

# Handling Exceptions

---



**Andrejs Doronins**

TEST AUTOMATION ENGINEER



# Good Exception Handling

```
File file = new File("file.txt");  
  
try (FileInputStream inputStream = new FileInputStream(file)) {  
    // read a file  
}  
catch (FileNotFoundException e) {  
    log.error(e);  
}  
catch (IOException e) {  
    log.error(e);  
}  
}
```



# Good Exception Handling?

```
File file = new File("file.txt");  
FileInputStream stream;  
  
try {  
    // read file  
} catch (FileNotFoundException e) {  
    e.printStackTrace();  
} catch (Throwable e) {  
    // oops...  
}
```



# Module Overview



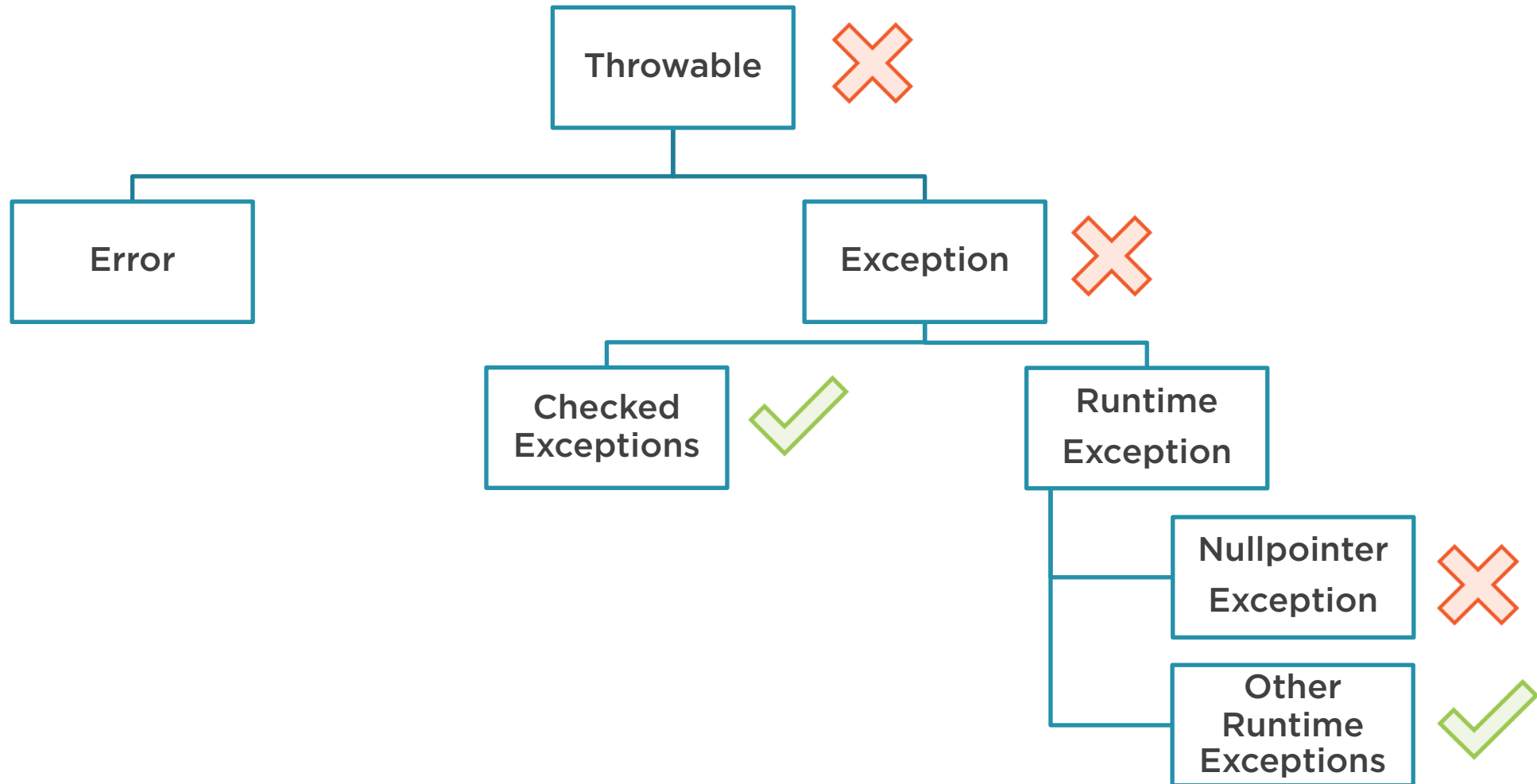
Catching the wrong exceptions

Ignoring exceptions

Useless code in catch block

Exceptions in finally block

# Don't Catch Them All

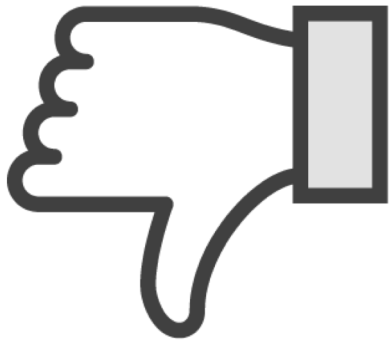




**Don't handle NPEs.  
Prevent them!**









# Catch Block Anti-patterns



## Catch block shouldn't:

- Be empty
- Have only comments
- Contain unhelpful code

<code>catch {</code>	<code>}</code>	
<code>catch { // should never happen</code>	<code>}</code>	
<code>catch { return null;</code>	<code>}</code>	
<code>catch { e.printStackTrace();</code>	<code>}</code>	
<code>catch { log.error(e);</code>	<code>}</code>	
<code>catch { throw new CustomException(e);</code>	<code>}</code>	







Avoid exceptions in finally { }



# Summary



Catch specific exceptions



Proper handling in catch { }



Finally block and Java 7 try-with-resources