Exception Handling - Throw and Throws

Both `throw` and `throws` are used in Java exception handling, but they serve different purposes. Let's look at each in detail.

---

1. `throw` Keyword

- **Definition**: Used to explicitly throw an exception in a program.
- **Usage**:
    - It is followed by an instance of an exception.
    - It is typically used within a method or block of code.
- **Syntax**:

    ```
    throw new ExceptionType("Error message");
    ```

Example: Using `throw`

```
public class ThrowExample {
    public static void main(String[] args) {
        int age = 17;

        // Check eligibility for voting
        if (age < 18) {
            throw new IllegalArgumentException("Age must be 1
8 or above to vote.");
        }

        System.out.println("Eligible to vote!");
    }
}
```

- **Explanation**:
  - The program explicitly throws an `IllegalArgumentException` if the `age` is less than 18.
  - This stops the program's execution unless the exception is handled.

---

2. `throws` Clause

- **Definition**: Declares that a method might throw one or more exceptions.
- **Usage**:
  - Added to a method signature.
  - It informs the caller of the method that it must handle the specified exceptions.
- **Syntax**:

```
returnType methodName(parameters) throws ExceptionType1, E
xceptionType2 {
    // Method body
}
```

Example: Using `throws`

```
import java.io.*;

public class ThrowsExample {
    public static void main(String[] args) {
        try {
            readFile("nonexistentfile.txt");
        } catch (IOException e) {
            System.out.println("Exception caught: " + e.getMe
ssage());
        }
    }
    // Method that declares it might throw IOException
    public static void readFile(String fileName) throws IOExc
eption {
        BufferedReader reader = new BufferedReader(new FileRe
ader(fileName));
        System.out.println(reader.readLine());
    }
}
```

- **Explanation**:
    - The `readFile` method declares using `throws` that it might throw an `IOException`.
    - The caller (in this case, `main`) is responsible for handling the exception.

Key Differences Between `throw` and `throws`

Feature	`throw`	`throws`
Purpose	Used to explicitly throw an exception	Used to declare exceptions a method might throw
Location	Used within the method body	Used in the method signature
Followed By	An exception object (e.g., `new ExceptionType`)	Exception class names
Example	`throw new ArithmeticException();`	`void methodName() throws IOException`

How `throw` and `throws` Work Together

These two are often used together to handle exceptions effectively:

1. `throws` informs the caller about the exceptions.
2. `throw` actually triggers the exception.

Example: Combined Usage

```
public class ThrowAndThrowsExample {
    public static void main(String[] args) {
        try {
            validateAge(16);
        } catch (Exception e) {
            System.out.println("Exception caught: " + e.getMe
ssage());
        }
    }

    // Method that throws an exception
    public static void validateAge(int age) throws IllegalArg
umentException {
        if (age < 18) {
            throw new IllegalArgumentException("Age must be 1
8 or above.");
        }
        System.out.println("Valid age!");
    }
}
```

Summary

- `throw`: Used to actually throw an exception.
- `throws`: Used to declare potential exceptions in a method signature.
- Together, they help handle exceptions effectively, ensuring robust error handling and clear communication between methods.