Java Snippets (OWASP Top 10)

1. SQL Injection (A03: Injection)

```
String user = request.getParameter("username");
String pass = request.getParameter("password");
String query = "SELECT * FROM users WHERE username='" + user + "' AND
password='" + pass + "'";
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(query);
```

🔴 **Vulnerability:** SQL Injection – unsanitized concatenation in SQL query.
✅ **Fix:** Use `PreparedStatement`.

---

2. XML External Entity (XXE) (A05: Security Misconfiguration / Injection)

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
DocumentBuilder db = dbf.newDocumentBuilder();
Document doc = db.parse(new File("input.xml"));
```

🔴 **Vulnerability:** XXE attack enabled by default.
✅ **Fix:** Disable DTD/Entity resolution.

---

3. Broken Authentication via Cookie (A07: Identification & Authentication Failures)

```
if(request.getParameter("remember").equals("true")) {
    Cookie c = new Cookie("auth", "admin:true");
    response.addCookie(c);
}
```

🔴 **Vulnerability:** Authentication state stored in modifiable cookie.
✅ **Fix:** Use signed/secure cookies or session management.

---

4. Cross-Site Request Forgery (CSRF) (A05: Broken Access Control)

```
doPost(HttpServletRequest req, HttpServletResponse res) {
    String action = req.getParameter("transfer");
    if(action != null) {
        bank.transfer(req.getParameter("amount"));
    }
}
```

🔴 **Vulnerability:** No CSRF protection.
✅ **Fix:** Implement CSRF tokens in forms.

---

5. Insecure Deserialization (A08: Software & Data Integrity Failures)

```
ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("data.ser"));
Object obj = ois.readObject();
```

🔴 **Vulnerability:** Arbitrary code execution if attacker supplies serialized object.
✅ **Fix:** Use validation + whitelisting of classes.

---

6. Using Vulnerable Component – Log4j (A06: Vulnerable Components)

```
// Using Log4j < 2.15 (CVE-2021-44228)
logger.info("User input: " + request.getParameter("input"));
```

🔴 **Vulnerability:** Log4Shell exploit via user-controlled input in vulnerable Log4j.
✅ **Fix:** Upgrade Log4j, sanitize inputs.

---

7. Security Misconfiguration – Stack Trace Disclosure (A05: Security Misconfiguration)

```
<%@ page import="java.sql.*" %>
<%@ page isErrorPage="true" %>
```

```
<%= exception %>
```

🔴 **Vulnerability:** Full stack trace exposed to users → info leakage.
✅ **Fix:** Show custom error page, disable exception exposure.

---

8. Cross-Site Scripting (XSS – JSP) (A03: Injection)

```
<%= request.getParameter("msg") %>
```

🔴 **Vulnerability:** Reflected XSS → input not sanitized/encoded.
✅ **Fix:** Use `<c:out>` or proper output encoding.

---

9. XSS in Script Context (A03: Injection)

```
<script>
  var msg = "<%= request.getParameter("q") %>";
  document.write(msg);
</script>
```

🔴 **Vulnerability:** Injected JavaScript from unsanitized input.
✅ **Fix:** Encode before insertion, use CSP.

---

10. Insufficient Logging/Monitoring (A09: Security Logging & Monitoring Failures)

```
try {
    login(user, pass);
} catch(Exception e) {
    // ignore
}
```

🔴 **Vulnerability:** Silent failure → attacks unnoticed.
✅ **Fix:** Log authentication failures.

📌 Summary Table – Java Specific Vulnerabilities

#	Vulnerability	OWASP Top 10 (2021)
1	SQL Injection via string concatenation	A03: Injection
2	XML External Entity (XXE)	A05: Security Misconfiguration / Injection
3	Weak Cookie-Based Authentication	A07: Identification & Authentication Failures
4	No CSRF Protection	A05: Security Misconfiguration
5	Insecure Deserialization	A08: Software & Data Integrity Failures
6	Log4j Exploit (Log4Shell)	A06: Vulnerable & Outdated Components
7	Stack Trace Disclosure	A05: Security Misconfiguration
8	Reflected XSS in JSP	A03: Injection
9	Script-Based XSS in JSP	A03: Injection
10	No Logging of Failures	A09: Security Logging & Monitoring Failures