`java.util.regex` package contains three main classes/interfaces for processing regular expressions: **`Pattern`**, **`Matcher`**, and **`PatternSyntaxException`** (plus the **`MatchResult`** interface)

Here are the usage examples of the most common and vital methods for each class in the package.

### 1. `java.util.regex.Pattern`

The `Pattern` class represents a compiled regular expression.

- **`compile(String regex)`** / **`compile(String regex, int flags)`**
  Compiles a regular expression string into a reusable `Pattern` object.

```
Pattern pattern = Pattern.compile("\\d+");
```

matcher(CharSequence input)
**Creates a Matcher to match the given input against the pattern.**

```
Matcher matcher = pattern.matcher("Order 1234");
```

matches(String regex, CharSequence input)
**A convenience method to check if the entire input sequence matches the regex.**

```
boolean isMatch = Pattern.matches("\\d+", "12345"); // returns true
```

split(CharSequence input)

**Splits the input sequence around matches of this pattern.**

```
String[] result = pattern.split("a1b2c3");
```

quote(String s)

**Returns a literal pattern string for the specified string (escapes special characters).**

```
String literal = Pattern.compile(Pattern.quote("[abc]")).pattern();
```

## 2. java.util.regex.Matcher

**The Matcher class is the engine that interprets the pattern and performs operations on an input text.**

matches()

**Attempts to match the *entire* input sequence against the pattern.**

```
boolean fullMatch = matcher.matches();
```

find()

**Finds the *next* subsequence of the input that matches the pattern.**

```
while (matcher.find()) {
   System.out.println("Found: " + matcher.group());
}
```

group()

**Returns the input subsequence matched by the previous match operation.**

```
String match = matcher.group();
```

replaceAll(String replacement)

**Replaces every subsequence of the input sequence that matches the pattern.**

```
String updated =
matcher.replaceAll("NUM");
```

start() / end()

**Returns the start or end index of the previous match.**

```
int startIndex = matcher.start();
```

### 3. java.util.regex.PatternSyntaxException

**An unchecked exception thrown to indicate a syntax error in a regular expression pattern.**

getDescription() / getIndex() / getPattern()
**Extracts details about where and why the regex parsing failed.**

```
try {
    Pattern.compile("[Unclosed");
} catch (PatternSyntaxException e) {
    System.out.println("Error: " +
e.getDescription());
    System.out.println("Index: " +
e.getIndex());
}
```