Day-20

Exception Handling

Exception

- An **exception** is an unwanted or unexpected event that occurs during the execution of a program, which disrupts the normal flow of instructions.
- Exceptions can happen due to various reasons like invalid input, file not found, or trying to divide by zero.

Exception Handling

- **Exception handling** is a mechanism in Java to manage runtime exceptions and maintain the normal flow of the application.
- Java provides a way to catch and handle exceptions so that the program doesn't crash.
- The most common approach is using **try, catch**, and **finally** blocks.

Types of Exceptions in Java

Java exceptions are broadly classified into two categories:

- **Checked Exceptions**:
  - Occur at compile-time.
  - Must handle them using a **try-catch** block or by declaring them using **throws**.
  - Example: IOException, FileNotFoundException
- **Unchecked Exceptions**:
  - Occur at runtime.
  - Can be handled using **try-catch**.
  - Example: NullPointerException, ArithmeticException, ArrayIndexOutOfBoundsException

Handling Unchecked Exceptions Using Try-Catch Block

Example:

```java
public class UncheckedExceptionDemo {
    public static void main(String[] args) {
        // ArithmeticException
        try {
            int result = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Cannot divide by zero!");
        }

        // NullPointerException
        try {
            String str = null;
            System.out.println(str.length());
        } catch (NullPointerException e) {
            System.out.println("Null pointer exception occurred!");
        }

        // NumberFormatException
        try {
            String num = "abc";
            int value = Integer.parseInt(num);
        } catch (NumberFormatException e) {
            System.out.println("Invalid number format!");
        }

        // ArrayIndexOutOfBoundsException
        try {
            int[] arr = new int[3];
            System.out.println(arr[5]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array index is out of bounds!");
        }
    }
}
```

Multiple Catch Blocks

You can have multiple catch blocks to handle different exceptions separately.

Example:

```java
public class MultipleCatchDemo {
    public static void main(String[] args) {
        try {
            int[] arr = new int[3];
            System.out.println(arr[5]);
            int result = 10 / 0;
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array index error!");
        } catch (ArithmeticException e) {
            System.out.println("Arithmetic error!");
        }
    }
}
```

Finally Block

The **finally** block is always executed, whether an exception occurs or not. It is typically used for cleaning up resources like closing files or releasing memory.

Example:

```java
public class FinallyBlockDemo {
    public static void main(String[] args) {
        try {
            int result = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Exception caught!");
        } finally {
            System.out.println("This will always execute!");
        }
    }
}
```

Handling Checked Exceptions Using Try-Catch and Throws

Example:

- **Try-Catch**:

```java
import java.io.*;

public class CheckedExceptionDemo {
    public static void main(String[] args) {
        try {
            FileReader file = new FileReader("nonexistent.txt");
        } catch (FileNotFoundException e) {
            System.out.println("File not found exception!");
        }
    }
}
```

- **Throws**:

```java
import java.io.*;

public class ThrowsExample {
    public static void main(String[] args) throws IOException {
        FileReader file = new FileReader("nonexistent.txt");
    }
}
```

“Throw” Keyword

The **throw** keyword is used to manually throw an exception.

Example:

```
public class ThrowKeyword {

    void findSquare(int num) {
        if (num < 1) {
            throw new ArithmeticException("Number is negative, cannot calculate square.");
        } else {
            System.out.println(num * num);
        }
    }

    public static void main(String[] args) {
        ThrowKeyword tk = new ThrowKeyword();
        try {
            tk.findSquare (-1);
        } catch (ArithmeticException e) {
            System.out.println("Data is not valid.");
        }
    }
}
```

Difference Between Throws and Throw

- **Throw**:
  - Used to explicitly throw an exception.
  - Example: throw new ArithmeticException("error message");
- **Throws**:
  - Used in method declarations to specify that a method might throw an exception.
  - Example: public void method() throws IOException { }

Java Exceptions Keywords summary:

Keyword	Description
try	Used to define a block of code that might throw an exception. It is followed by either catch, finally, or both.
catch	Used to catch exceptions thrown by the try block. It defines the block of code to be executed if an exception occurs.
finally	A block that always executes after the try block, whether an exception is handled or not. Commonly used for cleanup code.
throw	Used to explicitly throw an exception from a method or block of code. It is followed by an instance of Throwable.
throws	Used in the method signature to declare exceptions that a method might throw. It is followed by the exception type(s).
Throwable	The superclass of all exceptions and errors. All exception classes derive from this class.

Quiz - Exception Handling

1. **What is an exception in Java?**
   - a) A compile-time error
   - b) A runtime error
   - c) A logical error
   - d) A syntax error
2. **Which of the following is the parent class of all exceptions in Java?**
   - a) Throwable
   - b) Exception
   - c) Error
   - d) RuntimeException
3. **Which keyword is used to handle exceptions in Java?**
   - a) catch
   - b) throw
   - c) try
   - d) finally
4. **What happens if an exception is not caught in the program?**
   - a) The program continues normally
   - b) The program terminates abruptly
   - c) The exception is ignored
   - d) The exception is automatically resolved
5. **Which of the following exceptions are checked exceptions?**
   - a) ArithmeticException
   - b) FileNotFoundException
   - c) NullPointerException
   - d) ArrayIndexOutOfBoundsException
6. **What is the use of the finally block in Java?**
   - a) To catch exceptions
   - b) To execute code after try/catch regardless of an exception
   - c) To throw an exception

   - d) To skip error handling

7. **Which of the following statements is correct about unchecked exceptions?**
   - a) They are checked at compile-time
   - b) They are checked at runtime
   - c) They must be handled explicitly
   - d) They cannot be handled using try-catch blocks
8. **Which of the following is not an unchecked exception?**
   - a) NullPointerException
   - b) ArrayIndexOutOfBoundsException
   - c) ClassCastException
   - d) IOException
9. **What is the difference between throw and throws in Java?**
   - a) throw is used to explicitly throw an exception, throws declares an exception
   - b) throw declares an exception, throws is used to throw an exception
   - c) They are the same
   - d) None of the above
10. **What does the throws keyword in a method declaration indicate?**
    - a) The method handles the exceptions internally
    - b) The method can throw one or more exceptions
    - c) The method does not throw any exceptions
    - d) The method is free from exceptions
11. **Which of these is a common unchecked exception?**
    - a) IOException
    - b) FileNotFoundException
    - c) NullPointerException
    - d) InterruptedException
12. **Which block of code must always execute, even if an exception is not caught?**
    - a) catch block
    - b) try block
    - c) finally block
    - d) throw block

13. **What happens if the finally block includes a return statement?**
    - a) The method exits before executing the return statement in finally
    - b) The return statement in the try or catch block is overridden
    - c) Both return statements execute
    - d) The method runs normally without executing the finally block

True/False Questions

14. **You can have multiple catch blocks to handle different exceptions in Java.**
    - True / False
15. **A method must always handle an exception if it declares it using the throws keyword.**
    - True / False

Answers:

1. b) A runtime error
2. a) Throwable
3. c) try
4. b) The program terminates abruptly
5. b) FileNotFoundException
6. b) To execute code after try/catch regardless of an exception
7. b) They are checked at runtime
8. d) IOException
9. a) throw is used to explicitly throw an exception, throws declares an exception
10. b) The method can throw one or more exceptions
11. c) NullPointerException
12. c) finally block
13. b) The return statement in the try or catch block is overridden
14. True
15. False