RESUMEN RÁPIDO – java.util.regex, Pattern y Matcher (Java SE)

1. Paquete java.util.regex

- Objetivo: proporcionar clases para compilar y aplicar expresiones regulares.
- Clases clave: Pattern (patrón compilado), Matcher (motor que busca coincidencias).
- Excepción clave: PatternSyntaxException cuando la regex tiene un error.
- Regla general: pasar null produce NullPointerException.
- Flujo mental: “Defino un patrón → creo un matcher → itero con find() o evalúo con matches()”.

2. Clase Pattern

- Crear patrón: Pattern p = Pattern.compile("regex");
- Crear con flags: Pattern.compile("regex", Pattern.CASE_INSENSITIVE | Pattern.MULTILINE)
- Flags útiles:
• CASE_INSENSITIVE
• MULTILINE (^ y $ aplican por línea)
• DOTALL (el punto . coincide con salto de línea)
• UNICODE_CASE, UNICODE_CHARACTER_CLASS
- Métodos esenciales:
• matcher(CharSequence) → devuelve un Matcher
• matches(regex, input) → comprobación puntual (no eficiente repetidamente)
• split(input) → divide usando la regex
• pattern() → devuelve la regex original
- Constructos básicos de regex (según docs de Pattern):
• Literales: a, b, c
• Clases: [abc], [^abc], [a-zA-Z], [a-d[m-p]]
• Predefinidas: \d dígito, \w alfanum., \s espacio (doble barra en String Java: "\d")
• Cuantificadores: x*, x+, x?, x{n}, x{n,}, x{n,m}
• Grupos: (…), grupos de captura numerados desde 1. Grupo 0 = toda la coincidencia.
• Alternancia: a|b|c
• Anchuras: ^ inicio, $ fin, ■ límite de palabra, \A y \Z anclajes absolutos.
- Nota esencial: en literales Java hay que escapar la barra invertida → “\d”, “\b”, “\w”.

3. Clase Matcher

- Crear matcher: Matcher m = p.matcher(input);
- Métodos esenciales:
• matches(): la ENTIRE input debe coincidir.
• lookingAt(): coincide desde inicio, no necesita cubrir todo.
• find(): busca siguiente coincidencia parcial → clave para extraer múltiples matches.
• group(): coincidencia actual completa.
• group(i): grupo de captura i.
• start(), end(): índices de la coincidencia/grupo.
• reset() / reset(nuevaSecuencia): reutilizar matcher.
- Patrón de uso típico:

```
Pattern p = Pattern.compile("regex");
Matcher m = p.matcher(texto);
while (m.find()) {
m.group(); m.start(); m.end();
}
```

- Errores típicos:
• Usar matches() cuando realmente se quiere find() → matches exige coincidencia total.
• Acceder a grupos inexistentes → IndexOutOfBounds.
• Regex mal formada → PatternSyntaxException al compilar.

4. Mini■guía práctica para ejercicios

- Si buscas patrones dentro de texto: usa find().
- Si verificas formato completo (ej. validar fecha): usa matches().

- Divide siempre: “¿qué clase de caracteres necesito?” → “¿qué cuantificador?” → “¿necesito grupos?”.
- Usa grupos para extraer partes del texto.
- Usa flags para alterar comportamiento global (como multiline o ignore case).
- Relee ‘Summary of regular-expression constructs’ en la docs de Pattern cuando dudes.