

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`

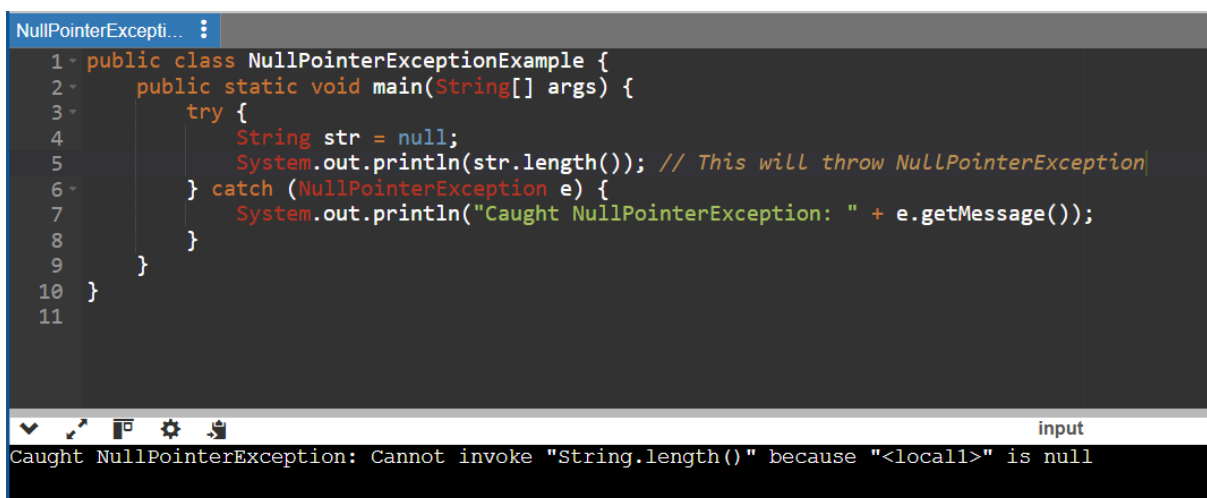
Write a Java program where you initialize a `String` as `null` and try to call the `.length()` method on it. Handle the exception using a `try-catch` block.

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

A screenshot of an IDE window titled 'NullPointerExceptionExample.java'. The code is as follows:

```
1 public class NullPointerExceptionExample {
2     public static void main(String[] args) {
3         try {
4             String str = null;
5             System.out.println(str.length()); // This will throw NullPointerException
6         } catch (NullPointerException e) {
7             System.out.println("Caught NullPointerException: " + e.getMessage());
8         }
9     }
10 }
11
```

The IDE's output console at the bottom shows the message: 'Caught NullPointerException: Cannot invoke "String.length()" because "<local1>" is null'. The console also has a tab labeled 'input'.

2. Implement ArithmeticException

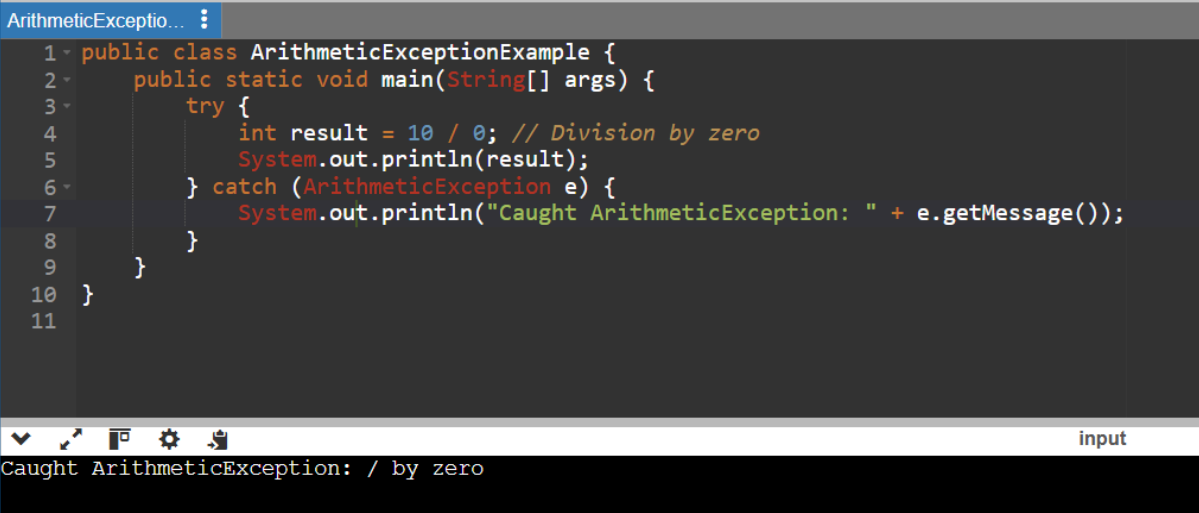
Write a Java program that performs division by zero and catches the ArithmeticException.

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



The screenshot shows an IDE window titled 'ArithmeticException...' containing the following Java code:

```
1 public class ArithmeticExceptionExample {  
2     public static void main(String[] args) {  
3         try {  
4             int result = 10 / 0; // Division by zero  
5             System.out.println(result);  
6         } catch (ArithmeticException e) {  
7             System.out.println("Caught ArithmeticException: " + e.getMessage());  
8         }  
9     }  
10 }  
11
```

At the bottom of the IDE, the output is displayed: 'Caught ArithmeticException: / by zero'.

3. Implement ArrayIndexOutOfBoundsException

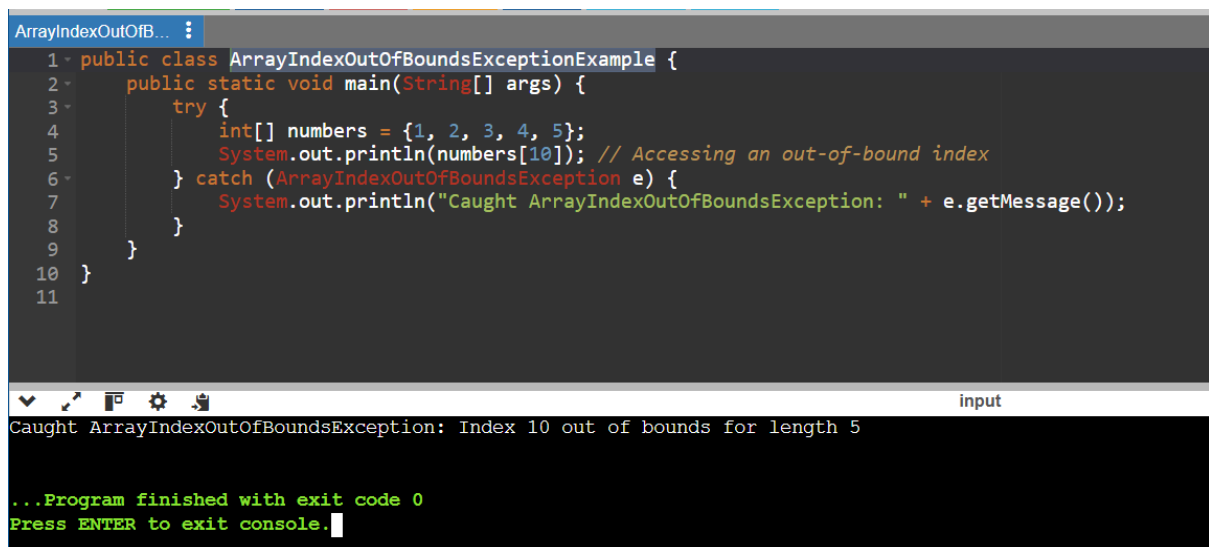
Create an array of 5 elements and try to access an index that does not exist (e.g., index 10). Handle the exception properly.

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

A screenshot of an IDE window titled 'ArrayIndexOutOfB...'. The editor shows the Java code from the previous block. Below the editor, the console output is visible, showing the exception message: 'Caught ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 5'. At the bottom, it says '...Program finished with exit code 0' and 'Press ENTER to exit console.' with a cursor.

```
ArrayIndexOutOfB... :
1 public class ArrayIndexOutOfBoundsExceptionExample {
2     public static void main(String[] args) {
3         try {
4             int[] numbers = {1, 2, 3, 4, 5};
5             System.out.println(numbers[10]); // Accessing an out-of-bound index
6         } catch (ArrayIndexOutOfBoundsException e) {
7             System.out.println("Caught ArrayIndexOutOfBoundsException: " + e.getMessage());
8         }
9     }
10 }
11

input
Caught ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 5

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

4. Implement NumberFormatException

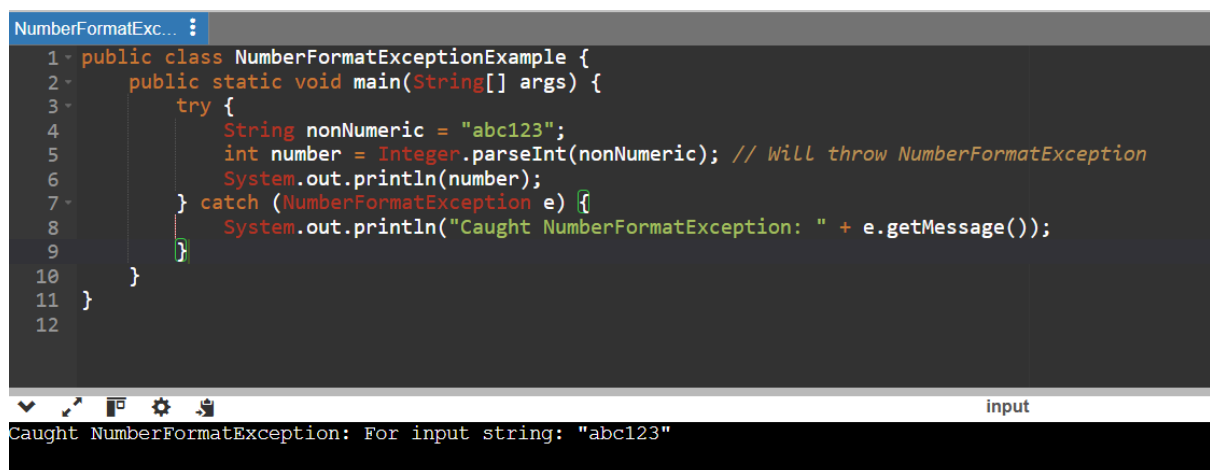
Write a Java program that tries to convert a non-numeric string to an integer using `Integer.parseInt()` and catches the `NumberFormatException`.

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"

A screenshot of an IDE window titled "NumberFormatExceptionExc...". The editor shows the Java code from the previous block. The output console at the bottom displays the message "Caught NumberFormatException: For input string: "abc123"". The word "input" is visible in the top right corner of the console area.

```
1 public class NumberFormatExceptionExample {
2     public static void main(String[] args) {
3         try {
4             String nonNumeric = "abc123";
5             int number = Integer.parseInt(nonNumeric); // Will throw NumberFormatException
6             System.out.println(number);
7         } catch (NumberFormatException e) {
8             System.out.println("Caught NumberFormatException: " + e.getMessage());
9         }
10    }
11 }
12
```

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.

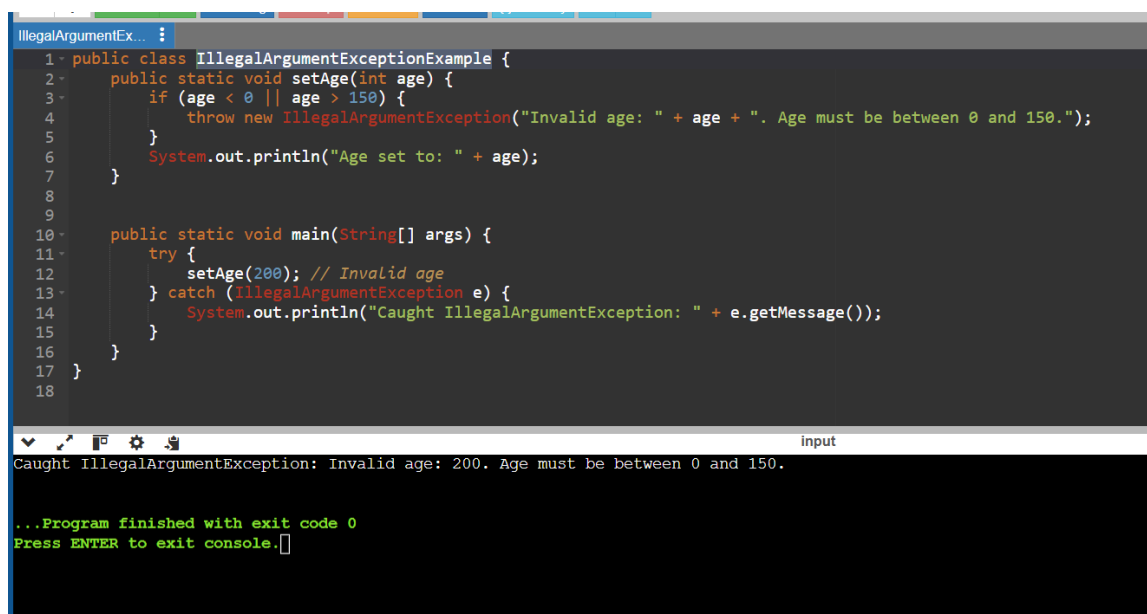
Sol:

```
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.

A screenshot of a Java IDE. The top pane shows the source code for the `IllegalArgumentExceptionExample` class, with line numbers 1 through 18. The code defines a `setAge` method that throws an `IllegalArgumentException` for invalid ages and a `main` method that calls `setAge(200)` and catches the exception. The bottom pane shows the output: "Caught IllegalArgumentException: Invalid age: 200. Age must be between 0 and 150." followed by "...Program finished with exit code 0" and "Press ENTER to exit console." The IDE interface includes a toolbar with icons for running, debugging, and other actions, and a tab labeled "input" at the bottom right.

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)

```
IOExceptionExam... :
1 import java.io.*;
2
3
4 public class IOExceptionExample {
5     public static void main(String[] args) {
6         try {
7             FileReader reader = new FileReader("non_existent_file.txt");
8             reader.close();
9         } catch (IOException e) {
10             System.out.println("Caught IOException: " + e.getMessage());
11         }
12     }
13 }
14
```

input

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

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

7. Implement FileNotFoundException

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

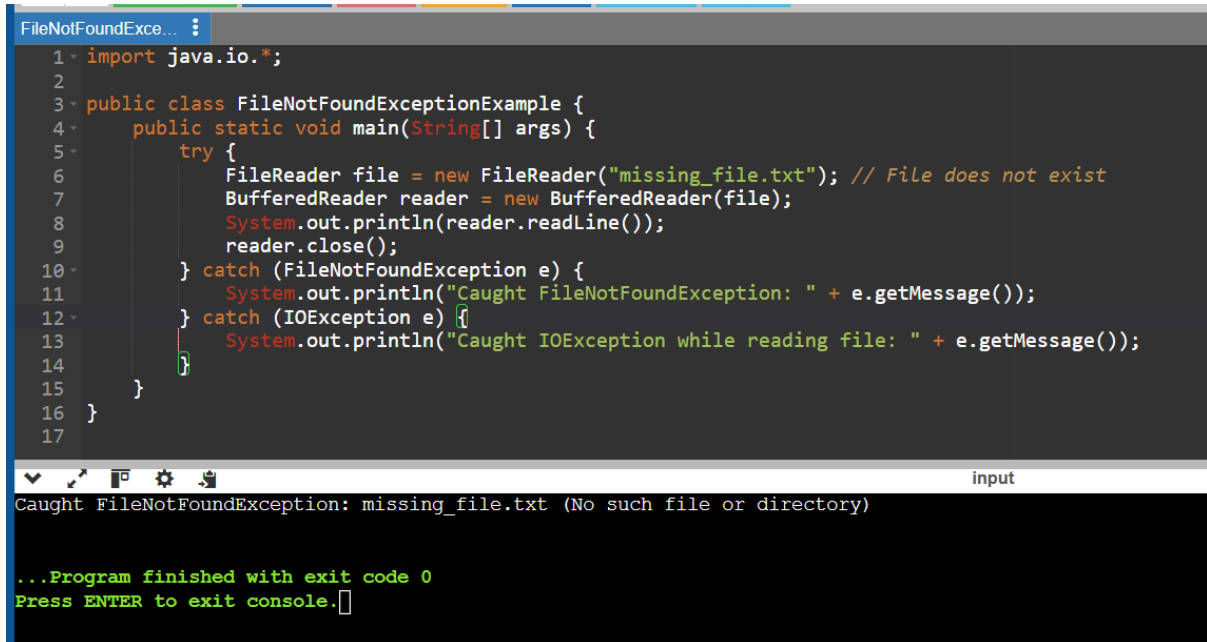
Sol:

```
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)



The screenshot shows a Java IDE with a file named 'FileNotFoundExceptionExample.java'. The code attempts to read a file named 'missing_file.txt', which does not exist. The code includes a try-catch block to handle the 'FileNotFoundException' and a general 'IOException' catch block. The console output shows the caught exception message: 'Caught FileNotFoundException: missing_file.txt (No such file or directory)'. The program then finishes with exit code 0 and prompts the user to press ENTER to exit the console.

```
1 import java.io.*;
2
3 public class FileNotFoundExceptionExample {
4     public static void main(String[] args) {
5         try {
6             FileReader file = new FileReader("missing_file.txt"); // File does not exist
7             BufferedReader reader = new BufferedReader(file);
8             System.out.println(reader.readLine());
9             reader.close();
10        } catch (FileNotFoundException e) {
11            System.out.println("Caught FileNotFoundException: " + e.getMessage());
12        } catch (IOException e) {
13            System.out.println("Caught IOException while reading file: " + e.getMessage());
14        }
15    }
16 }
17
```

input

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

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