Implement the following questions to understand Exception handling properly:

Unchecked Exceptions (Runtime Exceptions)

Unchecked exceptions extend `RuntimeException` and do not require explicit handling with `throws` or `try-catch`.

1. Implement `NullPointerException`

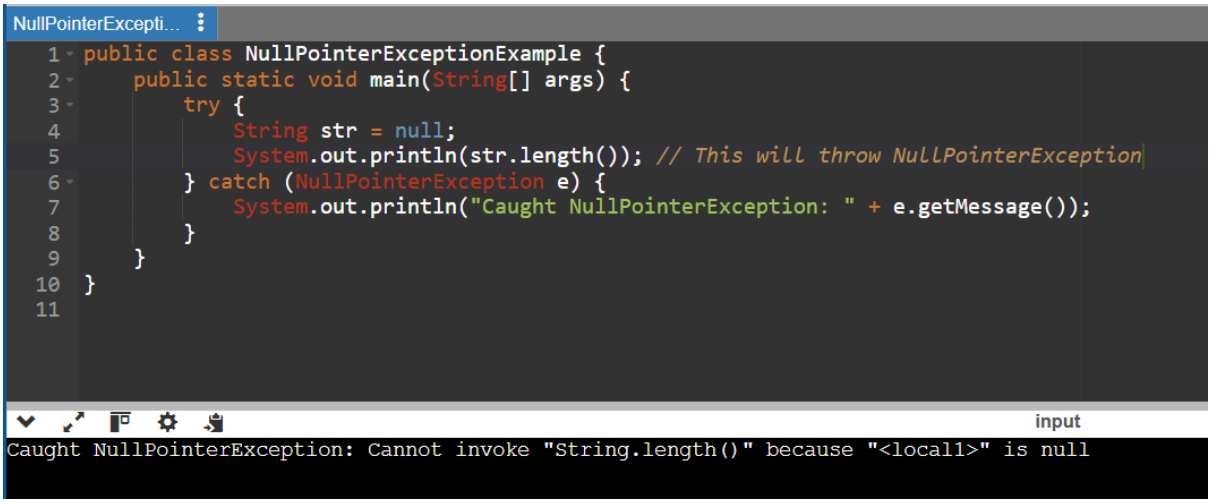
Write a Java program where you initialize a `String` as `null` and try to call the `.length()` method on it. Handle the exception using a `try-catch` block.

Sol:

```
public class NullPointerExceptionExample {
    public static void main(String[] args) {
        try {
            String str = null;
            System.out.println(str.length()); // This will throw NullPointerException
        } catch (NullPointerException e) {
            System.out.println("Caught NullPointerException: " + e.getMessage());
        }
    }
}
```

Output:

Caught NullPointerException: Cannot invoke "String.length()" because "<local1>" is null



2. Implement `ArithmeticException`

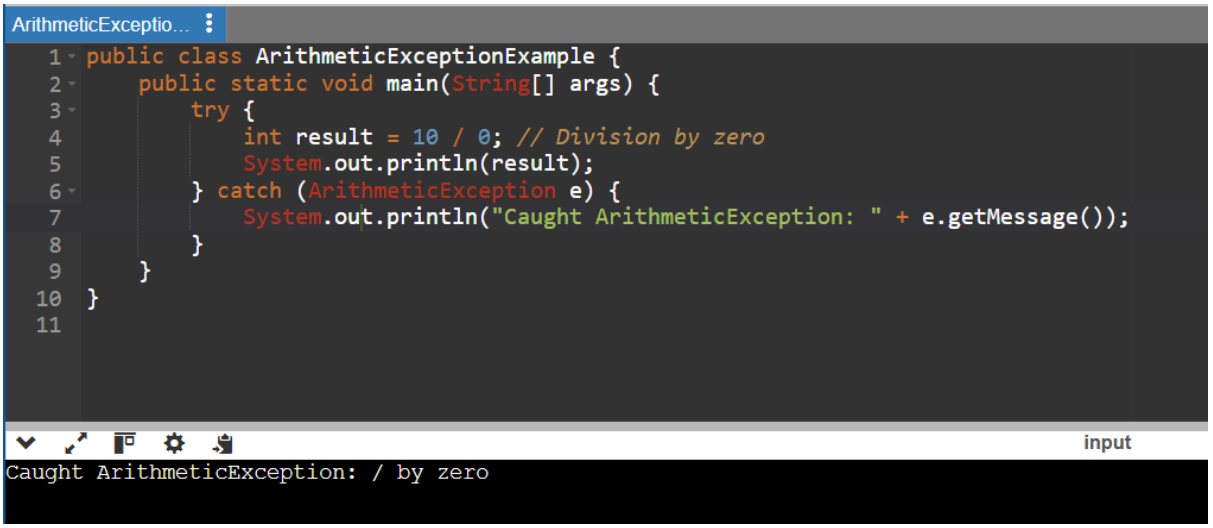
Write a Java program that performs division by zero and catches the `ArithmeticException`.

Sol:

```
public class ArithmeticExceptionExample {
    public static void main(String[] args) {
        try {
            int result = 10 / 0; // Division by zero
            System.out.println(result);
        } catch (ArithmeticException e) {
            System.out.println("Caught ArithmeticException: " + e.getMessage());
        }
    }
}
```

Output:

Caught ArithmeticException: / by zero



3. Implement `ArrayIndexOutOfBoundsException`

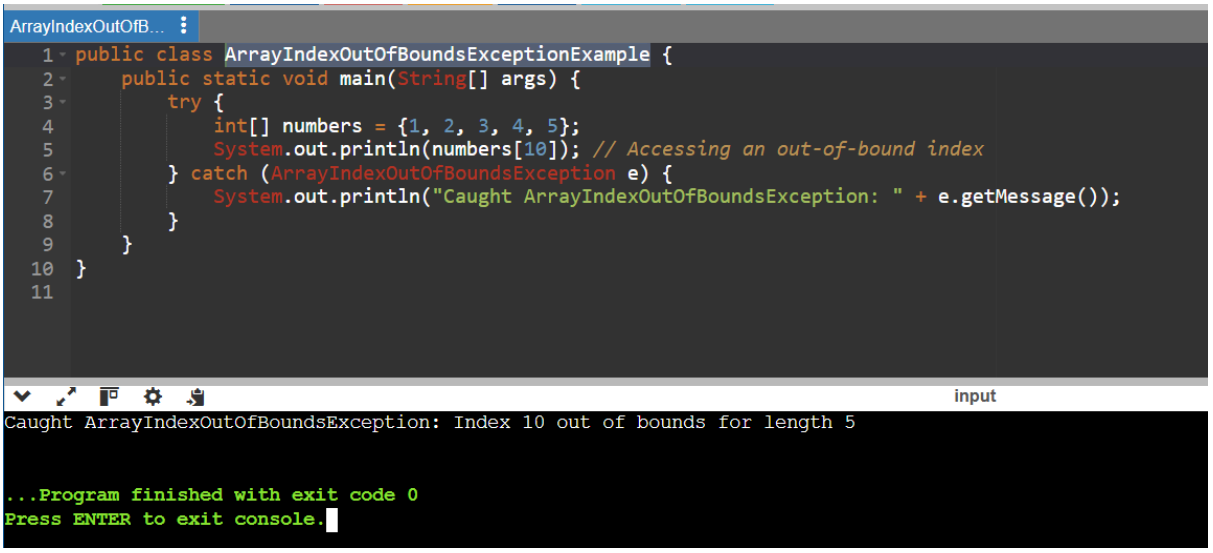
Create an array of 5 elements and try to access an index that does not exist (e.g., index `10`). Handle the exception properly.

Sol:

```
public class ArrayIndexOutOfBoundsExceptionExample {
    public static void main(String[] args) {
        try {
            int[] numbers = {1, 2, 3, 4, 5};
            System.out.println(numbers[10]); // Accessing an out-of-bound index
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Caught ArrayIndexOutOfBoundsException: " +
e.getMessage());
        }
    }
}
```

Output:

Caught ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 5



4. Implement `NumberFormatException`

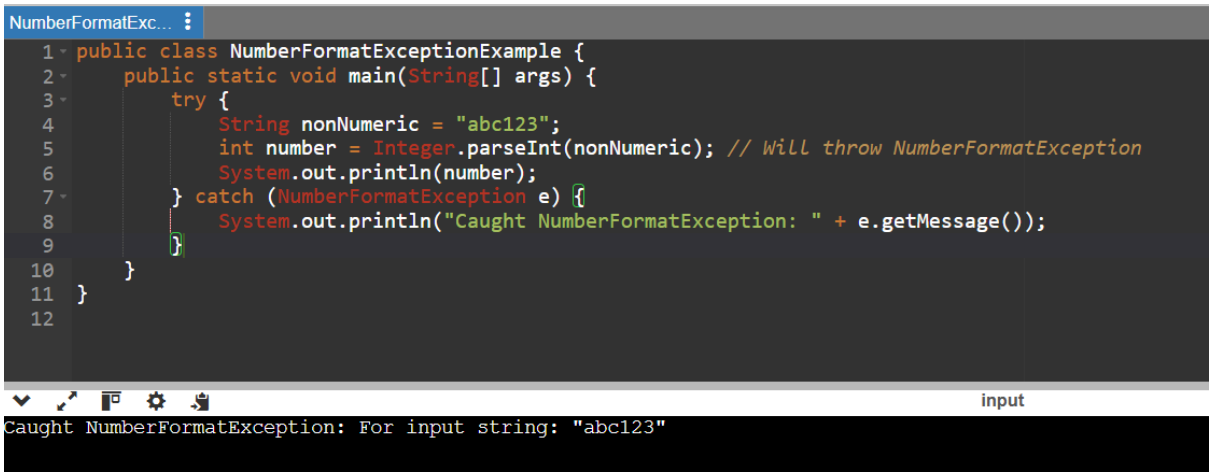
Write a Java program that tries to convert a non-numeric string to an integer using `Integer.parseInt()` and catches the `NumberFormatException`.

Sol:

```
public class NumberFormatExceptionExample {
    public static void main(String[] args) {
        try {
            String nonNumeric = "abc123";
            int number = Integer.parseInt(nonNumeric); // Will throw
NumberFormatException
            System.out.println(number);
        } catch (NumberFormatException e) {
            System.out.println("Caught NumberFormatException: " + e.getMessage());
        }
    }
}
```

Output:

Caught NumberFormatException: For input string: "abc123"



5. Implement `IllegalArgumentException`

Write a Java method `setAge(int age)` that throws an `IllegalArgumentException` if the age is negative or greater than 150.

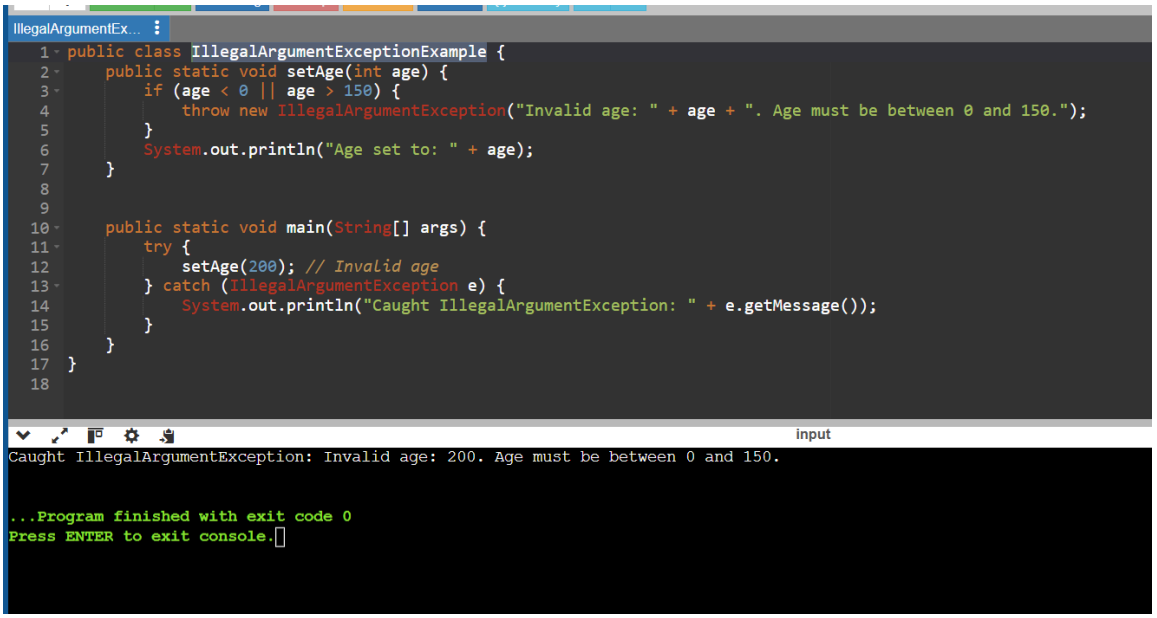
Sol:

```
public class IllegalArgumentExceptionExample {
    public static void setAge(int age) {
        if (age < 0 || age > 150) {
            throw new IllegalArgumentException("Invalid age: " + age + ". Age must be between 0 and 150.");
        }
        System.out.println("Age set to: " + age);
    }

    public static void main(String[] args) {
        try {
            setAge(200); // Invalid age
        } catch (IllegalArgumentException e) {
            System.out.println("Caught IllegalArgumentException: " + e.getMessage());
        }
    }
}
```

Output:

Caught IllegalArgumentException: Invalid age: 200. Age must be between 0 and 150.



Checked Exceptions

Checked exceptions extend `Exception` and must be either handled using `try-catch` or declared with `throws`.

6. Implement `IOException`

Write a Java program that attempts to read from a file that does not exist and catches `IOException`.

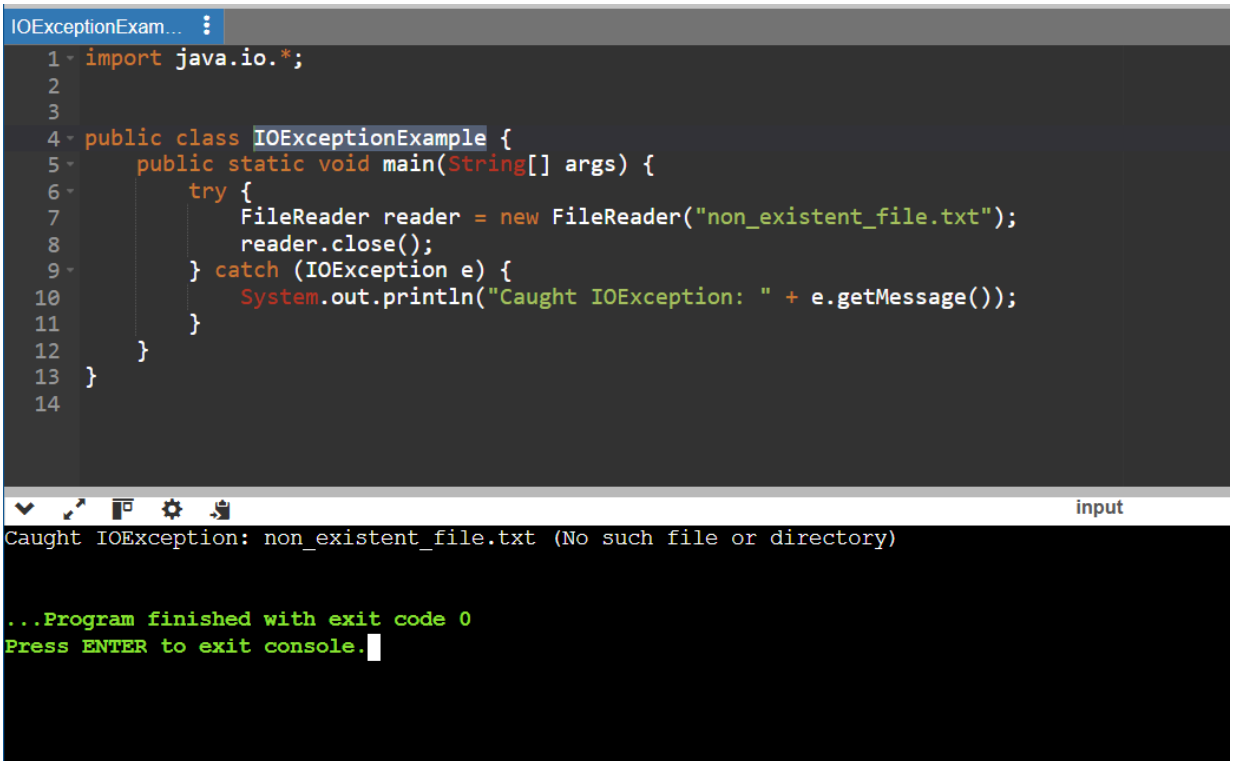
Sol:

```
import java.io.*;

public class IOExceptionExample {
    public static void main(String[] args) {
        try {
            FileReader reader = new FileReader("non_existent_file.txt");
            reader.close();
        } catch (IOException e) {
            System.out.println("Caught IOException: " + e.getMessage());
        }
    }
}
```

Output:

```
Caught IOException: non_existent_file.txt (No such file or directory)
```



7. Implement `FileNotFoundException`

Write a Java program that tries to open a file that does not exist using `FileReader`, and handle the `FileNotFoundException`.

Sol:

```
import java.io.*;

public class FileNotFoundExceptionExample {
    public static void main(String[] args) {
        try {
            FileReader file = new FileReader("missing_file.txt"); // File does not exist
            BufferedReader reader = new BufferedReader(file);
            System.out.println(reader.readLine());
            reader.close();
        } catch (FileNotFoundException e) {
            System.out.println("Caught FileNotFoundException: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("Caught IOException while reading file: " + e.getMessage());
        }
    }
}
```

Output:

Caught FileNotFoundException: missing_file.txt (No such file or directory)

FileNotFoundExce...

```java
import java.io.*;

public class FileNotFoundExceptionExample {
    public static void main(String[] args) {
        try {
            FileReader file = new FileReader("missing_file.txt"); // File does not exist
            BufferedReader reader = new BufferedReader(file);
            System.out.println(reader.readLine());
            reader.close();
        } catch (FileNotFoundException e) {
            System.out.println("Caught FileNotFoundException: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("Caught IOException while reading file: " + e.getMessage());
        }
    }
}
```

input

```
Caught FileNotFoundException: missing_file.txt (No such file or directory)


...Program finished with exit code 0
Press ENTER to exit console.
```