EXCEPTION HANDLING – PART 1

(Introduction + Why Needed + Basic try–catch)

1️⃣ EXCEPTION KYA HOTI HAI? (Simple Definition)

Exception = Runtime par aane wali error jo program ko crash kar sakti hai.
Jaise:

- 10 ÷ 0
- Null object par method call
- Galat file path
- Array index out of range

👉 Ye sab **compile time par nahi**, balki **runtime par aati hain** → isliye inhe **EXCEPTIONS** kehte hain.

2️⃣ BINA EXCEPTION HANDLING KE KYA HOTA HAI?

```
public class Test {
    public static void main(String[] args) {
        int a = 10;
        int b = 0;

        int c = a / b;   // ❌ crash
        System.out.println(c);

        System.out.println("Program end");
    }
}
```

Output:

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

👉 `"Program end"` print hi **nahi hoga**
👉 Program **beech me crash ho gaya**

3️⃣ EXCEPTION HANDLING KYU ZAROORI HAI?

Exception Handling se:

✅ Program crash nahi hota
✅ Program safe rehta hai
✅ User ko proper message milta hai
✅ Application professional dikhti hai
✅ Backend APIs safe hoti hain

4️⃣ TRY–CATCH BLOCK (Basic Syntax)

```
try {
    // risky code
} catch (Exception e) {
    // handling code
}
```

5️⃣ SAME PROGRAM WITH TRY–CATCH

```
public class Test {
    public static void main(String[] args) {

        int a = 10;
        int b = 0;

        try {
            int c = a / b;   // error yahan aayegi
            System.out.println(c);
        } catch (Exception e) {
            System.out.println("Error aaya lekin program crash nahi
hua");
        }

        System.out.println("Program end");
    }
}
```

Output:

```
Error aaya lekin program crash nahi hua
Program end
```

✅ Ab program crash nahi hua
✅ Last line bhi chal gayi
✅ Ye hi **Exception Handling ka main fayda hai**

6️⃣ EXCEPTION OBJECT (e)

Catch block me jo e hota hai wo:

- Error ka **poora detail** rakhta hai
- Error ka **type** batata hai
- Line number bhi batata hai

Example:

```
catch (Exception e) {
    System.out.println(e);
}
```

Output:

```
java.lang.ArithmeticException: / by zero
```

7️⃣ printStackTrace() (Interview Favourite)

```
catch (Exception e) {
    e.printStackTrace();
}
```

✅ Ye poora error detail deta hai:

- Class name
- Line number
- Error type

Backend developers ise **logs ke liye** use karte hain.

8️⃣ REAL LIFE EXAMPLE (Interview Level)

Socho:

- ATM se paisa nikaal rahe ho
- Balance zero hai

Agar system crash ho jaaye ❌
Par agar message aaye:
"Insufficient Balance" ✅
👉 Ye hi **Exception Handling** hai real life me.

EXCEPTION HANDLING – PART 2

Types of Exceptions (Checked, Unchecked, Error)

1️⃣ JAVA ME EXCEPTIONS KE 3 MAIN TYPES

Java me **3 category** hoti hain:

Type	Kab aati hai	Handle karna compulsory?
✅ Checked Exception	Compile time par	✅ Haan

✅ Unchecked Exception	Runtime par	❌ Nahi (But recommended)
❌ Error	JVM related	❌ Handle nahi karte

2️⃣ CHECKED EXCEPTION (Compile Time Exception)

👉 Ye wo exception hoti hai jo **compile hone se pehle hi Java bol deta hai:**
"Isko handle karo, warna code chalega hi nahi."

Example 1: File Not Found

```
import java.io.FileReader;

public class Test {
    public static void main(String[] args) throws Exception {
        FileReader fr = new FileReader("abc.txt");  // ❌ Checked
Exception
    }
}
```

❌ Agar tum `try-catch` ya `throws` nahi lagao ge → **compile error aayega**

Correct Way (try–catch)

```
import java.io.FileReader;

public class Test {
    public static void main(String[] args) {
        try {
            FileReader fr = new FileReader("abc.txt");
        } catch (Exception e) {
            System.out.println("File nahi mili");
        }
    }
}
```

✅ Ye hai **Checked Exception handling**

Common Checked Exceptions

- `IOException`
- `FileNotFoundException`
- `SQLException`
- `ClassNotFoundException`

3️⃣ UNCHECKED EXCEPTION (Runtime Exception)

👉 Ye wo exception hoti hai jo **program chalne ke baad aati hai**
👉 Java isko **force nahi karta handle karne ke liye**

Example 1: Divide by Zero

```
int a = 10;
int b = 0;
int c = a / b;   // ❌ ArithmeticException
```

👉 Ye **Runtime Exception hai**

Example 2: NullPointerException

```
String name = null;
System.out.println(name.length());  // ❌ NullPointerException
```

Example 3: ArrayIndexOutOfBoundsException

```
int[] a = {10,20,30};
System.out.println(a[5]);  // ❌ Out of range
```

Common Unchecked Exceptions

- `ArithmeticException`
- `NullPointerException`
- `ArrayIndexOutOfBoundsException`
- `NumberFormatException`

4️⃣ ERROR (Very Dangerous)

👉 Error wo problems hoti hain jo **JVM level par hoti hain**, application level par nahi.

Example:

- `OutOfMemoryError`
- `StackOverflowError`
- `VirtualMachineError`

Inhe **hum normally handle nahi karte**

5️⃣ CHECKED vs UNCHECKED (Interview Favourite Table)

Feature	Checked	Unchecked
Kab aati	Compile Time	Runtime
Handle Compulsory?	✅ Yes	❌ No
Example	IOException	NullPointerException
Package	java.lang + others	java.lang
Real Use	File, DB	Logic mistakes

COMPILE TIME vs RUNTIME (Simple Language)

Compile Time Exception

➞ Code likhte hi error

➞ Program chalne nahi deta

Runtime Exception

➞ Program chal raha hota hai

➞ Beech me crash karta hai

EXCEPTION HANDLING – PART 3

Multiple Catch, finally, try–catch–finally Flow

1️⃣ MULTIPLE CATCH BLOCK

👉 Jab **ek hi try block me multiple type ke exception aane ka chance ho**, tab hum **multiple catch** use karte hain.

Example 1: Multiple Catch

```
public class Test {
    public static void main(String[] args) {

        try {
            int a = 10 / 0;          // ArithmeticException
            String s = null;
            System.out.println(s.length()); // NullPointerException
```

```
        } catch (ArithmeticException e) {
            System.out.println("Divide by zero error");

        } catch (NullPointerException e) {
            System.out.println("Null value ka error");

        } catch (Exception e) {
            System.out.println("General exception");
        }
    }
}
```

Rules (Interview Very Important):

1️⃣ **Specific exception upar likhte hain**
2️⃣ **General exception (Exception) last me likhte hain**
❌ Agar `Exception` upar likh diya → **compile error aayega**

2️⃣ MULTI-CATCH (Java 7+ Feature)

👉 Ek hi catch me **multiple exception handle kar sakte ho**

```
try {
    int a = 10 / 0;
    String s = null;
    System.out.println(s.length());

} catch (ArithmeticException | NullPointerException e) {
    System.out.println("Multiple exception handled");
}
```

Isse code **short & clean** ho jaata hai
Interview me ye feature pucha jaata hai

3️⃣ FINALLY BLOCK

👉 `finally` block **hamesha chalega**
Chahe exception aaye ya na aaye.

Use:

- File close karne ke liye
- DB connection close karne ke liye
- Scanner close karne ke liye

Example with finally

```
public class Test {
    public static void main(String[] args) {
```

```
        try {
            int a = 10 / 2;
            System.out.println(a);

        } catch (Exception e) {
            System.out.println("Error");

        } finally {
            System.out.println("Ye finally hamesha chalega");
        }
    }
}
```

Output:

```
5
Ye finally hamesha chalega
```

Example with Exception + finally

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

} catch (Exception e) {
    System.out.println("Error aaya");

} finally {
    System.out.println("Finally block executed");
}
```

Output:

```
Error aaya
Finally block executed
```

✅ Finally **hamesha chalega**

4️⃣ TRY–CATCH–FINALLY FLOW (Execution Order)

Case 1: ❌ Exception aati hai

```
try → catch → finally
```

Case 2: ✅ Exception nahi aati

```
try → finally
```

5️⃣ AGAR catch me bhi exception aa jaaye?

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

} catch (Exception e) {
    int b = 10 / 0;  // ❌ yahan bhi exception

} finally {
    System.out.println("Finally chalega hi chalega");
}
```

Output:

```
Finally chalega hi chalega
Exception in thread main ...
```

Matlab:

- Finally chala
- Uske baad program crash

6️⃣ finally block kab nahi chalta? (Rare Interview Question)

`finally` **sirf ek case me nahi chalta**:

👉 `System.exit(0);` ke case me

```
try {
    System.exit(0);
} finally {
    System.out.println("Ye print nahi hoga");
}
```

EXCEPTION HANDLING – PART 4

throws Keyword (Checked Exception Forward Karna)

1️⃣ throws KYA HOTA HAI?

👉 throws ka matlab hota hai:

“Main is exception ko yahan handle nahi kar raha,
main isse upar wale method ko bhej raha hoon.”

✅ Ye **Checked Exception** ke saath zyada use hota hai
✅ Program crash se bachta hai
✅ Responsibility **caller method** par chali jaati hai

2️⃣ BINA `throws` KE CHECKED EXCEPTION (Error AAYEGA)

```
import java.io.FileReader;

public class Test {
    public static void main(String[] args) {
        FileReader fr = new FileReader("abc.txt"); // ❌ Compile Time
Error
    }
}
```

❌ Error aayega:

```
Unhandled exception: java.io.FileNotFoundException
```

3️⃣ `throws` LAGA KAR SAHI TARIKA

```
import java.io.FileReader;

public class Test {

    public static void main(String[] args) throws Exception {
        FileReader fr = new FileReader("abc.txt");  // ✅ Now OK
    }
}
```

✅ Ab compile ho jaayega
✅ Exception **JVM ko forward ho gayi**

4️⃣ METHOD LEVEL PAR `throws`

```
import java.io.FileReader;

public class Test {

    static void readFile() throws Exception {
        FileReader fr = new FileReader("abc.txt");
```

```
    }

    public static void main(String[] args) throws Exception {
        readFile();
    }
}
```

✅ Exception flow:
`readFile() → main() → JVM`

5️⃣ REAL LIFE EXAMPLE (INTERVIEW LEVEL)

Socho:

- **Controller → Service → Repository**
- Service me DB error aaya
- Service `throws` karke Controller ko bhej raha hai

```
// Service
public Employee getEmployee(int id) throws Exception {
    return repository.findById(id).orElseThrow(() -> new
Exception("Not Found"));
}

// Controller
@GetMapping("/emp/{id}")
public Employee getEmp(@PathVariable int id) throws Exception {
    return service.getEmployee(id);
}
```

✅ Yahan:

- Service **throws** kar rahi hai
- Controller **handle kar sakta hai ya Global Exception ko bhej sakta hai**

6️⃣ `throws` vs `try-catch` (INTERVIEW FAVOURITE)

Feature	try–catch	throws
Handling yahin hoti?	✅ Yes	❌ No
Exception forward hoti?	❌ No	✅ Yes
Code ka control	Yahin	Upar wale method ko
Use kab?	Jab yahin solution ho	Jab higher layer handle kare

Spring Boot project me mostly `throws` + Global Exception use hota hai

7️⃣ EK SAATH MULTIPLE `throws`

```
public void test() throws IOException, SQLException {
    // risky code
}
```

Matlab:
Ye method **2 tarah ki exceptions forward kar sakta hai**

8️⃣ AGAR `throws` KE SAATH BHI EXCEPTION AAYI?

Agar:

- tumne `throws` lagaya
- aur upar bhi handle nahi kiya

👉 JVM program **crash kar dega**

EXCEPTION HANDLING – PART 5

Custom / User Defined Exception (Deep Detail)

1️⃣ CUSTOM EXCEPTION KYA HOTI HAI?

👉 Jab **Java ki built-in exceptions aapki business requirement ko clearly express na kar paaye**,
tab hum **apni khud ki exception banate hain** → isi ko **Custom Exception** kehte hain.

Example Real Life:

- “Employee Not Found”
- “Duplicate Email”
- “Insufficient Balance”
- “Invalid Age”

Ye sab Java ki default exceptions se **clear nahi hota**, isliye hum **custom exception** banate hain.

2️⃣ CUSTOM EXCEPTION BANANE KA RULE

Custom exception banane ke liye:

Rule:

```
class MyException extends Exception
```

ya

```
class MyException extends RuntimeException
```

Do type hoti hain:

Type	extends

✅ Checked Custom Exception	`Exception`
✅ Unchecked Custom Exception	`RuntimeException`

✅ **Spring Boot me mostly RuntimeException use hota hai**

3️⃣ CHECKED CUSTOM EXCEPTION EXAMPLE

Step 1: Custom Exception Class Banao

```
class InvalidAgeException extends Exception {

    public InvalidAgeException(String message) {
        super(message);
    }
}
```

Step 2: Use in Program

```
public class Test {

    static void validateAge(int age) throws InvalidAgeException {

        if (age < 18) {
            throw new InvalidAgeException("Age 18 se kam hai");
        } else {
            System.out.println("Valid age");
        }
    }

    public static void main(String[] args) throws InvalidAgeException
{
        validateAge(15);
    }
}
```

Output:

```
Exception in thread "main" InvalidAgeException: Age 18 se kam hai
```

Yahan:

- `throw` → exception throw karta hai
- `throws` → method ke signature me likhte hain

4️⃣ UNCHECKED CUSTOM EXCEPTION (REAL PROJECT STYLE)

Spring Boot + Backend ke liye **ye hi standard hai**

Step 1: Custom Exception Class

```
public class EmployeeNotFoundException extends RuntimeException {

    public EmployeeNotFoundException(String message) {
        super(message);
    }
}
```

Step 2: Use in Service Layer

```
public Employee getEmployee(int id) {

    Employee emp = repository.findById(id)
            .orElseThrow(() -> new EmployeeNotFoundException("Employee
not found with id: " + id));

    return emp;
}
```

Agar employee nahi mila:

- Java automatically **EmployeeNotFoundException throw karega**

5️⃣ throw vs throws (Interview Favourite)

throw	throws
Exception ko **manually throw karta hai**	Method ke signature me likhte hain
Method ke andar	Method ke bahar
Ek hi exception	Multiple ho sakti
`throw new` use hota	`throws` keyword use hota

Example:

```
throw new EmployeeNotFoundException("Employee nahi mila");

public void getData() throws SQLException, IOException
```

6️⃣ REAL SPRING BOOT FLOW (INTERVIEW GOLD 🔥)

```
Controller → Service → Repository
                 |
                 ↓
        Custom Exception throw
                 |
                 ↓
       Global Exception Handler
```

Matlab:

- Service exception throw karegi
- Controller me try–catch nahi likho ge
- Global handler automatically client ko error bhej dega

7️⃣ Custom Exception kyun use karte hain?

✅ Clear error message
✅ Industry standard
✅ REST API me proper response codes
✅ Debugging easy
✅ Interview me strong impact

EXCEPTION HANDLING – PART 6

GLOBAL EXCEPTION HANDLING (Spring Boot – Production Level)

1️⃣ GLOBAL EXCEPTION HANDLING KYA HOTI HAI?

👉 Jab hum **poori application ke liye ek hi jagah par sab exceptions handle karte hain**, use **Global Exception Handling** kehte hain.

✅ Matlab:

- Har controller me `try-catch` likhne ki zarurat nahi
- Clean code
- Proper HTTP Status Codes
- Professional API response

2️⃣ GLOBAL HANDLER BANANE KE LIYE 2 MAIN ANNOTATIONS

Annotation	Kaam
`@ControllerAdvice`	Global exception handler class banata hai
`@ExceptionHandler`	Kaunsi exception handle karni hai ye batata hai

3️⃣ STEP BY STEP – GLOBAL EXCEPTION HANDLER BANANA

File kaha banani hai?

```
src/main/java
 → exception
     → GlobalExceptionHandler.java ✅
```

4️⃣ BASIC GLOBAL HANDLER CLASS

```
package com.app.ems.exception;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;

@ControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(Exception.class)
    public ResponseEntity<String> handleGeneralException(Exception ex)
{
        return new ResponseEntity<>(ex.getMessage(),
HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

Ab:

- Puri application me koi bhi `Exception` aayi
- Ye method automatically handle karega
- Client ko `500 INTERNAL SERVER ERROR` milega

5️⃣ CUSTOM EXCEPTION KE SAATH GLOBAL HANDLER (REAL PROJECT STYLE)

Tumne abhi padha tha:

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

Ab isko global handler me handle karte hain:

```
@ExceptionHandler(EmployeeNotFoundException.class)
public ResponseEntity<String>
handleEmployeeNotFound(EmployeeNotFoundException ex) {
    return new ResponseEntity<>(ex.getMessage(),
HttpStatus.NOT_FOUND);
}
```

✅ Ab jab bhi:

```
throw new EmployeeNotFoundException("Employee not found");
```

✅ Response hoga:

```
Status: 404 NOT FOUND
Body: Employee not found
```

🔥 Ye **industry standard REST API behavior** hai.

6️⃣ VALIDATION EXCEPTION HANDLE KARNA (@Valid)

Jab tum aage chal kar `@NotNull`, `@Email` use karoge, tab:

```
@ExceptionHandler(MethodArgumentNotValidException.class)
public ResponseEntity<String>
handleValidation(MethodArgumentNotValidException ex) {
    return new ResponseEntity<>("Validation failed",
HttpStatus.BAD_REQUEST);
}
```

✅ HTTP Status: **400 BAD REQUEST**

7️⃣ PROFESSIONAL ERROR RESPONSE FORMAT (JSON)

Real API me response aise hota hai:

```
{
  "timestamp": "2025-11-26",
  "status": 404,
  "error": "NOT_FOUND",
  "message": "Employee not found",
  "path": "/api/employees/10"
}
```

Iske liye hum ek class banate hain:

```
public class ErrorResponse {
    private int status;
    private String message;
    private String path;

    // getters & setters
}
```

Aur handler me return karte hain:

```
@ExceptionHandler(EmployeeNotFoundException.class)
public ResponseEntity<ErrorResponse>
handleEmployeeNotFound(EmployeeNotFoundException ex,
HttpServletRequest request) {

    ErrorResponse error = new ErrorResponse();
    error.setStatus(404);
    error.setMessage(ex.getMessage());
    error.setPath(request.getRequestURI());

    return new ResponseEntity<>(error, HttpStatus.NOT_FOUND);
}
```

✅ Ye **production-grade API response** hota hai.

8️⃣ GLOBAL EXCEPTION HANDLING KYUN ZAROORI HAI?

- ✅ Clean Controllers
- ✅ No repeated try–catch
- ✅ Proper HTTP status codes
- ✅ Client ko clear error
- ✅ Debugging easy
- ✅ Interview me strong impression

9️⃣ INTERVIEW QUESTIONS DIRECT FROM THIS TOPIC

- ✅ Difference between `@ControllerAdvice` and `@ExceptionHandler`
- ✅ What is Global Exception Handling?
- ✅ How do you handle custom exceptions globally in Spring Boot?
- ✅ How do you return proper HTTP status codes in exception handling?