Chapter-8: Exception Handling

PART-II: Sample Short Questions & Answers

1. What are Java exceptions?

An exception is an event that disrupts the normal flow of a program during execution. It represents errors like invalid input, division by zero, or file not found.

👉 Example:

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

---

2. How to handle exceptions in Java?

Exceptions are handled using **try-catch-finally** blocks. The risky code is inside try, and errors are handled in catch.

👉 Example:

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

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

---

3. What is difference between exceptions and error in Java?

- **Exception:** Recoverable problems (e.g., FileNotFoundException, NullPointerException).
- **Error:** Irrecoverable problems (e.g., OutOfMemoryError, StackOverflowError).

---

4. What are checked and unchecked exceptions in Java?

- **Checked:** Checked at compile time (e.g., IOException, SQLException).
- **Unchecked:** Checked at runtime (e.g., ArithmeticException, NullPointerException).

---

5. Why the term finally used for?

finally block ensures execution of code regardless of exception (commonly used for closing files or database).

👉 Example:

```
finally { System.out.println("Always executes"); }
```

---

6. Give some examples of unchecked exceptions.

- ArithmeticException
- NullPointerException
- ArrayIndexOutOfBoundsException

---

7. Give some examples of checked exceptions.

- IOException
- FileNotFoundException
- SQLException

---

8. Can we throw an exception manually?

Yes, using the throw keyword we can manually throw exceptions.

👉 Example:

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

---

9. What is Java multicatch exception?

Since Java 7, multiple exceptions can be caught in a single catch block using |.

👉 Example:

```
catch(IOException | SQLException e) { ... }
```

---

10. What is meant by print exception in Java?

Exceptions can be printed using printStackTrace(), getMessage(), or toString().

👉 Example:

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

PART-III: Sample Long Questions & Answers

1. Briefly describe Exception handling in Java and its types.

- Exception handling ensures normal execution even when runtime errors occur.
- **Types of Exceptions:**
  1. **Checked:** Compile-time errors (e.g., IOException).
  2. **Unchecked:** Runtime errors (e.g., NullPointerException).
  3. **Errors:** System failures (e.g., OutOfMemoryError).

     👉 Example:

```
try { int a = 5/0; }
catch(ArithmeticException e){ System.out.println("Cannot divide by zero"); }
```

---

2. Briefly describe with example the print exception messages in Java.

There are three main ways:

- getMessage() → returns exception detail.
- toString() → returns name + description.
- printStackTrace() → prints full error with line number.

  👉 Example:

```
try { int a = 5/0; }
catch(Exception e){
  System.out.println(e.getMessage());
  e.printStackTrace();
}
```

---

3. Explain the hierarchy of Java exception.

- **Throwable** → Parent class of all errors/exceptions.
    - **Error** → Serious issues (OutOfMemoryError).
    - **Exception** → Recoverable problems.
        - **Checked Exceptions** (IOException, SQLException).
        - **Unchecked Exceptions** (ArithmeticException, NullPointerException).

👉 Hierarchy: Object → Throwable → Exception/Error

---

4. Briefly explain the Java Catch Multiple Exceptions.

- From Java 7, one catch block can handle multiple exceptions using |.
- This avoids code duplication.
  👉 Example:

```
try {
  int arr[] = new int[5];
  arr[5] = 10;
}
catch(ArrayIndexOutOfBoundsException | ArithmeticException e) {
  System.out.println("Error: " + e);
}
```

---

5. Briefly explain how to handle exceptions in Java.

There are **five keywords** in Java exception handling:

1. try → risky code.
2. catch → handles exceptions.
3. finally → always executes.
4. throw → manually throw exception.
5. throws → declare exceptions in method signature.

👉 Example:

```
void readFile() throws IOException {
  FileReader fr = new FileReader("data.txt");
}
```