Java Exception Handling: throw vs throws

1. throw Keyword

Purpose: Used to manually throw an exception (built-in or custom).

Usage: Within a method, using 'new' keyword to create an exception object.

Syntax: throw new ExceptionType("Custom message");

Example (Built-in):

```
System.out.println(10 / 0); // Throws ArithmeticException
```

Example (Custom):

```
class VotingAgeException extends RuntimeException {
    public VotingAgeException(String message) {
        super(message);
    }
}

public class Main {
    public static void main(String[] args) {
        int age = 15;
        if (age < 18) {
            throw new VotingAgeException("You are not allowed to vote");
        } else {
            System.out.println("You are allowed to vote");
        }
    }
}
```

2. throws Keyword

Purpose: Declares exceptions a method might throw. Used for checked exceptions.

Usage: In method signature to delegate handling to the caller.

Java Exception Handling: throw vs throws

Syntax:

```
returnType methodName() throws ExceptionType1, ExceptionType2 {
    // code
}
```

Example:

```
public void readFile() throws IOException {
    // file reading logic
}
```

3. Difference Between throw and throws

Feature	throw	throws
Purpose	Explicitly throw exception	Declare exceptions in method signature
Location	Inside method	In method signature
Exceptions	Only one	Multiple can be declared
Example	throw new IOException("Error");	void method() throws IOException {}

4. NumberFormatException Example

Valid:

```
System.out.println(Integer.parseInt("10")); // Output: 10
```

Invalid:

```
System.out.println(Integer.parseInt("ten")); // Throws NumberFormatException
```

Summary

- throw: Manually throws an exception using new.
- throws: Declares exception possibility in method signature.
- Custom exceptions extend RuntimeException or Exception.
- Example: VotingAgeException when age < 18.