1Z0-808 Exam Topic Reviewer

TopicId: 1028
Topic: Exception Hierarchy and Types

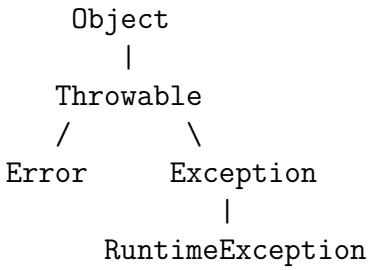
August 5, 2025

Exception Handling: Preparing for the Unexpected

Good afternoon. So far, we've written code that we assume works perfectly. But in the real world, and especially on the 1Z0-808 exam, things go wrong. Networks fail, files don't exist, users input invalid data. Exception handling is Java's structured way of managing these errors. For your exam, you must know the class hierarchy like the back of your hand and understand the strict rules the compiler enforces.

0.1 The `Throwable` Hierarchy

Every error condition you can handle or that can crash your program is an object of a class that descends from `java.lang.Throwable`.



- `Throwable`: The superclass of everything that can be thrown. You almost never interact with it directly.
- `Error`: Represents severe, abnormal problems that your application should not try to handle. These are typically related to the JVM itself. Examples: `StackOverflowError`, `OutOfMemoryError`. You are not expected to catch these.
- `Exception`: This is the parent class for all "normal" exceptions that a well-written application should anticipate and recover from. This class is the key to understanding the most important concept in this topic.

0.2 Checked vs. Unchecked Exceptions: The Compiler Is Your Boss

The single most critical concept for the exam is the difference between checked and unchecked exceptions. This distinction determines whether the compiler forces you to handle a potential problem.

0.2.1 Checked Exceptions

These are exceptions that a method is expected to handle. They represent predictable, though not ideal, situations (like a file not being found).

- **Who are they?**: Any direct subclass of `Exception`, **excluding** `RuntimeException` and its subclasses.
- **Examples:** `IOException`, `SQLException`, `FileNotFoundException`.
- **The Rule (Handle or Declare):** The compiler *checks* your code. If you call a method that can throw a checked exception, you **must** do one of two

things:

(a) Handle it using a `try-catch` block.

(b) Declare it in your method signature using the `throws` keyword.

Failure to do so results in a **COMPILE ERROR**. This is not optional.

0.2.2 Unchecked Exceptions

These represent bugs or logic errors in your code. They are "unchecked" because the compiler does not force you to handle them.

- **Who are they?**: All subclasses of `RuntimeException` and all subclasses of `Error`.
- **Examples:** `NullPointerException`, `ArrayIndexOutOfBoundsException`, `ArithmeticExce` `ClassCastException`, `NumberFormatException`.
- **The Rule:** You *can* handle them with a `try-catch` block, but you are not required to. They signal that you, the programmer, have made a mistake that should be fixed.

Key Takeaways for the 1Z0-808 Exam

- Memorize the hierarchy: `Throwable` is the parent of `Error` and `Exception`. `RuntimeException` is a child of `Exception`.
- **Checked Exceptions** are children of `Exception` (but not `RuntimeException`). You MUST handle or declare them.
- **Unchecked Exceptions** are children of `RuntimeException` or `Error`. You are NOT required to handle them.
- Be able to classify common exceptions: `IOException` is checked, `NullPointerException` is unchecked.