Chuleta Completa de Expresiones Regulares y Logging en Java

Este documento sirve como referencia rápida (chuleta) para el uso de expresiones regulares (Regex) y sistemas de logging en Java. Incluye patrones habituales, ejemplos prácticos y buenas prácticas, pensado especialmente para estudio y exámenes.

1. Expresiones Regulares en Java (java.util.regex)

En Java, las expresiones regulares se gestionan principalmente con las clases Pattern y Matcher del paquete java.util.regex.
Ejemplo básico:

```
Pattern p = Pattern.compile("abc"); Matcher m = p.matcher("123abc456");
System.out.println(m.find()); // true
```

1.1 Metacaracteres básicos

- **.** cualquier carácter
- **\d** dígito [0-9]
- **\D** no dígito
- **\w** carácter de palabra [a-zA-Z0-9_]
- **\W** no carácter de palabra
- **\s** espacio en blanco
- **\S** no espacio

1.2 Cuantificadores

- ***** 0 o más
- **+** 1 o más
- **?** 0 o 1
- **{n}** exactamente n veces
- **{n,}** al menos n veces
- **{n,m}** entre n y m veces

1.3 Anclas

- **^** inicio de línea
- **$** fin de línea
- **\b** límite de palabra

1.4 Comprobaciones típicas (muy usadas en exámenes)

Email
Regex: ^[\w._%+-]+@[\w.-]+\.[a-zA-Z]{2,}$

```
Java: email.matches("^[\\w._%+-]+@[\\w.-]+\\.[a-zA-Z]{2,}$")
```

DNI español
Regex: ^[0-9]{8}[A-Z]$

```
Java: dni.matches("^[0-9]{8}[A-Z]$")
```

Teléfono (9 dígitos)
Regex: ^[0-9]{9}$
```
Java: tel.matches("^[0-9]{9}$")
```

Código postal España
Regex: ^[0-9]{5}$
```
Java: cp.matches("^[0-9]{5}$")
```

Contraseña fuerte
Regex: ^(?=.*[a-z])(?=.*[A-Z])(?=.*\d).{8,}$
```
Java: pass.matches("^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d).{8,}$")
```

1.5 Métodos importantes de Matcher

- **matches()**: comprueba toda la cadena
- **find()**: busca coincidencias parciales
- **group()**: devuelve el grupo capturado
- **start() / end()**: posiciones de la coincidencia

2. Logging en Java

El logging permite registrar información de ejecución de un programa: errores, avisos y trazas. Es esencial en aplicaciones reales.

2.1 java.util.logging (JUL)

Ejemplo básico:

```
import java.util.logging.Logger; Logger logger =
Logger.getLogger(MiClase.class.getName()); logger.info("Mensaje informativo");
logger.warning("Mensaje de aviso"); logger.severe("Error grave");
```

Niveles de logging:

- SEVERE
- WARNING
- INFO
- CONFIG
- FINE / FINER / FINEST

2.2 Log4j / SLF4J (muy usado en proyectos)

SLF4J es una fachada de logging. Log4j o Logback son implementaciones reales.

Ejemplo con SLF4J:

```
import org.slf4j.Logger; import org.slf4j.LoggerFactory; private static final Logger
log = LoggerFactory.getLogger(MiClase.class); log.debug("Depuración");
log.info("Información"); log.warn("Advertencia"); log.error("Error");
```

2.3 Buenas prácticas

- No usar System.out.println en aplicaciones grandes
- Usar niveles adecuados (info, warn, error)
- No concatenar strings: usar parámetros
- Configurar salida a fichero

```
Ejemplo correcto: log.info("Usuario {} ha iniciado sesión", usuario);
```

3. Consejos típicos de examen

- matches() valida toda la cadena, find() no
- En Java, \ debe escaparse como \\
- Regex + logging = validaciones bien registradas