🌈 EXCEPTION HANDLING IN JAVA – COMPLETE MASTER NOTES

🟢 1. Introduction – Why Exception Handling?

When we write a program, we expect it to run smoothly. But sometimes unexpected problems occur during execution. These problems are called **exceptions**.

Without exception handling:

- Program crashes suddenly
- Remaining code does not execute
- User experience becomes poor

With exception handling:

- Program handles errors gracefully
- Execution continues safely
- Software becomes stable and professional

🔵 2. Types of Errors in Java

🧱 Compile-Time Errors

- Syntax mistakes
- Wrong datatype usage
- Missing semicolon
- Detected by compiler

Example:

```
int x = "Hello";
```

💥 Runtime Errors (Exceptions)

- Occur during program execution
- Detected by JVM
- May terminate program if not handled

Example:

```
int x = 10 / 0;
```

🟣 3. What is an Exception?

Definition:

An Exception is an object that represents an abnormal condition occurring at runtime that disrupts the normal flow of execution.

Important Points:

- Exception is an OBJECT
- Created at runtime
- Handled using try-catch mechanism

🏗️ 4. Exception Hierarchy

```
Object

 ↓

Throwable

 ↓

-------------------------

|                 |

Exception          Error
```

🔴 Error

- Serious system-level problems
- Not meant to be handled
- Example: OutOfMemoryError, StackOverflowError

🔵 Exception

- Problems that applications can handle
- Caused by program logic

🟡 5. Types of Exceptions

```
Exception

  |

  |---- Checked Exceptions

  |

  |---- RuntimeException (Unchecked)
```

🟢 Checked Exceptions

- Checked at compile time
- Compiler forces handling
- Must use try-catch or throws

Examples:

- IOException
- SQLException
- InterruptedException

🔴 Unchecked Exceptions (RuntimeException)

- Occur at runtime
- Compiler does NOT force handling
- Usually caused by programming mistakes

Examples:

- ArithmeticException
- NullPointerException
- ArrayIndexOutOfBoundsException

🛠 6. Default Exception Handling (JVM)

If an exception is not handled:

1. JVM creates exception object
2. JVM prints stack trace
3. Program terminates

Example Output:

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
```

🟢 7. try-catch Block

Structure:

```
try {
  // risky code
}
```

```
catch (ExceptionType e) {
  // handling code
}
```

Meaning:

- try → Risky code
- catch → Handling code

🔵 8. Multiple Catch Blocks

Rules:

- Child exception must come before parent exception
- Only one catch block executes
- Order matters

🟣 9. finally Block

Used for cleanup operations:

- Closing files
- Closing database connections
- Releasing resources

Structure:

```
try { }
catch() { }
finally {
  // always executes
}
```

Finally executes:

- Whether exception occurs or not
- Even if return statement exists

Does NOT execute only if:

- JVM crashes
- System.exit() is called

🟠 10. Exception Propagation (Stack Mechanism)

When exception occurs inside a method:

- JVM checks current method for handler
- If not found, method is removed from stack
- Control moves to calling method

This process is called:

- Stack Unwinding
- Exception Propagation

🔥 11. throw Keyword

Used to manually create and throw an exception.

Example:

```
throw new ArithmeticException("Invalid operation");
```

Used inside method body.

🔥 12. throws Keyword

Used in method declaration.

Example:

```
void display() throws IOException
```

Meaning:
"This method may throw this exception."

🟢 13. Custom Exception

We can create our own exception to represent business logic errors.

Example:

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

Used for:

- Invalid login
- Insufficient balance
- Invalid input

🟣 14. Rules of Exception Handling

- Exception is an object
- Only objects of Throwable can be thrown
- try must be followed by catch or finally
- One try can have multiple catch blocks
- Parent catch must be last
- Checked exceptions must be handled

🟡 15. Method Overriding Rules with Exceptions

When overriding a method:

Child class can:

- Throw same exception
- Throw child exception
- Throw no exception

Child class cannot:

- Throw broader checked exception

🌍 16. Real-World Usage of Exception Handling

Used in:

- Banking systems
- Payment gateways
- File handling
- Database operations
- Networking
- Multithreading

🌟 17. Best Practices

- Do not leave catch block empty
- Do not catch generic Exception unnecessarily
- Always print meaningful message
- Use custom exceptions for business logic
- Always close resources in finally

🏆 18. Final Summary

Exception handling is a mechanism to manage runtime errors in a structured way so that:

- Programs do not crash suddenly
- Errors are handled safely
- Code becomes clean and maintainable
- Applications become robust and professional

🎯 One-Line Understanding

Exception Handling means:

"When something unexpected happens during program execution, the program knows how to react safely instead of crashing."