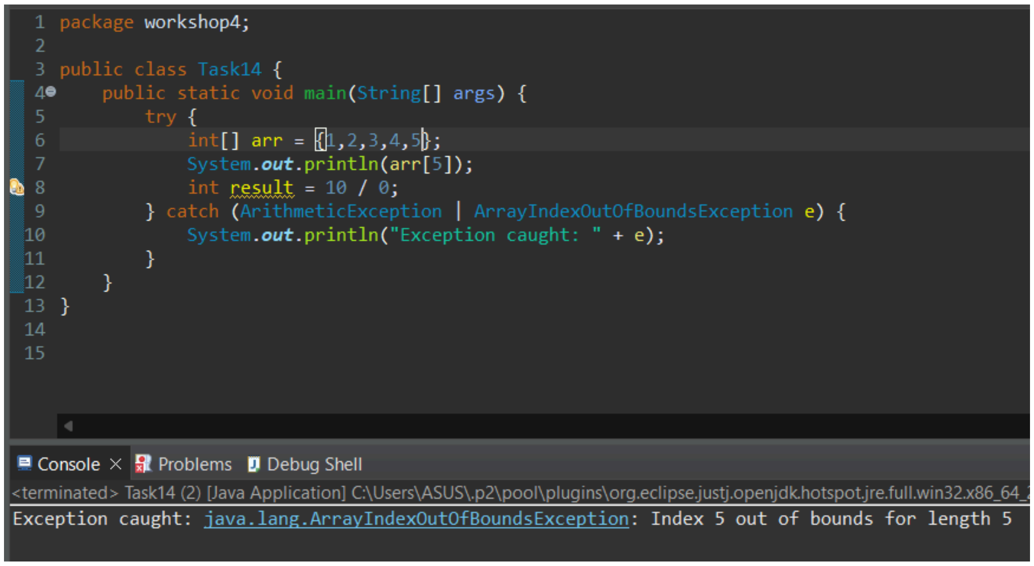
13. Write a program to demonstrate try-with-resources by reading data from a file safely.



```java
package workshop4;

import java.io.BufferedReader;

public class Task13 {
    public static void main(String[] args) {
        try (BufferedReader br = new BufferedReader(new FileReader("test.txt"))) {
            System.out.println(br.readLine());
        } catch (Exception e) {
            System.out.println("Error reading file");
        }
    }
}
```

```
<terminated> Task13 (2) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win3
Error reading file
```

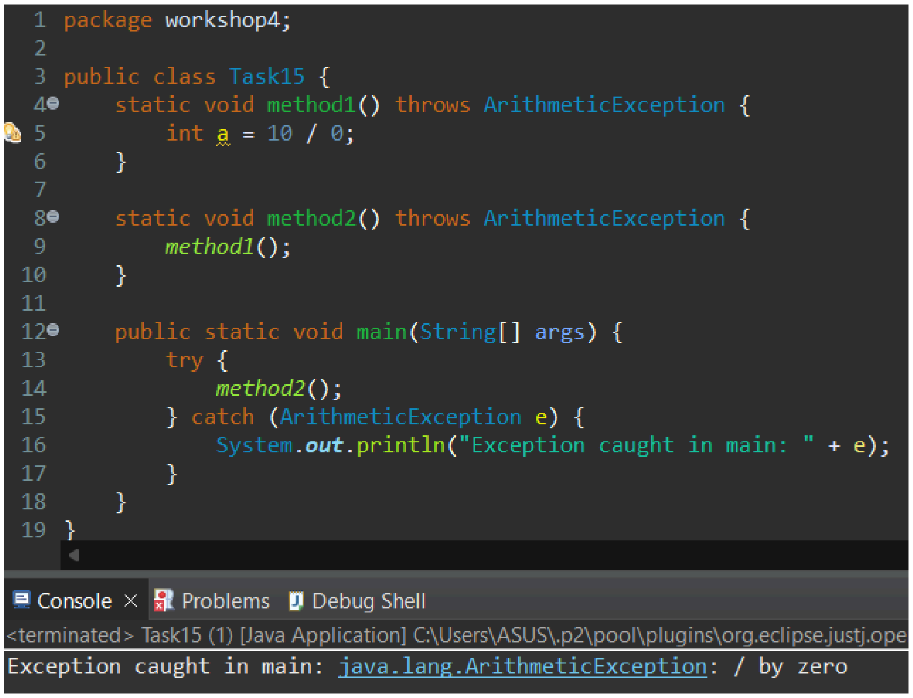
14. Develop a Java program to handle multiple exceptions using multi-catch syntax (catch(Exception1 | Exception2 e)).



```java
package workshop4;

public class Task14 {
    public static void main(String[] args) {
        try {
            int[] arr = {1,2,3,4,5};
            System.out.println(arr[5]);
            int result = 10 / 0;
        } catch (ArithmeticException | ArrayIndexOutOfBoundsException e) {
            System.out.println("Exception caught: " + e);
        }
    }
}
```

```
<terminated> Task14 (2) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_
Exception caught: java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5
```

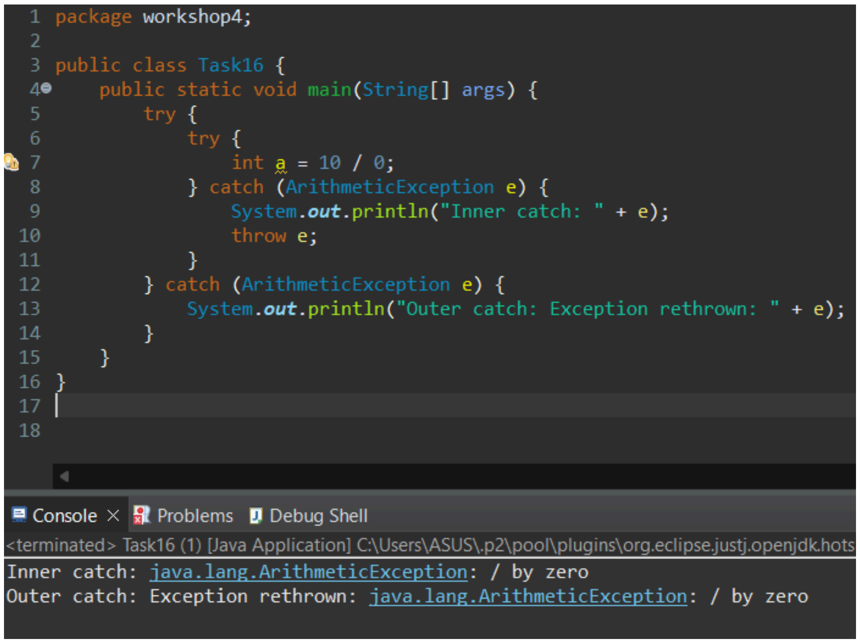
15. Write a program that shows how an exception is propagated from one method to another using throws.



```java
package workshop4;

public class Task15 {
    static void method1() throws ArithmeticException {
        int a = 10 / 0;
    }

    static void method2() throws ArithmeticException {
        method1();
    }

    public static void main(String[] args) {
        try {
            method2();
        } catch (ArithmeticException e) {
            System.out.println("Exception caught in main: " + e);
        }
    }
}
```

```
<terminated> Task15 (1) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.ope
Exception caught in main: java.lang.ArithmeticException: / by zero
```

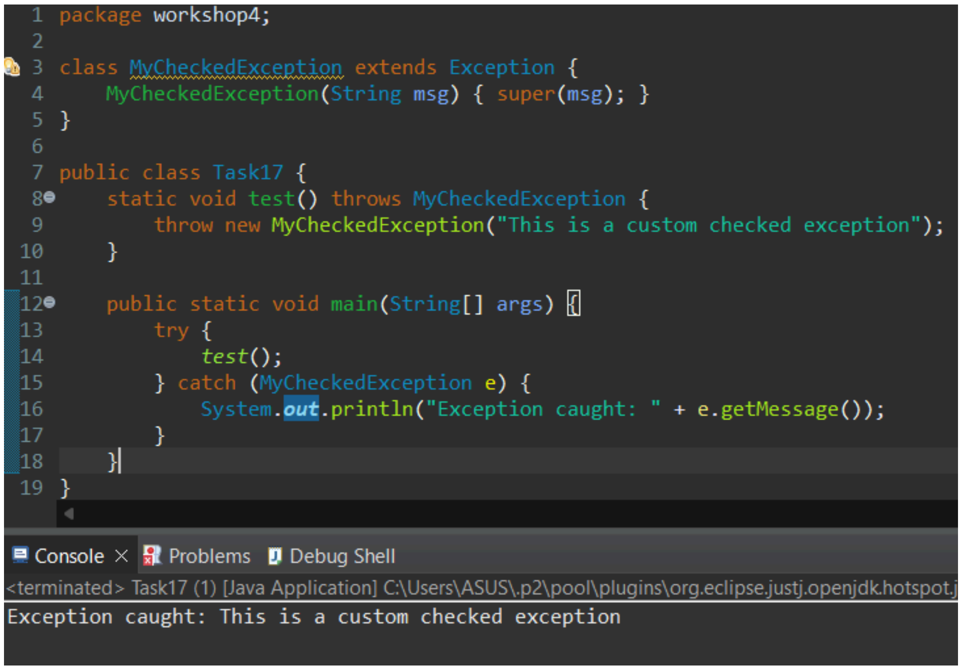
16. Create a program to demonstrate the concept of rethrowing an exception.



```java
package workshop4;

public class Task16 {
    public static void main(String[] args) {
        try {
            try {
                int a = 10 / 0;
            } catch (ArithmeticException e) {
                System.out.println("Inner catch: " + e);
                throw e;
            }
        } catch (ArithmeticException e) {
            System.out.println("Outer catch: Exception rethrown: " + e);
        }
    }
}

```

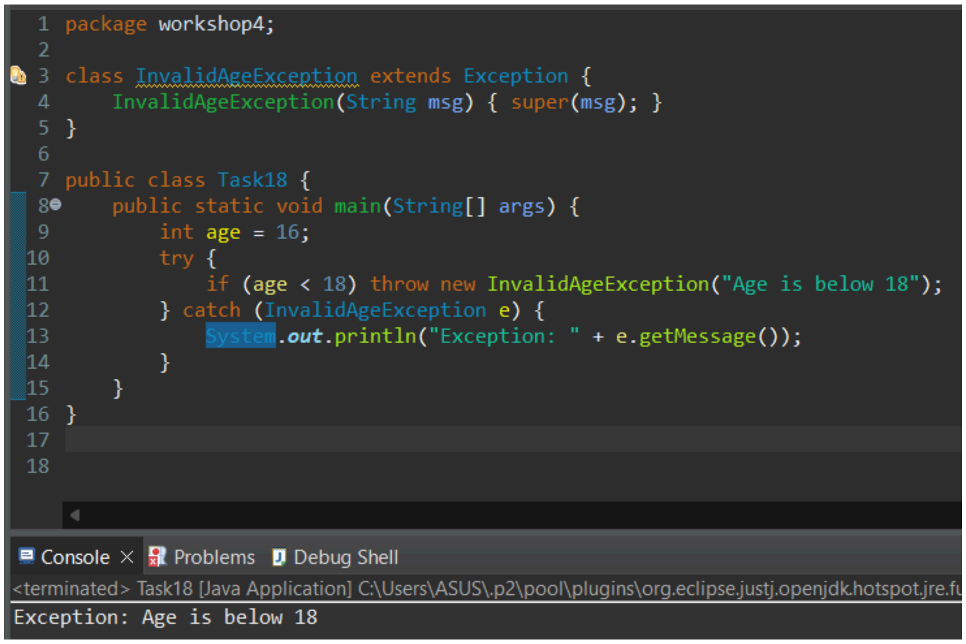
```
<terminated> Task16 (1) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hots
Inner catch: java.lang.ArithmeticException: / by zero
Outer catch: Exception rethrown: java.lang.ArithmeticException: / by zero
```

17. Write a Java program to demonstrate the difference between checked and unchecked exceptions by creating your own custom checked exception.


```java
package workshop4;

class MyCheckedException extends Exception {
    MyCheckedException(String msg) { super(msg); }
}

public class Task17 {
    static void test() throws MyCheckedException {
        throw new MyCheckedException("This is a custom checked exception");
    }

    public static void main(String[] args) {
        try {
            test();
        } catch (MyCheckedException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
    }
}
```

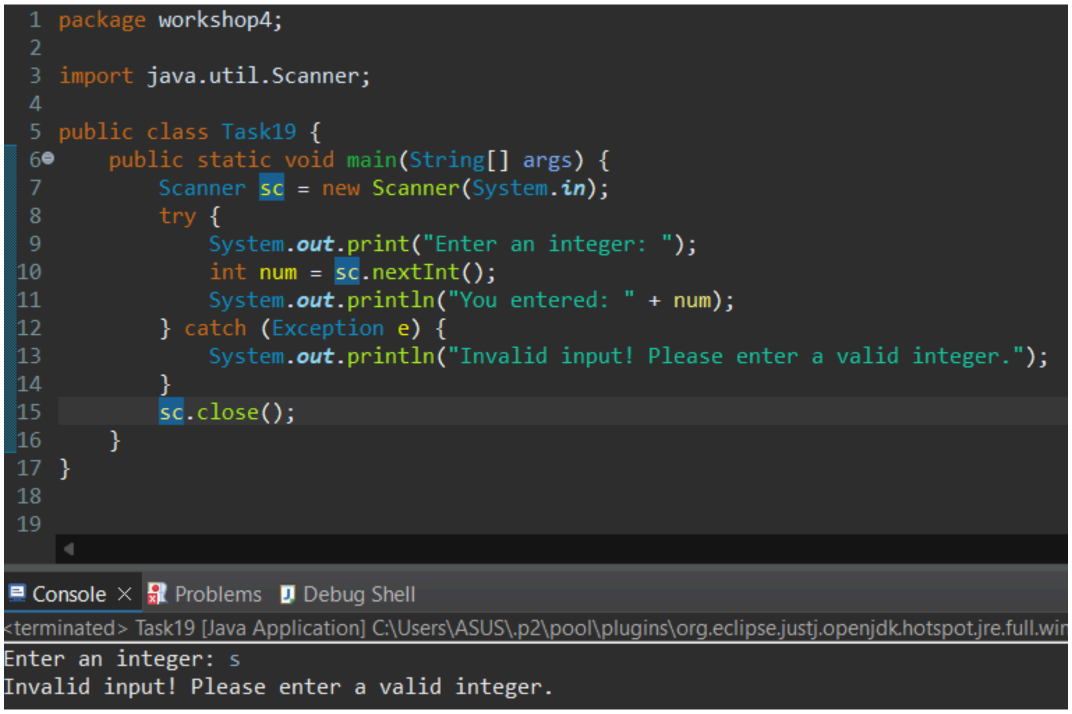
```
<terminated> Task17 (1) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.j
Exception caught: This is a custom checked exception
```

18. Create a class InvalidAgeException and write a program that throws this exception if a person's age is below 18.


```java
package workshop4;

class InvalidAgeException extends Exception {
    InvalidAgeException(String msg) { super(msg); }
}

public class Task18 {
    public static void main(String[] args) {
        int age = 16;
        try {
            if (age < 18) throw new InvalidAgeException("Age is below 18");
        } catch (InvalidAgeException e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
```

```
<terminated> Task18 [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.fu
Exception: Age is below 18
```

19. Write a Java program to handle user input errors gracefully using exception handling.

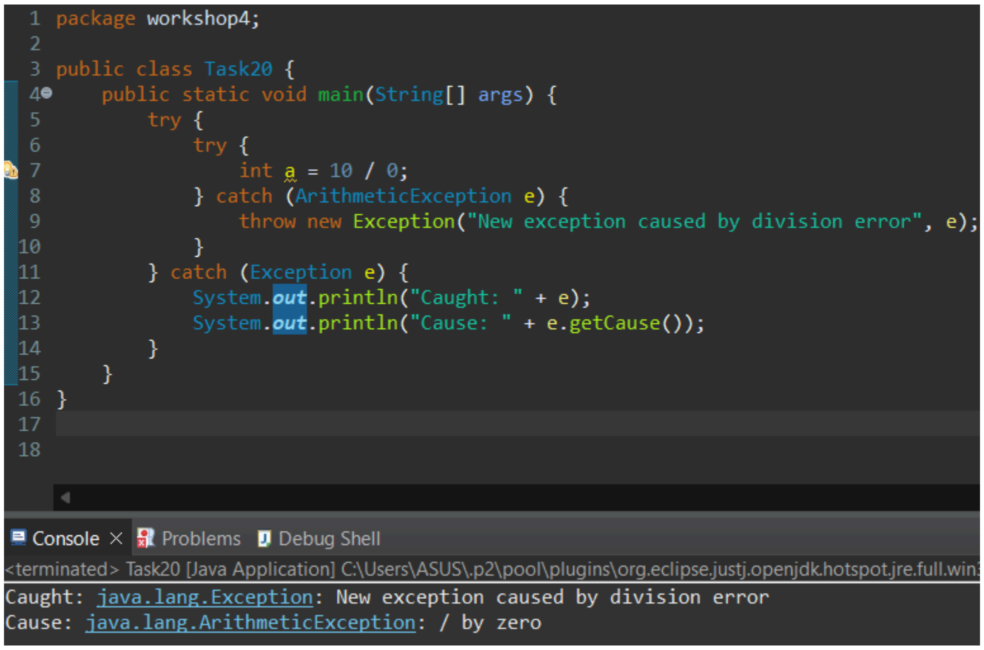


```java
package workshop4;

import java.util.Scanner;

public class Task19 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        try {
            System.out.print("Enter an integer: ");
            int num = sc.nextInt();
            System.out.println("You entered: " + num);
        } catch (Exception e) {
            System.out.println("Invalid input! Please enter a valid integer.");
        }
        sc.close();
    }
}
```

<terminated> Task19 [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.wir

```
Enter an integer: s
Invalid input! Please enter a valid integer.
```

20. Develop a Java program to demonstrate exception chaining (new Exception("msg", cause)).

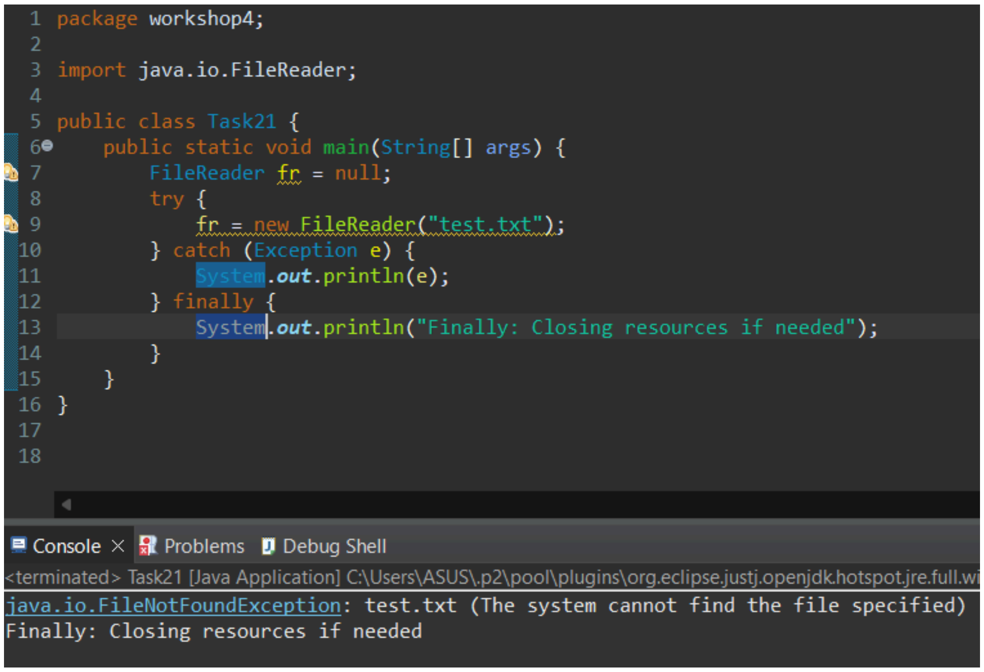


```java
package workshop4;

public class Task20 {
    public static void main(String[] args) {
        try {
            try {
                int a = 10 / 0;
            } catch (ArithmeticException e) {
                throw new Exception("New exception caused by division error", e);
            }
        } catch (Exception e) {
            System.out.println("Caught: " + e);
            System.out.println("Cause: " + e.getCause());
        }
    }
}
```

<terminated> Task20 [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win

```
Caught: java.lang.Exception: New exception caused by division error
Cause: java.lang.ArithmeticException: / by zero
```

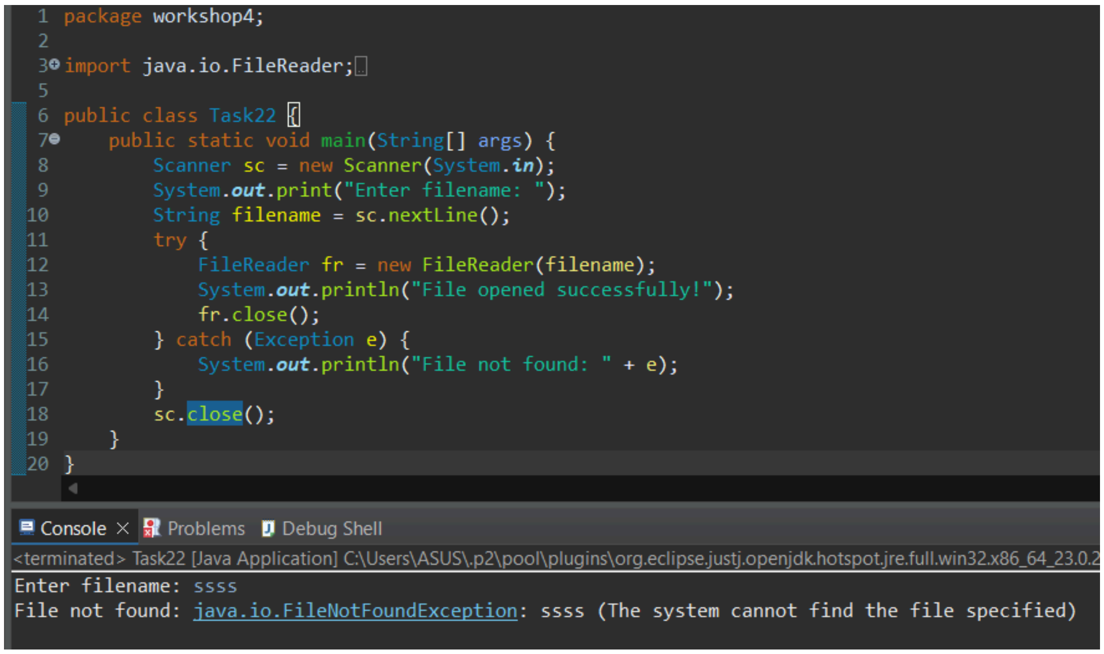
21. Create a program to demonstrate how finally is used for closing resources like files or database connections.



```java
package workshop4;

import java.io.FileReader;

public class Task21 {
    public static void main(String[] args) {
        FileReader fr = null;
        try {
            fr = new FileReader("test.txt");
        } catch (Exception e) {
            System.out.println(e);
        } finally {
            System.out.println("Finally: Closing resources if needed");
        }
    }
}
```

```
<terminated> Task21 [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.wi
java.io.FileNotFoundException: test.txt (The system cannot find the file specified)
Finally: Closing resources if needed
```

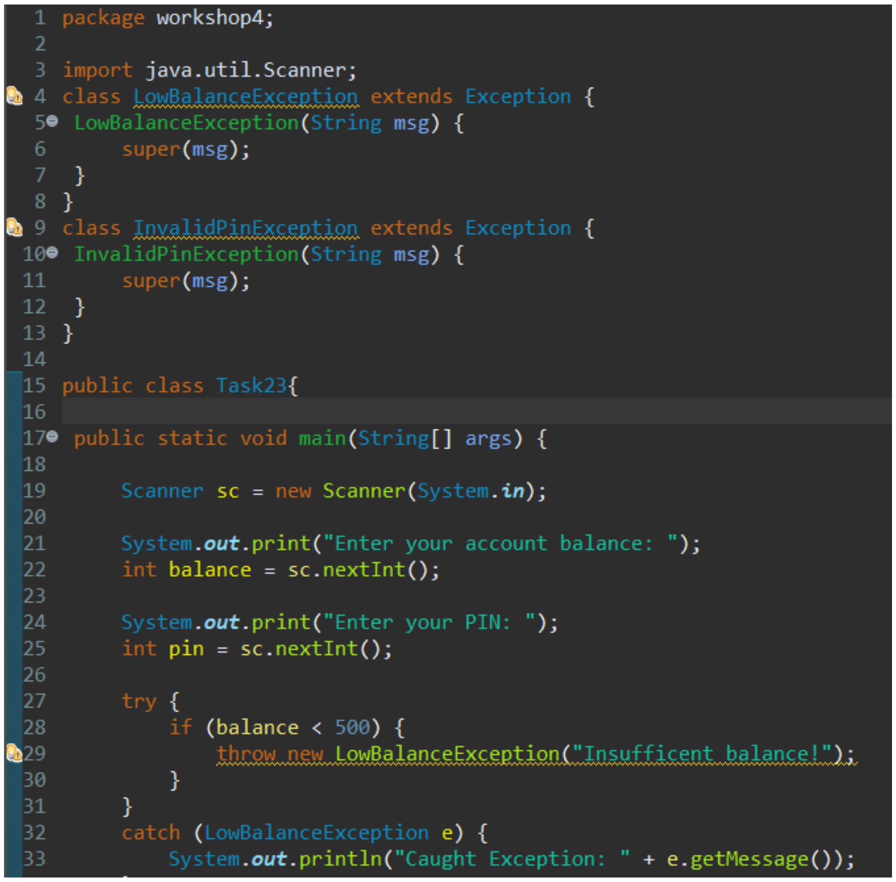
22. Write a program that takes a filename from the user, reads its content, and handles FileNotFoundException.



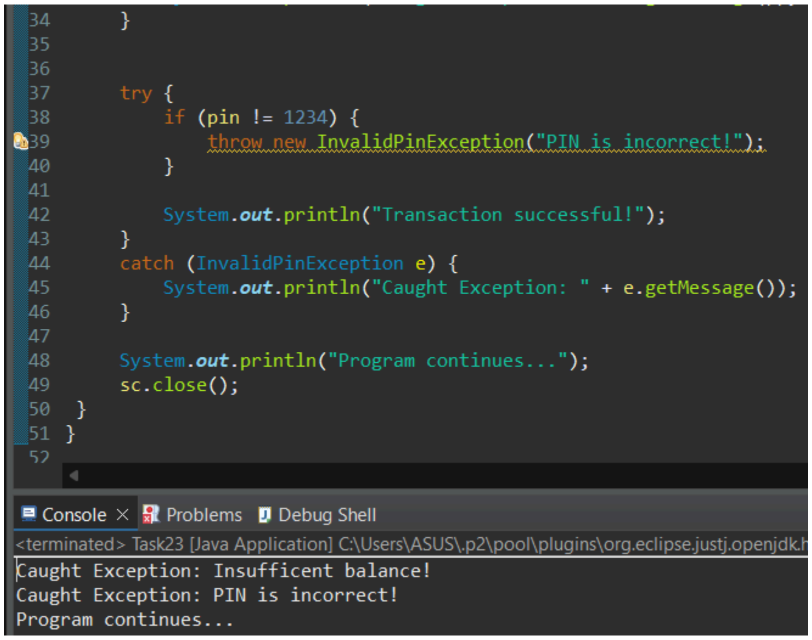
```java
package workshop4;

import java.io.FileReader;

public class Task22 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter filename: ");
        String filename = sc.nextLine();
        try {
            FileReader fr = new FileReader(filename);
            System.out.println("File opened successfully!");
            fr.close();
        } catch (Exception e) {
            System.out.println("File not found: " + e);
        }
        sc.close();
    }
}
```

```
<terminated> Task22 [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_23.0.2
Enter filename: ssss
File not found: java.io.FileNotFoundException: ssss (The system cannot find the file specified)
```

23. Write a program that demonstrates how to create and handle multiple custom exceptions in a single program.



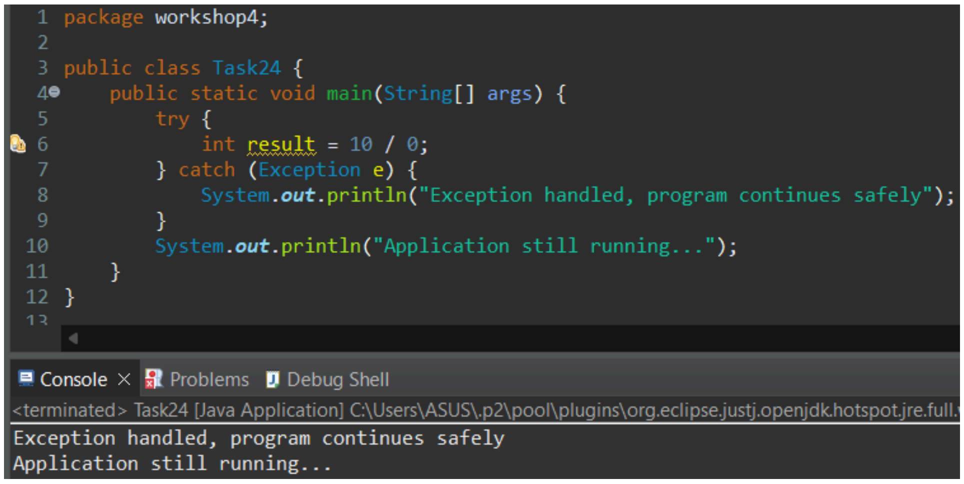
```java
package workshop4;

import java.util.Scanner;
class LowBalanceException extends Exception {
 LowBalanceException(String msg) {
     super(msg);
 }
}
class InvalidPinException extends Exception {
 InvalidPinException(String msg) {
     super(msg);
 }
}

public class Task23{

 public static void main(String[] args) {

     Scanner sc = new Scanner(System.in);

     System.out.print("Enter your account balance: ");
     int balance = sc.nextInt();

     System.out.print("Enter your PIN: ");
     int pin = sc.nextInt();

     try {
         if (balance < 500) {
             throw new LowBalanceException("Insufficent balance!");
         }
     }
     catch (LowBalanceException e) {
         System.out.println("Caught Exception: " + e.getMessage());
```



```java
     }


     try {
         if (pin != 1234) {
             throw new InvalidPinException("PIN is incorrect!");
         }

         System.out.println("Transaction successful!");
     }
     catch (InvalidPinException e) {
         System.out.println("Caught Exception: " + e.getMessage());
     }

     System.out.println("Program continues...");
     sc.close();
 }
}
```

```
<terminated> Task23 [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.h
Caught Exception: Insufficent balance!
Caught Exception: PIN is incorrect!
Program continues...
```

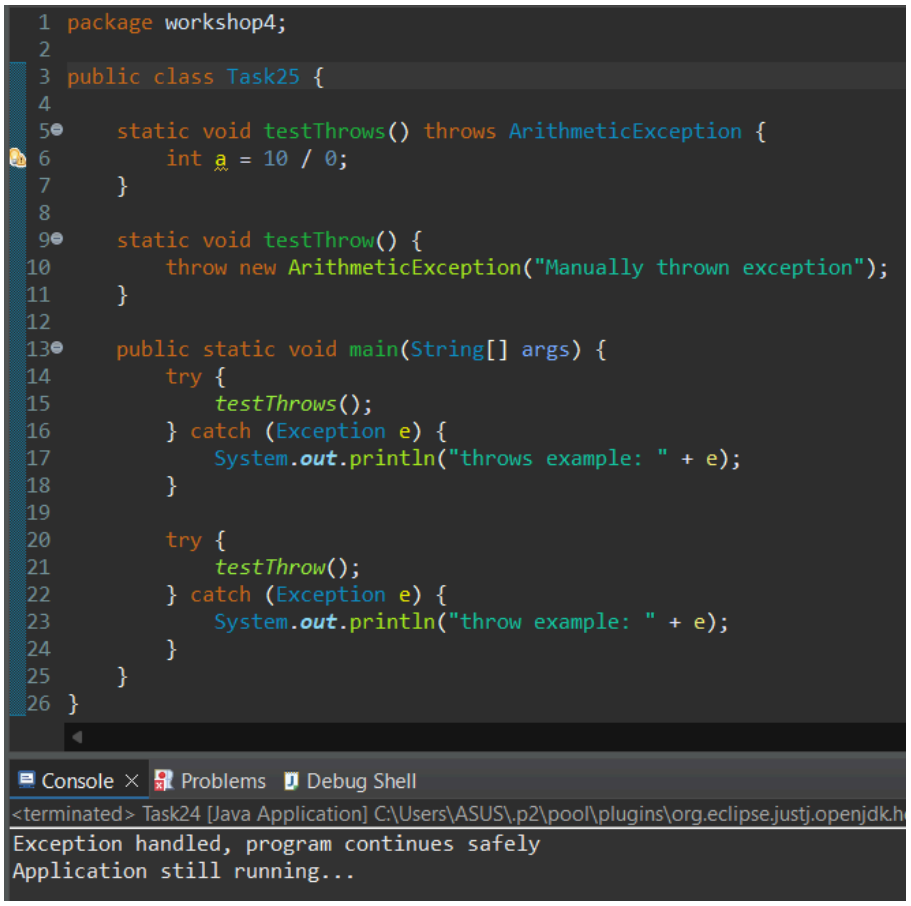
24. Create a Java program to demonstrate how exception handling improves program reliability and flow control.



```
package workshop4;

public class Task24 {
    public static void main(String[] args) {
        try {
            int result = 10 / 0;
        } catch (Exception e) {
            System.out.println("Exception handled, program continues safely");
        }
        System.out.println("Application still running...");
    }
}
```

```
<terminated> Task24 [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.
Exception handled, program continues safely
Application still running...
```

25. Develop a Java program that demonstrates the difference between throw and throws with examples in different methods.



```
package workshop4;

public class Task25 {

    static void testThrows() throws ArithmeticException {
        int a = 10 / 0;
    }

    static void testThrow() {
        throw new ArithmeticException("Manually thrown exception");
    }

    public static void main(String[] args) {
        try {
            testThrows();
        } catch (Exception e) {
            System.out.println("throws example: " + e);
        }

        try {
            testThrow();
        } catch (Exception e) {
            System.out.println("throw example: " + e);
        }
    }
}
```

```
<terminated> Task24 [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.h
Exception handled, program continues safely
Application still running...
```