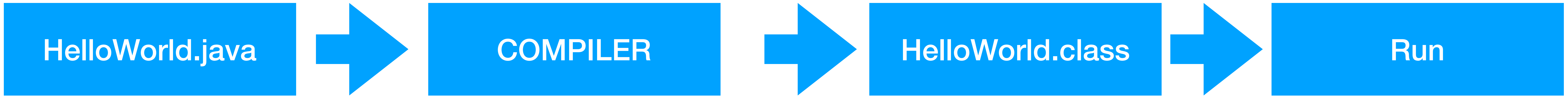
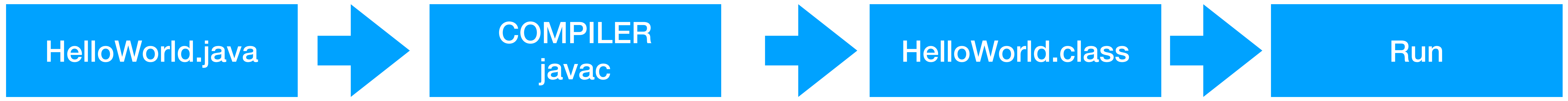
JAVA PROGRAM EXECUTION FLOW



JAVA PROGRAM EXECUTION FLOW



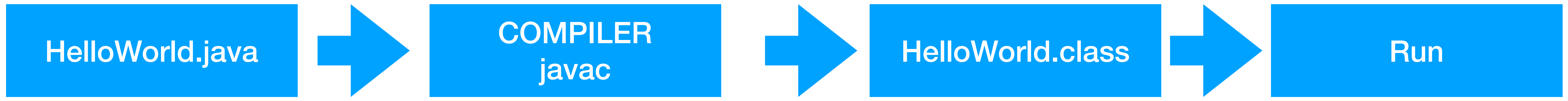
```
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello World!")
   }
}
```

COMPILE ERROR! Missing semicolon

```
public class HelloWorld {
   public static void main(String[] args) {
      system.out.println("Hello World!”);
   }
}
```

COMPILE ERROR! system is unknown

JAVA PROGRAM EXECUTION FLOW



```
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello World!”);
   }
}
```

NO COMPILER ERRORS

RUNS SUCCESSFULLY

JAVA PROGRAM EXECUTION FLOW

$$\begin{tikzcd}
\text{HelloWorld.java} \arrow[r] & \text{COMPILER javac} \arrow[r] & \text{HelloWorld.class} \arrow[r] & \text{Run}
\end{tikzcd}$$

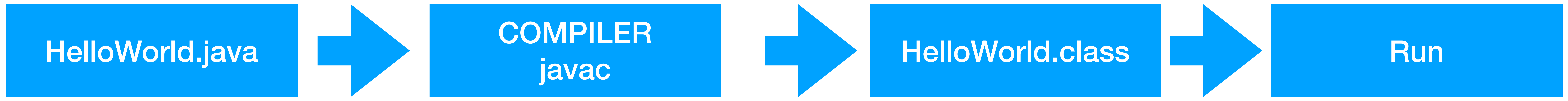
```java
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello World!”);
      int[] nums = {34,12,5};
      System.out.println(nums[3]);
      System.out.println("Hello B15!”);

   }
}
```

NO COMPILER ERRORS

```
Hello World!
ArrayIndexOutOfBoundsException
invalid index: 3
```

Program runs line by line, when line reaches where we are reading from invalid index, it will THROW EXCEPTION and execution stops. The remaining statements will not run.

JAVA PROGRAM EXECUTION FLOW

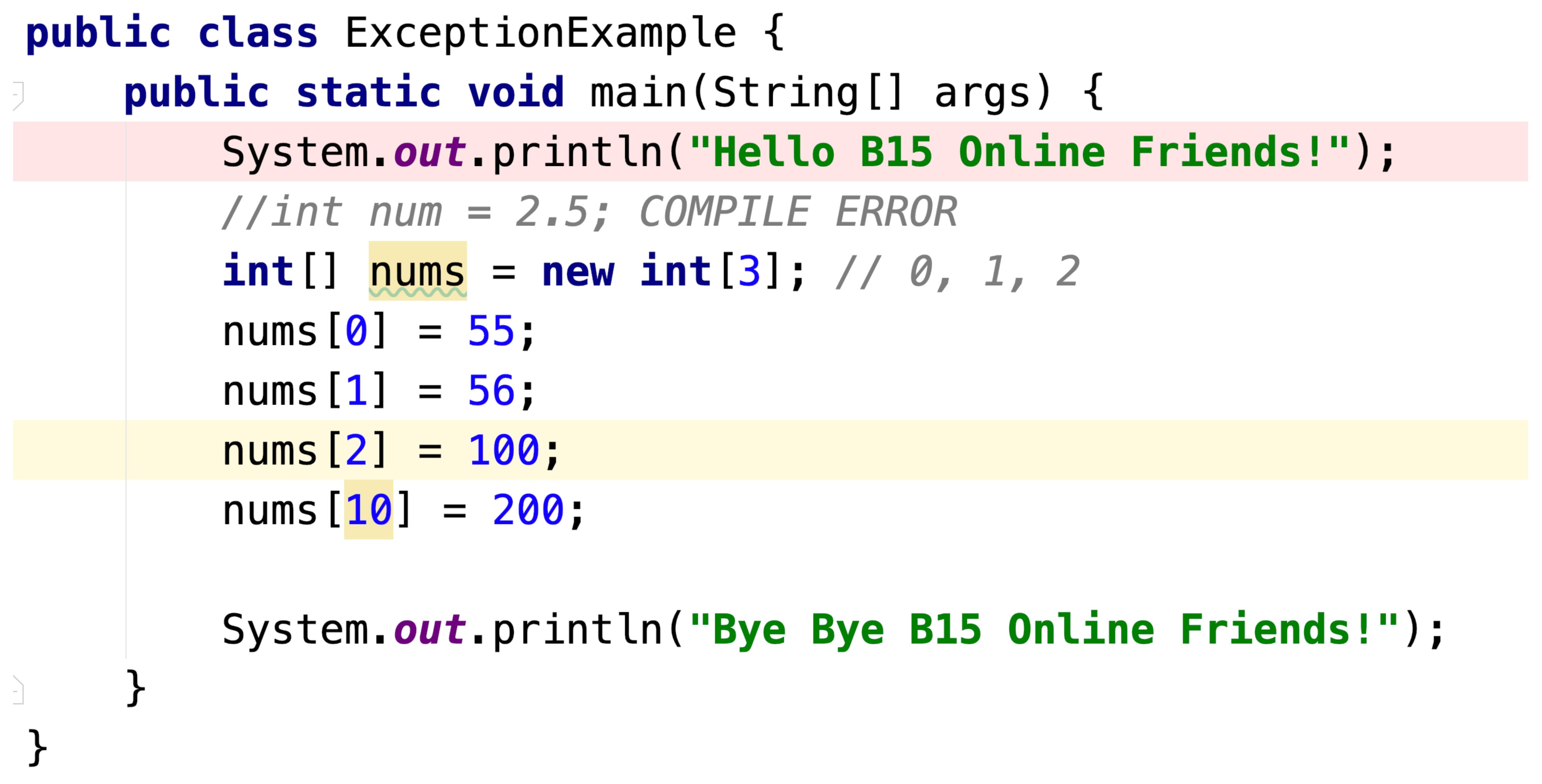


```
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello World!”);
      String str = “java”;
      System.out.println(str.charAt(10));
      System.out.println("Hello B15!”);

   }
}
```

NO COMPILER ERRORS.
Compiles successfully

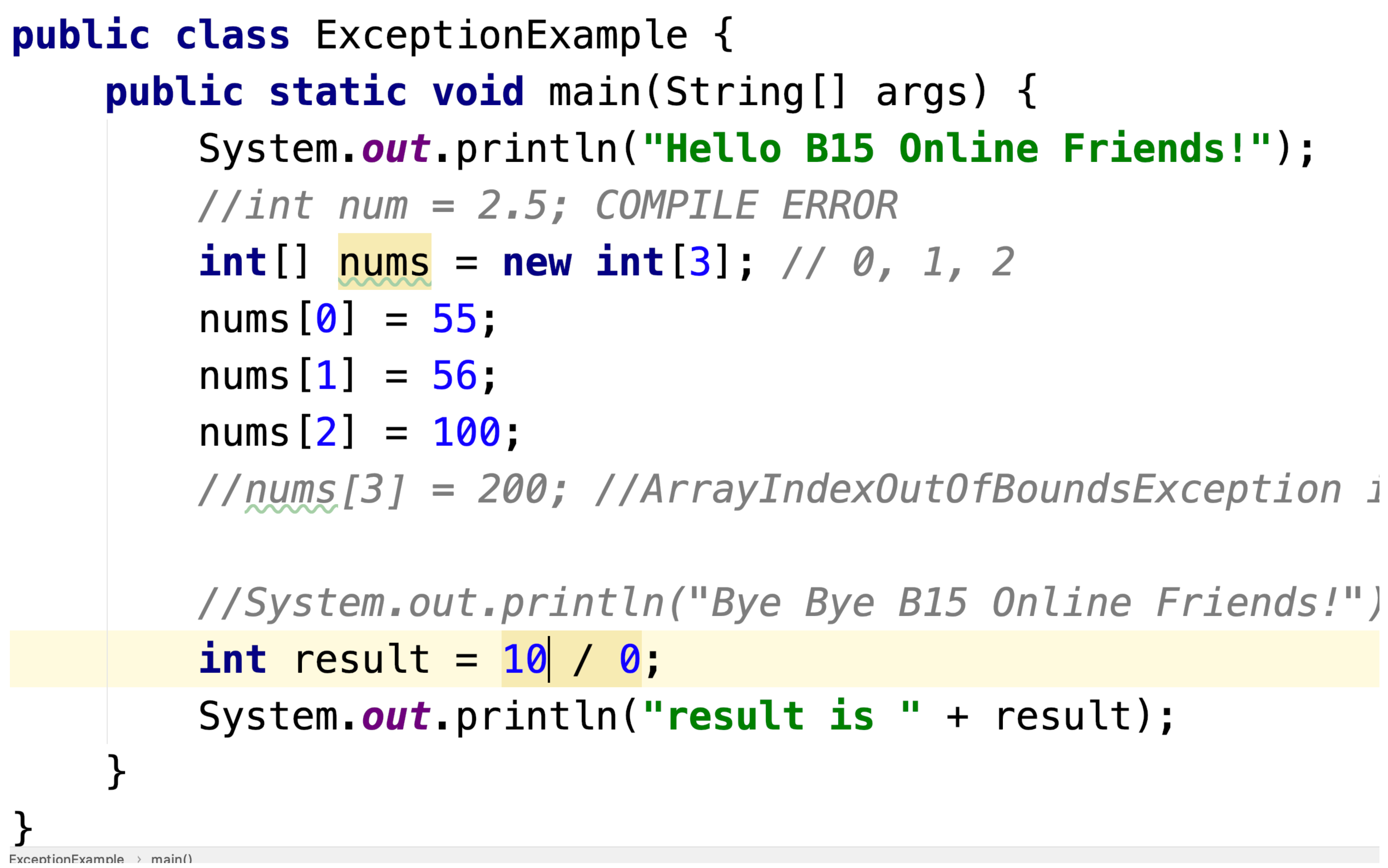
```
Hello World!
StringIndexOutOfBoundsException
```

Program runs line by line, when line reaches where we are reading from invalid index, it will THROW EXCEPTION and execution stops. The remaining statements will not run.



```java
public class ExceptionExample {
    public static void main(String[] args) {
        System.out.println("Hello B15 Online Friends!");
        //int num = 2.5; COMPILE ERROR
        int[] nums = new int[3]; // 0, 1, 2
        nums[0] = 55;
        nums[1] = 56;
        nums[2] = 100;
        nums[10] = 200;

        System.out.println("Bye Bye B15 Online Friends!");
    }
}
```

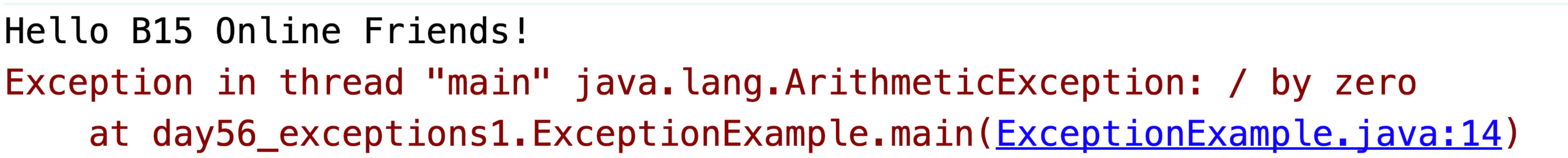
```
Hello B15 Online Friends!

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10
	at day56_exceptions1.ExceptionExample.main(ExceptionExample.java:11)
```



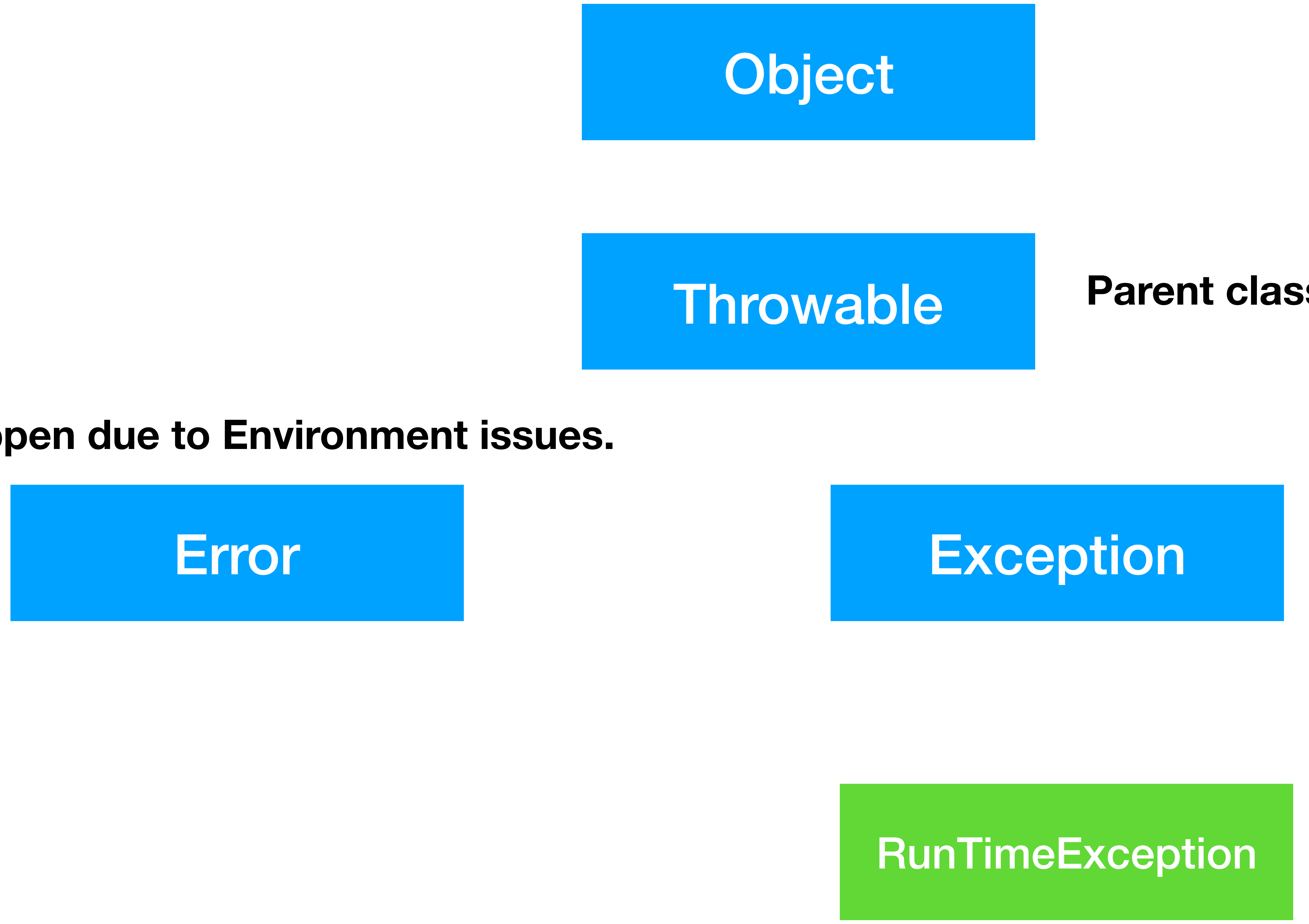
```
public class ExceptionExample {
    public static void main(String[] args) {
        System.out.println("Hello B15 Online Friends!");
        //int num = 2.5; COMPILE ERROR
        int[] nums = new int[3]; // 0, 1, 2
        nums[0] = 55;
        nums[1] = 56;
        nums[2] = 100;
        //nums[3] = 200; //ArrayIndexOutOfBoundsException i

        //System.out.println("Bye Bye B15 Online Friends!")
        int result = 10 / 0;
        System.out.println("result is " + result);
    }
}
```

ExceptionExample › main()

STACK TRACE



```
Hello B15 Online Friends!
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at day56_exceptions1.ExceptionExample.main(ExceptionExample.java:14)
```



Parent class of all exceptions in java

**Runtime Error happen due to Environment issues.
Not due to code.**

**CHECKED EXCEPTIONS are subclasses
Of Exception class**

**UNCHECKED EXCEPTIONS are subclasses
Of RunTimeException class**

Runtime Error

Throwable

Error

StackOverflowError

Happens when stack memory is full

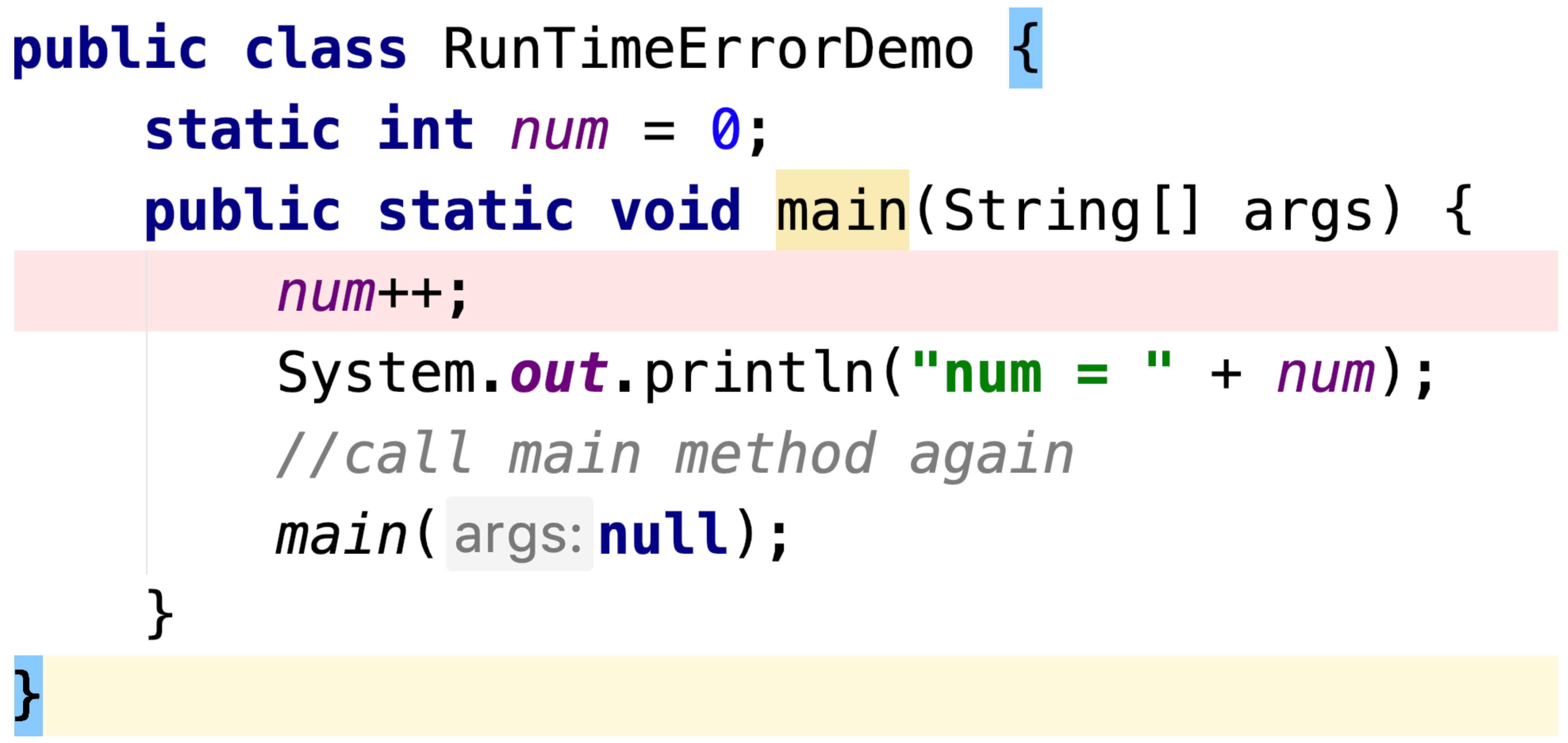
STACK

OutOfMemoryError

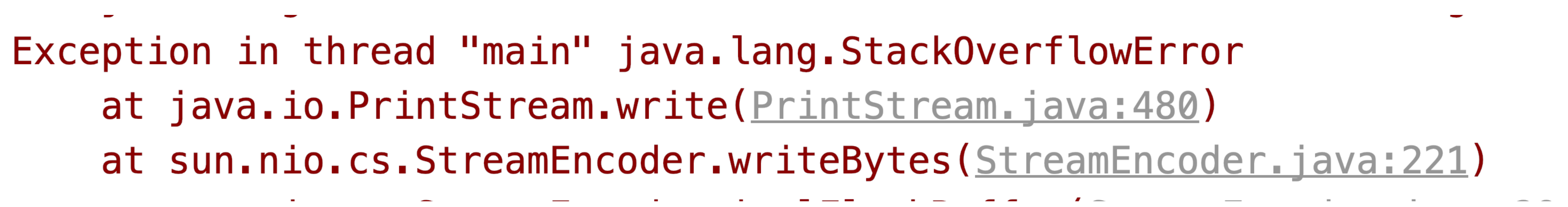
Happens when heap memory is full

HEAP

STACKOVERFLOWERROR



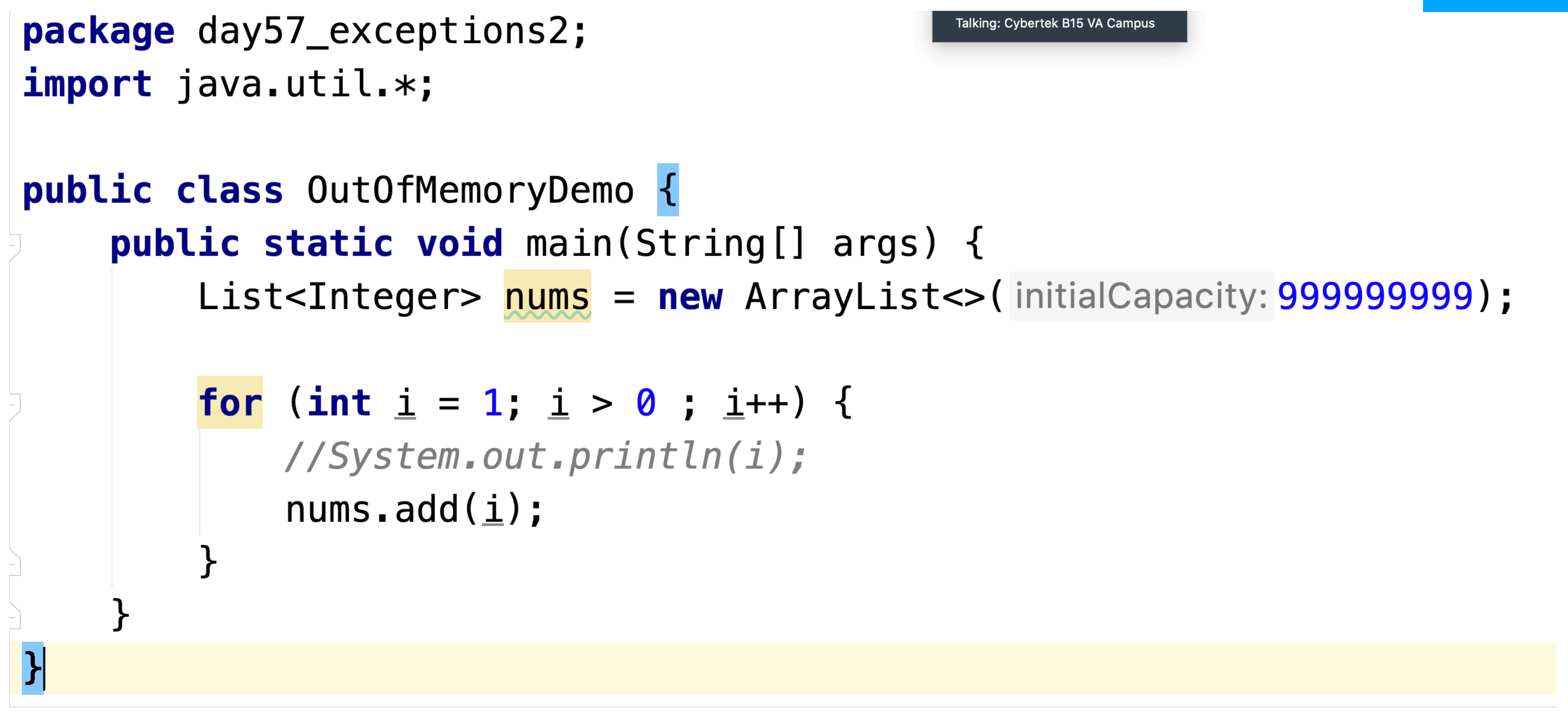
```java
public class RunTimeErrorDemo {
    static int num = 0;
    public static void main(String[] args) {
        num++;
        System.out.println("num = " + num);
        //call main method again
        main( args: null);
    }
}
```



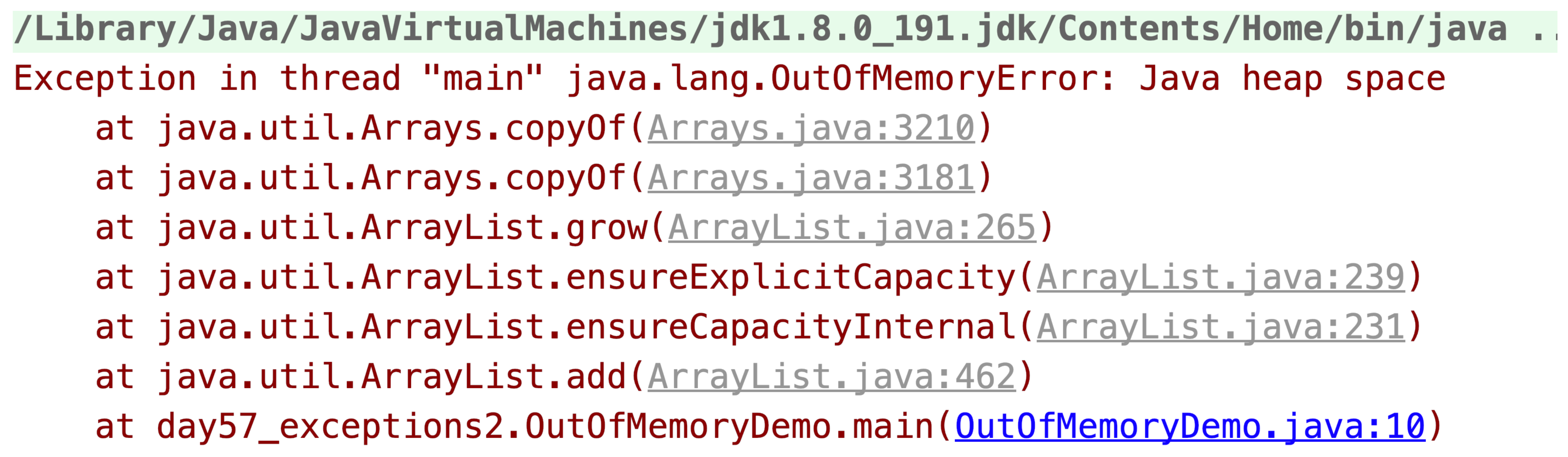
```
Exception in thread "main" java.lang.StackOverflowError
	at java.io.PrintStream.write(PrintStream.java:480)
	at sun.nio.cs.StreamEncoder.writeBytes(StreamEncoder.java:221)
```

Whenever method is called in Java, a frame is placed in Stack memory for that method call. If method calls itself recursively, another frame is placed on existing frame. If it keeps continuing , Stack will be eventually full and StackOverFlow Error is thrown.

OUTOFMEMORYERROR



```
package day57_exceptions2;
import java.util.*;

public class OutOfMemoryDemo {
    public static void main(String[] args) {
        List<Integer> nums = new ArrayList<>( initialCapacity: 999999999);

        for (int i = 1; i > 0 ; i++) {
            //System.out.println(i);
            nums.add(i);
        }
    }
}
```



```
/Library/Java/JavaVirtualMachines/jdk1.8.0_191.jdk/Contents/Home/bin/java ..
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
    at java.util.Arrays.copyOf(Arrays.java:3210)
    at java.util.Arrays.copyOf(Arrays.java:3181)
    at java.util.ArrayList.grow(ArrayList.java:265)
    at java.util.ArrayList.ensureExplicitCapacity(ArrayList.java:239)
    at java.util.ArrayList.ensureCapacityInternal(ArrayList.java:231)
    at java.util.ArrayList.add(ArrayList.java:462)
    at day57_exceptions2.OutOfMemoryDemo.main(OutOfMemoryDemo.java:10)
```

This error happens when Heap Memory is full. Normally if we keep creating objects , or if we have One object that is very large. Ex: Arraylist object with many elements, Excel file is opened with Many rows.

CHECKED EXCEPTION

Throwable

Error

Exception

InterruptedException

FileNotFoundException

IOException

URLException

SQLException

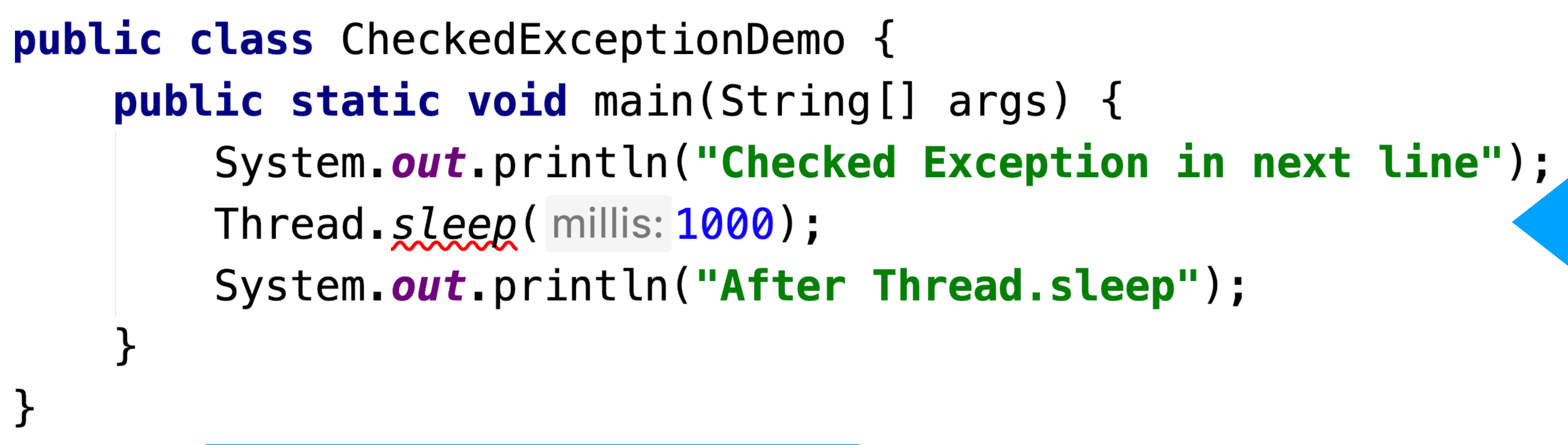
Some lines of code in java, might
Cause one of checkedExceptions during runtime.
Java knows about those statements.
PROGRAMMER MUST HANDLE CHECKED EXCEPTIONS
FOR THE PROGRAM TO COMPILE.

IQ: WHAT IS CHECKED EXCEPTION?

CHECKED EXCEPTIONS MUST BE HANDLED OR DECLARED
BY THE PROGRAMMER FOR THE CODE TO COMPILE.
OTHERWISE CODE WILL NOT COMPILE.

Thread.sleep(2000); InterruptedException

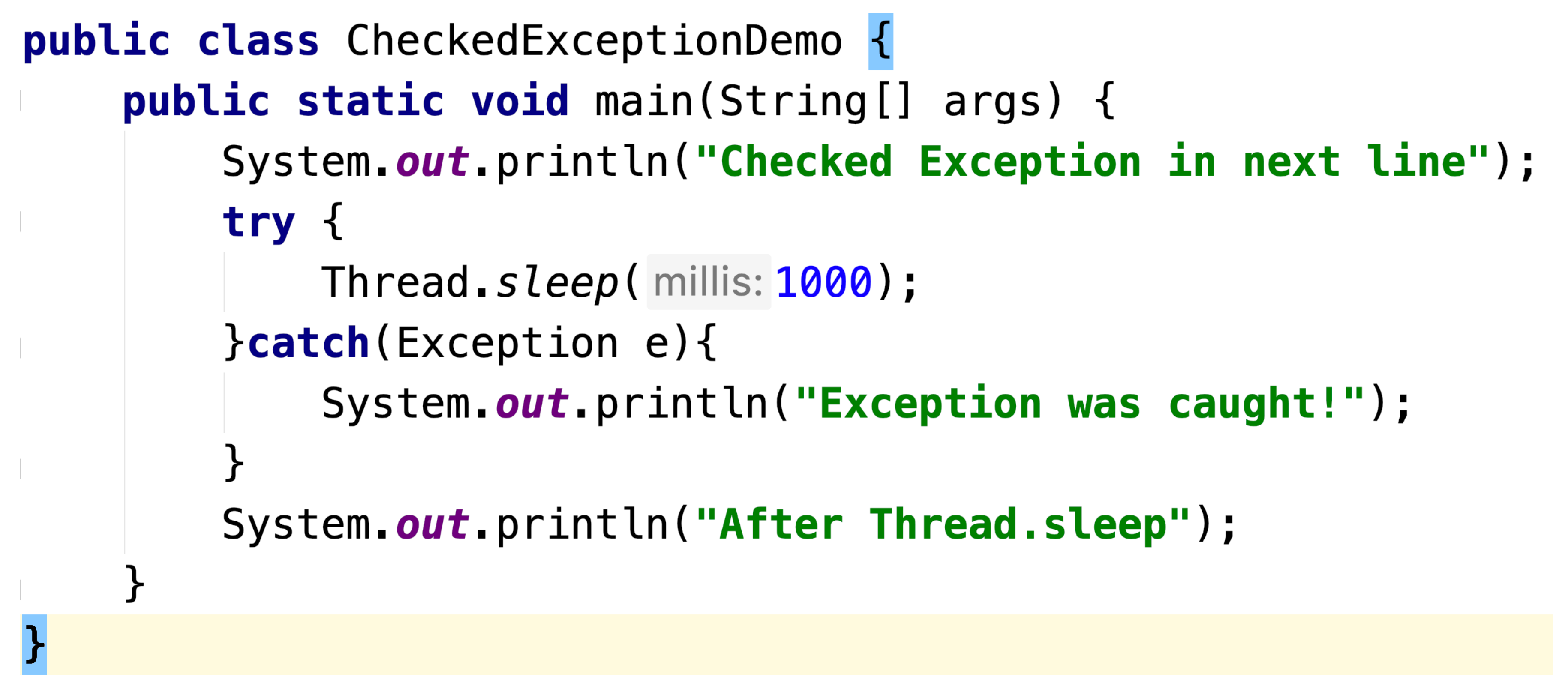
```
This line throws a CHECKED EXCEPTION -> InterruptedException
CHECKED EXCEPTIONS MUST BE: 1) Handled or 2) Declared for the code to compile.
```



```
public class CheckedExceptionDemo {
    public static void main(String[] args) {
        System.out.println("Checked Exception in next line");
        Thread.sleep( millis: 1000);
        System.out.println("After Thread.sleep");
    }
}
```

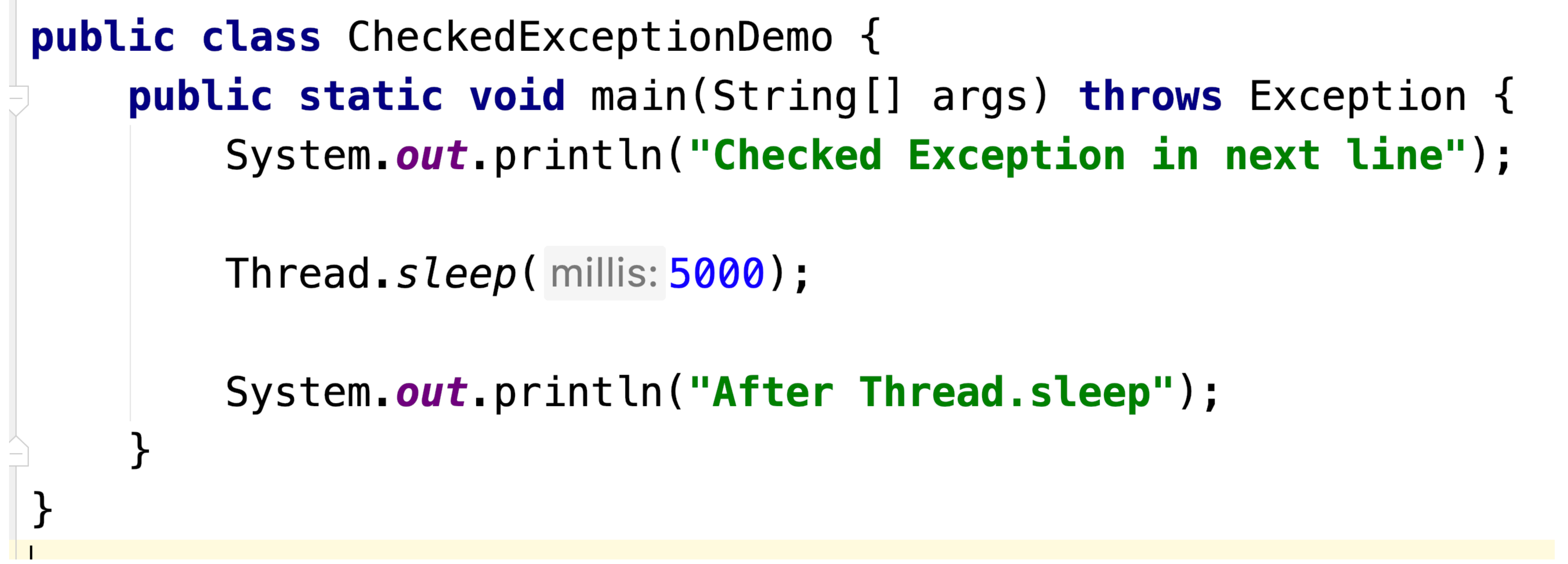
Unhandled Checked Exception

HANDLE using Try catch block



```
public class CheckedExceptionDemo {
    public static void main(String[] args) {
        System.out.println("Checked Exception in next line");
        try {
            Thread.sleep( millis: 1000);
        }catch(Exception e){
            System.out.println("Exception was caught!");
        }
        System.out.println("After Thread.sleep");
    }
}
```

DECLARE using throws keyword



```
public class CheckedExceptionDemo {
    public static void main(String[] args) throws Exception {
        System.out.println("Checked Exception in next line");

        Thread.sleep( millis: 5000);

        System.out.println("After Thread.sleep");
    }
}
```

UNCHECKED EXCEPTIONS

UnChecked exception happen during runtime and code will compile even if we do not handle them.

Unchecked exceptions are RuntimeException
And all of its sub classes

Throwable	**Checked**
Exception	**Checked**
RuntimeException	**UnChecked**
ArrayIndexOutOfBoundsException	**UnChecked**
NullPointerException	**UnChecked**
ArithmeticException	**UnChecked**

IQ: DIFFERENCE BETWEEN CHECKED AND UNCHECKED EXCEPTIONS IN JAVA?

CHECKED	UNCHECKED
CHECKED Exceptions MUST be handled or declared for code to COMPILE	Code will compile even if we do not handle UncheckedExceptions
CHECKED Exceptions are Throwable, Exception and all its sub classes except RuntimeException class	UncheckedExceptions are RuntimeException and all of its sub classes.