Interview Questions

JAVA-112: Session 8 - Exception Handling

Answering interview questions is crucial in your journey of applied learning. Review them to ensure your understanding of important topics covered in the previous session and to prepare yourself for upcoming challenges. Remember that it's important to answer these questions on your own before viewing the solution. The solutions are hyperlinked to community posts on our platform.

**Note: The questions below have been sourced from previous interviews**

Questions

1. [What are Exceptions? How do you handle Exceptions in Java?](https://crio.do/learn/crio-community/topic/what-are-exceptions-how-do-you-handle-exceptions-in-java/216296)
2. [What are the different types of exceptions in Java?](https://crio.do/learn/crio-community/topic/what-are-the-different-types-of-exceptions-in-java/216297)
3. [What are common practices for exception handling in Java?](https://crio.do/learn/crio-community/topic/what-are-common-practices-for-exception-handling-in-java/216298)
4. [Explain the difference between checked and unchecked exceptions.](https://crio.do/learn/crio-community/topic/explain-the-difference-between-checked-and-unchecked-exceptions/252740)
5. [Can we write a try block without a catch block?](https://crio.do/learn/crio-community/topic/can-we-write-a-try-block-without-a-catch-block/252741)
6. [Can we catch more than one exception in a single catch block?](https://crio.do/learn/crio-community/topic/can-we-catch-more-than-one-exception-in-a-single-catch-block/188100)
7. [How do you create a custom exception in Java?](https://crio.do/learn/crio-community/topic/how-do-you-create-a-custom-exception-in-java/252742)
8. [What is the purpose of exception handling in Java?](https://crio.do/learn/crio-community/topic/what-is-the-purpose-of-exception-handling-in-java/252776)
9. [What is a custom exception?](https://crio.do/learn/crio-community/topic/what-is-a-custom-exception/252876)

**1. What are Exceptions? How do you handle Exceptions in Java?**

* **Exception**: An **unexpected event** that disrupts the normal flow of a program (e.g., dividing by zero, invalid file access).
* **Handling in Java**: Using **try-catch-finally** blocks or **throws** keyword.
* try {
* int result = 10 / 0;
* } catch (ArithmeticException e) {
* System.out.println("Error: " + e.getMessage());
* } finally {
* System.out.println("Execution completed.");
* }

**2. What are the different types of exceptions in Java?**

1. **Checked exceptions** (compile-time exceptions) → Must be handled using try-catch or declared using throws.
   * Example: IOException, SQLException.
2. **Unchecked exceptions** (runtime exceptions) → Occur during execution, usually programming errors.
   * Example: NullPointerException, ArrayIndexOutOfBoundsException.
3. **Errors** → Serious issues that cannot be recovered.
   * Example: OutOfMemoryError, StackOverflowError.

**3. What are common practices for exception handling in Java?**

* Catch **specific exceptions** instead of generic ones.
* **Log exceptions** for debugging.
* Do not use **empty catch blocks**.
* Use **finally** (or try-with-resources) to release resources.
* Avoid using exceptions for **normal control flow**.
* Create **custom exceptions** for meaningful error messages.

**4. Explain the difference between checked and unchecked exceptions.**

* **Checked exceptions**:
  + Verified at **compile time**.
  + Must be either **handled** or declared using throws.
  + Example: IOException.
* **Unchecked exceptions**:
  + Occur at **runtime**.
  + Not required to be declared or handled.
  + Example: NullPointerException.

**5. Can we write a try block without a catch block?**

* **Yes**, but it must be followed by either:
  + A **catch block**, or
  + A **finally block**.
* Example:
* try {
* int x = 5 / 0;
* } finally {
* System.out.println("Cleanup code executed.");
* }

**6. Can we catch more than one exception in a single catch block?**

* **Yes**, from Java 7 onwards, using **multi-catch** syntax with |.
* Example:
* try {
* int arr[] = new int[5];
* arr[10] = 20; // ArrayIndexOutOfBoundsException
* } catch (ArithmeticException | ArrayIndexOutOfBoundsException e) {
* System.out.println("Exception: " + e);
* }

**7. How do you create a custom exception in Java?**

* Extend the Exception (checked) or RuntimeException (unchecked) class.
* Example:
* class InvalidAgeException extends Exception {
* public InvalidAgeException(String message) {
* super(message);
* }
* }
* public class Test {
* public static void main(String[] args) throws InvalidAgeException {
* int age = 15;
* if (age < 18) {
* throw new InvalidAgeException("Age must be 18 or above");
* }
* }
* }

**8. What is the purpose of exception handling in Java?**

* To maintain **normal program flow** when unexpected events occur.
* To prevent **abrupt termination** of programs.
* To provide **clear error messages** for debugging.
* To ensure **resources are released** properly (files, DB connections).

**9. What is a custom exception?**

* A **user-defined exception** created by extending Exception or RuntimeException.
* Used when predefined exceptions don’t describe the error well.
* Example: InsufficientBalanceException in a banking app.