**1. Explain different types of Errors in Java**Errors in Java are severe issues that occur during the runtime of a program, which the program usually cannot recover from.  
Types of Errors in Java:

* **OutOfMemoryError**: Occurs when the JVM runs out of memory.
* **StackOverflowError**: Happens when a recursive method call goes out of control, exhausting the stack space.
* **LinkageError**: Arises when there is a problem with loading a class, such as a missing dependency.
* **VirtualMachineError**: Related to the JVM, such as internal resource failures.

**2. What is an Exception in Java?**An exception in Java is an event that disrupts the normal flow of a program during execution. Unlike errors, exceptions are conditions that a program can handle, such as invalid user input or a failed network connection.

**3. How can you handle exceptions in Java? Explain with an example.**Exceptions in Java are handled using **try**, **catch**, **finally**, and **throw/throws** blocks.

* **try block**: Contains the code that might throw an exception.
* **catch block**: Handles the exception.
* **finally block**: Executes cleanup code, regardless of whether an exception occurred.

**Example**:

java

public class ExceptionExample {

public static void main(String[] args) {

try {

int result = 10 / 0; // ArithmeticException

} catch (ArithmeticException e) {

System.out.println("Exception caught: Division by zero.");

} finally {

System.out.println("Finally block executed.");

}

}

}

**4. Why do we need exception handling in Java?**

* To prevent abrupt termination of a program.
* To handle runtime issues (e.g., invalid inputs, file not found).
* To improve program reliability and user experience.
* To allow developers to isolate the problem and fix it without crashing the entire application.

**5. What is the difference between exception and error in Java?**

| **Aspect** | **Exception** | **Error** |
| --- | --- | --- |
| **Definition** | A condition that can be handled. | A serious issue beyond recovery. |
| **Recovery** | Recoverable using try-catch blocks. | Usually not recoverable. |
| **Type** | Subclass of Throwable. | Subclass of Throwable. |
| **Examples** | IOException, SQLException. | OutOfMemoryError, StackOverflowError. |

**6. Name the different types of exceptions in Java.**

* **Checked Exceptions**: Must be handled during compilation. Examples: IOException, SQLException.
* **Unchecked Exceptions**: Occur during runtime, not checked at compile time. Examples: NullPointerException, ArithmeticException.

**7. Can we just use try instead of finally and catch blocks?**No, we cannot use try alone. A try block must always be followed by at least one of the following:

* A **catch block** to handle the exception.
* A **finally block** to execute code regardless of exceptions.

**Example of Illegal Use**:

java

try {

System.out.println("This will cause a compilation error.");

}

// No catch or finally block

**Correct Usage**:

java

try {

int num = 5 / 0;

} finally {

System.out.println("Finally block is mandatory if no catch.");

}

4o