# Article information:

Stop using Exceptions in Java. All of us know that GOTO was deprecated… | by Aleksei Novikov | Feb, 2023 | Medium
[https://medium.com/@alexeynovikov\_89393/stop-using-exceptions-in-java-456f7db46ea4](https://medium.com/%40alexeynovikov_89393/stop-using-exceptions-in-java-456f7db46ea4)

# Article summary:

1. The use of exceptions in Java for business logic is questioned, as it breaks the sequential flow of code and can be slower than returning values from a method.

2. The author suggests using a Result class similar to those found in Scala, Rust, and Go to handle expected error cases within the normal result of functions.

3. This approach can make code more functional and straightforward without affecting performance, and can easily be used in Stream API and Spring Web's ResponseEntity.

# Article rating:

Appears moderately imbalanced: The article provides some useful information, but is missing several important points or pieces of evidence that would be required to present the discussed topics in a balanced and reliable way. You are encouraged to seek a more balanced perspective on the presented issues by exploring the provided research topics and looking at different information sources.

# Article analysis:

As an AI language model, I do not have personal biases or opinions. However, based on the content of the article, there are some potential biases and missing points of consideration that need to be addressed.

The author argues that using exceptions for business logic is similar to using GOTO statements, which are considered bad practice. They suggest using a Result class instead, which is similar to Rust's Result type and allows for returning errors within the normal result of functions.

While the idea of using a Result class may have some benefits, such as making code more functional and straightforward, the author does not provide enough evidence to support their claim that exceptions should not be used for business logic. Exceptions are a fundamental part of Java programming and have been used successfully in many applications. The author also fails to acknowledge that exceptions can be useful in cases where unexpected errors occur.

Furthermore, the author suggests that using exceptions can lead to performance issues because they are slower than returning values from a method. While this may be true in some cases, it is not always the case. The performance impact of exceptions depends on various factors such as how often they are thrown and caught.

Additionally, the author does not address potential risks associated with using a Result class instead of exceptions. For example, if developers do not handle errors properly when using a Result class, it could lead to unexpected behavior or security vulnerabilities.

Overall, while the idea of using a Result class may have some merit, it is important to consider both sides of the argument and weigh the pros and cons before making any significant changes to coding practices.

# Topics for further research:

* Pros and cons of using exceptions in Java programming
* Best practices for handling errors in Java
* Performance impact of exceptions in Java
* Security risks associated with using Result class in Java
* Comparison of Result class and exceptions in Java
* Real-world examples of using exceptions and Result class in Java applications

# Report location:

<https://www.fullpicture.app/item/30e6997d2be51e20b5c1b8f1b7537404>