Cheat Sheet: Java Core & Estructuras

1. Math: Operaciones y Aleatoriedad

Clase estática para cálculos. No requiere new.

Generación de Números Aleatorios (`Math.random()`)

Devuelve un `double` entre `0.0` (incluido) y `1.0` (excluido). Para rangos específicos, se usa la fórmula: `x = (int)(Math.random() * (MAX - MIN + 1)) + MIN;`

Rango Deseado	Explicación	Fórmula Java (Ejemplo en línea)
0 a 9	Multiplicar por 10 (0 a 9.99) y castear.	`int n = (int)(Math.random() * 10);`
1 a 10	Rango de 0-9 y sumar 1.	`int n = (int)(Math.random() * 10) + 1;`
1 a 100	Rango 0-99 y sumar 1.	`int n = (int)(Math.random() * 100) + 1;`
5 a 10	`(10-5+1)` = 6 valores posibles. Base 5.	`int n = (int)(Math.random() * 6) + 5;`
-10 a 10	`(10 - (-10) + 1)` = **21 valores. Base -10.**	`int n = (int)(Math.random() * 21) - 10;`

Métodos Matemáticos Comunes

Método	Descripción	Ejemplo	Resultado
`pow(base, exp)`	Potencia (retorna double).	`Math.pow(2, 3)`	**8.0**
`sqrt(n)`	Raíz cuadrada.	`Math.sqrt(25)`	**5.0**
`abs(n)`	Valor absoluto.	`Math.abs(-5)`	**5**
`round(n)`	Redondeo estándar (al más cercano).	`Math.round(5.6)`	**6**
`floor(n)`	Redondeo al suelo (hacia abajo).	`Math.floor(5.9)`	**5.0**
`ceil(n)`	Redondeo al techo (hacia arriba).	`Math.ceil(5.1)`	**6.0**
`max(a, b)` / `min`	**Máximo o mínimo de dos números.**	`Math.max(10, 20)`	**20**

2. String y StringBuilder

Gestión de texto. **String** es inmutable (crea nuevos objetos). **StringBuilder** es mutable (modifica el mismo objeto).

Clase `String`

Método	Descripción	Ejemplo (`s = "Hola"`)	Resultado
`length()`	Longitud de la cadena.	`"Hola".length()`	**`4`**
`charAt(i)`	Carácter en índice `i`.	`"Hola".charAt(0)`	**`'H'`**
`equals(str)`	**Importante:** Compara contenido.	`"Hola".equals("hola")`	**`false`**
`equalsIgnoreCase`	Compara ignorando mayús/minús.	`"Hola".equalsIgnoreCase("hola")`	**`true`**
`substring(ini, fin)`	Extrae rango. **`fin` excluido**.	`"Java".substring(0, 2)`	**`"Ja"`**
`indexOf(str)`	Posición de 1ª aparición (-1 si no está).	`"Banana".indexOf("a")`	**`1`**
`replace(old, new)`	Reemplaza caracteres/texto.	`"Lala".replace('a', 'o')`	**`"Lolo"`**
`trim()`	Elimina espacios extremos.	`" Hi ".trim()`	**`"Hi"`**
`split(regex)`	Divide en array por separador.	`"A,B".split(",")`	**`["A", "B"]`**
`String.valueOf(x)`	**Convierte primitivo a String.**	**`String.valueOf(123)`**	**`"123"`**

Clase `StringBuilder`

Declaración: `StringBuilder sb = new StringBuilder("Texto");`

Método	Descripción	Ejemplo (`sb = "Hola"`)	Resultado (sb)
`append(txt)`	Añade al final.	`sb.append(" Mundo")`	**`"Hola Mundo"`**
`insert(idx, txt)`	Inserta en posición `idx`.	`sb.insert(0, "¡")`	**`"¡Hola"`**
`delete(ini, fin)`	Borra rango (`fin` excluido).	`sb.delete(1, 3)`	**`"Ha"`**
`reverse()`	Invierte el texto.	`sb.reverse()`	**`"aloH"`**
`toString()`	**Convierte a String final.**	**`String s = sb.toString()`**	**`"aloH"`**

3. Arrays y java.util.Arrays

Colecciones de tamaño fijo. Índices empiezan en `0`.

Conceptos Básicos

Concepto	Sintaxis / Descripción	Ejemplo
Declaración	`Tipo[] nombre = new Tipo[tamaño];`	**`int[] n = new int[5];`**
Con Valores	Inicialización directa con llaves.	**`String[] s = {"A", "B"};`**
Longitud	Propiedad `.length` (sin paréntesis).	**`int x = n.length;`**
Recorrido	**Bucle `for-each` (lectura).**	**`for(String i : s) { ... }`**

Métodos Avanzados de `java.util.Arrays`

Se asume `import java.util.Arrays;` y array `arr = {10, 20, 30, 40, 50}`.

Método	Detalle / Funcionamiento	Ejemplo de Código	Salida
`toString(arr)`	Representación legible del array.	`Arrays.toString(arr)`	**`"[10, 20, 30, ...]"`**
`sort(arr)`	Ordena el array original (Quick/Merge).	`int[] x={2,1}; Arrays.sort(x);`	**`x es [1, 2]`**
`equals(a, b)`	Compara contenido (longitud y elementos).	`Arrays.equals(arr, arr2)`	**`true / false`**
`fill(arr, val)`	**Rellena todo el array con `val`.**	**`Arrays.fill(arr, 0)`**	**`[0, 0, 0, 0, 0]`**

Profundización: `copyOf` y `binarySearch`

Método	Caso	Explicación Técnica	Ejemplo	Salida
`copyOf(origen, len)` ***(Crea nuevo array)***	**Exacta**	Longitud igual a original.	`Arrays.copyOf(arr, 5)`	**`[10, 20, 30, 40, 50]`**
	Truncar	Longitud menor (corta datos).	`Arrays.copyOf(arr, 2)`	**`[10, 20]`**
	Padding	Longitud mayor (rellena 0/null).	`Arrays.copyOf(arr, 6)`	**`[10, ..., 50, 0]`**
`binarySearch(arr, key)` ***(Requiere array ordenado)***	**Éxito**	Devuelve índice del elemento.	`Arrays.binarySearch(arr, 30)`	**`2`**
	Fallo	Retorna `-(punto_inserción) - 1`.	`Arrays.binarySearch(arr, 35)`	**`-4`**

		Si key=35 (iría en índice 3): `-3-1`	
	Fallo	**Menor que todos (iría en 0): `-0-1`**	**`Arrays.binarySearch(arr, 5)` `-1`**

4. Enum (Enumeraciones Detalladas)

Tipo seguro para constantes. Son objetos con métodos y pueden tener atributos.

Uso Básico y Métodos

Declaración: `public enum Nivel { BAJO, MEDIO, ALTO }`

Método / Uso	Descripción	Ejemplo	Salida / Acción
Asignación	Tipo estricto.	`Nivel n = Nivel.MEDIO;`	**Variable n definida.**
`name()`	Nombre exacto como String.	`n.name()`	**`"MEDIO"`**
`ordinal()`	Posición (índice base 0).	`n.ordinal()`	**`1`**
`values()`	Array estático con todas las constantes.	`Nivel[] todos = Nivel.values();`	**`[BAJO, MEDIO, ALTO]`**
Comparación	**Seguro usar `==`.**	**`if (n == Nivel.ALTO)`**	**`false`**

Enums Avanzados (Con atributos)

Los Enums funcionan como clases. Pueden tener constructor privado y campos.

```
public enum Planeta {

    TIERRA(1.0), MARTE(0.38); // Invoca al constructor

    private double gravedad; // Atributo
    private Planeta(double g) { this.gravedad = g; } // Constructor (privado)
    public double getGravedad() { return gravedad; } // Getter
}
// Uso: double g = Planeta.MARTE.getGravedad(); // 0.38
```

Uso en Estructuras de Control

Estructura	Sintaxis Compacta
Iteración	**`for (Nivel n : Nivel.values()) { System.out.println(n); }`**
Switch	**`switch(nivel) { case BAJO: ...; break; case ALTO: ...; break; }`** ***(Nota: En los `case` no se pone `Nivel.BAJO`, solo `BAJO`)***