Java String DSA Cheat Sheet

Basics

```
String s = "hello";
s.length() // 5
s.charAt(1) // e
s.substring(2) // "llo"
s.substring(1,4) // "ell"
s.equals("hello") // true
s.equalsIgnoreCase("HELLO") // true
"".isEmpty() // true
```

Case Conversion

```
s.toUpperCase(); // "HELLO"
s.toLowerCase(); // "hello"
```

Concatenation

```
String a = "Hello";
String b = "World";
String c = a + " " + b; // "Hello World"
```

Trimming

```
" hello ".trim(); // "hello"
```

Searching

```
String s = "abcabc";
s.indexOf("b"); // 1
s.lastIndexOf("b"); // 4
s.contains("ab"); // true
s.startsWith("ab"); // true
s.endsWith("bc"); // true
```

Replace & Remove

```
s.replace('a', 'x'); // "xbcxbc"
s.replaceAll("a", "z"); // "zbczbc"
s.replaceFirst("a", "z"); // "zbcabc"
```

Splitting

```
String[] parts = "a,b,c".split(",");
for (String part : parts) System.out.println(part);
```

Reverse a String

```
new StringBuilder(s).reverse().toString(); // "cbacba"
```

Frequency of Characters

```
int[] freq = new int[26];
for (char c : s.toCharArray()) freq[c - 'a']++;
```

Palindrome Check

```
boolean isPalindrome(String s) {
int l = 0, r = s.length() - 1;
while (l < r) if (s.charAt(l++) != s.charAt(r--)) return false;
return true;
}
```

Permutations (Backtracking)

```
void permute(String str, String ans) {
if (str.length() == 0) {
System.out.println(ans); return;
}
for (int i = 0; i < str.length(); i++) {
permute(str.substring(0, i) + str.substring(i + 1), ans + str.charAt(i));
}
}
```

Longest Common Substring (DP)

```
int longestCommonSubstr(String a, String b) {
int[][] dp = new int[a.length()+1][b.length()+1];
int max = 0;
for (int i = 1; i <= a.length(); i++) {
for (int j = 1; j <= b.length(); j++) {
if (a.charAt(i-1) == b.charAt(j-1)) {
dp[i][j] = dp[i-1][j-1] + 1;
max = Math.max(max, dp[i][j]);
}
}
}
return max;
}
```

StringBuilder vs String

```
StringBuilder sb = new StringBuilder("abc");
sb.append("def"); // "abcdef"
```

Interning

```
String a = new String("java");
String b = "java";
System.out.println(a == b); // false
System.out.println(a.intern() == b); // true
```

Pattern Matching

```
Pattern p = Pattern.compile("[a-z]+");
Matcher m = p.matcher("abc123");
System.out.println(m.find()); // true
```

Longest Palindromic Substring (DP)

```
String longestPalindrome(String s) {
```

```
int n = s.length();
boolean[][] dp = new boolean[n][n];
String ans = "";
for (int l = n - 1; l >= 0; l--) {
for (int r = l; r < n; r++) {
if (s.charAt(l) == s.charAt(r) && (r - l <= 2 || dp[l+1][r-1])) {
dp[l][r] = true;
if (r - l + 1 > ans.length()) ans = s.substring(l, r+1);
}
}
}
return ans;
}
```

Tips

```
- Prefer StringBuilder for loops or appending.
- Always use .equals() for content comparison.
- Learn regex for strong string manipulation.
```