Java Exception Handling - Complete Revision with Definitions, Examples & Answers

---

1. Exception Definition

Exception: An abnormal situation that disrupts normal program flow. Handled in Java to prevent program crash.

Example:

```java
int a = 10 / 0; // throws ArithmeticException
```

---

2. try-catch Block

Definition: Used to handle exceptions and continue program execution.

Example:

```java
try {
    int result = 10 / 0;
} catch (ArithmeticException e) {
    System.out.println("Caught: " + e);
}
```

---

3. finally Block

Definition: Executes cleanup code that must run whether exception occurs or not.

Example:

```java
try {
    int a = 10 / 0;
} catch (ArithmeticException e) {
    System.out.println("Caught: " + e);
} finally {
    System.out.println("Cleanup: finally block runs always");
}
```

---

4. throw vs throws

throw: Used inside a method to actually throw an exception.

```
throw new ArithmeticException("Manually thrown");
```

throws: Declares exceptions in method signature, letting caller handle.

```
void risky() throws IOException {
    throw new IOException("Checked exception");
}
```

---

5. Checked vs Unchecked Exceptions

Checked Exception: Compile-time checked; must handle. - Example: IOException, SQLException

```
try { FileReader fr = new FileReader("file.txt"); } catch (IOException e) {}
```

Unchecked Exception: Runtime only; optional handling. - Example: ArithmeticException, NullPointerException

```
int a = 10 / 0;
```

---

6. Multiple Catch & Multi-Catch

Multiple Catch: Multiple catch blocks for different exceptions.

```
try { int[] arr = {1,2}; System.out.println(arr[5]); }
catch(ArrayIndexOutOfBoundsException e){}
catch(Exception e){}
```

Multi-Catch (Java 7+): One catch for multiple exceptions.

```
try { int[] arr = {1,2}; System.out.println(arr[5]); }
catch(ArrayIndexOutOfBoundsException | ArithmeticException e){}
```

---

7. Nested try

Definition: try block inside another try block.

```
try {
    int a = 10 / 0;
    try {
        int[] arr = {1,2};
        System.out.println(arr[5]);
    } catch(ArrayIndexOutOfBoundsException e){ }
} catch(ArithmeticException e){ }
```

---

8. Try-with-resources (Java 7+)

Definition: Automatically closes resources implementing AutoCloseable.

```
try (FileReader fr = new FileReader("test.txt")) {
    int c;
    while((c=fr.read())!=-1) System.out.print((char)c);
} catch(IOException e){}
```

---

9. Custom Exceptions

Checked Custom Exception:

```
class MyCheckedException extends Exception {
    MyCheckedException(String msg){ super(msg); }
}
```

Unchecked Custom Exception:

```
class MyUncheckedException extends RuntimeException {
    MyUncheckedException(String msg){ super(msg); }
}
```

Interview Questions & Answers

1. **Difference between throw and throws?**
2. throw: inside method to throw exception.
3. throws: declares exceptions in method signature.
4. **Checked vs Unchecked Exceptions?**
5. Checked: compile-time, must handle.
6. Unchecked: runtime, optional.
7. **Can constructor throw exceptions?**
8. Yes, can declare throws and throw exceptions.
9. **Can we have try without catch?**
10. Yes, if finally is present.
11. **Purpose of finally?**
12. Cleanup code, always runs (unless JVM exits).
13. **final vs finally vs finalize()?**
14. final: keyword (constant, no override, no inheritance)
15. finally: block for cleanup
16. finalize(): called by GC before object destruction
17. **What is try-with-resources?**
18. Automatically closes resources implementing AutoCloseable.
19. **Multiple catch blocks?**
20. Allowed, child exceptions must come before parent.

21. **Multi-catch?**
22. Catch multiple unrelated exceptions in one catch block.
23. **If exception not handled?**
24. Program terminates with stack trace.
25. **Can we rethrow exception?**
26. Yes, either same or new exception.
27. **Base class of all exceptions?**
28. Throwable
29. **Should Errors be handled?**
30. No, not recoverable.
31. **Multiple try blocks?**
32. Yes, independent or nested.
33. **Override method that throws exception?**
34. Child method can throw narrower or fewer exceptions.

This is a **complete cheat sheet with definitions, examples, and answers** for Exception Handling in Java.