**Introducción a Java**

* Instalar Eclipse en clase:

**curl https://fquintana.ddns.net/fquintana-eclipse | bash**

* POO: Clases y objetos
* Lenguaje estructurado
  + Estructuras de condición (if, else, switch)
  + Estructuras repetitivas (for, while, do while)
* POO y estructuras
* La clase Principal para ejecutar los programas contendrá el método **main**
* **Comentarios en Java:**
  + Una línea: // Soy un comentario
  + Más de una línea: /\* esto

es

un

comentario \*/

* Tipos de atributos:
  + **int** numero = 7;
  + **double** decimal = 9.5;
  + **char** caracter = '@';
  + **String** palabra = "achus";
  + **boolean** soyHumano = true;
* **Imprimir valores por pantalla**
  + System.out.print(decimal); 🡪 No hace salto de línea
  + System.out.println(palabra); 🡪 Sí hace salto de línea
  + Para imprimir por pantalla un texto seguido del valor de una variable escribimos el texto entre comillas dobles seguido de + nombre variable
    - System.out.println("El decimal es:"+decimal);
  + Para imprimir por pantalla el texto con un intro o con una tabulación, utilizamos \n y \t, respectivamente
    - System.out.println("El decimal es:\n"+decimal);
    - System.out.println("Oscar ha hecho \t"+palabra);
  + Para no tener que escribir la línea entera, TRUCO! Escribirmos la palabra **sysout** (y a continuación pulsamos CONTROL BARRAESPACIADORA)
* **Estructura condicional IF:**
  + 1. El IF solitario:

if (numero > 0) {

System.out.println("El numero es mayor que 0");

}

* + 2. El IF-ELSE:

if (numero > 0) {

System.out.println("El numero es mayor que 0");

}

else {

System.out.println("El numero es menor o igual a 0");

}

* + 3. El IF-ELSE encadenado:

if (numero > 0) {

System.out.println("El numero es mayor que 0");

}

else {

if (numero == 0) {

System.out.println("El numero es igual a 0");

}

else {

System.out.println("El numero es menor que 0");

}

}

* + Consideraciones del IF/ELSE:

Comparaciones matemáticas: > (mayor), < (menor), >= (mayor o igual), <=

(menor o igual), == (igualdad), != (distinto)

Comparaciones lógicas: && (AND), OR (||)

* + - && 🡪 V, V = V V, F = F F, V = F F, F = F
    - || 🡪 V, V = V V, F = V F, V = V F, F = F

if (numero>5 || numero2<=13) {

System.out.println("Estoy dentro del if");

}

else {

System.out.println("Estoy dentro del else");

}

* **Estructura condicional SWITCH:**

switch(numero) {

case 0: System.out.println("Soy cero");

break;

case 1: System.out.println("Soy uno");

break;

case 2: System.out.println("Soy dos");

break;

default: System.out.println("Soy otro");

break;

}

* **Estructura repetitiva FOR:**

La estructura for tiene 3 partes: la primera para definir el índice, la segunda para comparar hasta su final y la tercera para el incremento o decremento.

Los métodos charAt(i) y length() ayudan a conocer el carácter de la posición de la palabra y la longitud de dicha palabra.

Ejemplo con incremento:

for (int i=0;i<alumno.length();i++) {

// con charAt(i) obtengo la letra en cada iteración

char letra = alumno.charAt(i);

System.out.println(letra);

}

Ejemplo con decremento:

for (int i=alumno.length()-1;i>=0;i--) {

// con charAt(i) obtengo la letra en cada iteración

char letra = alumno.charAt(i);

System.out.println(letra);

}

* **Estructura repetitiva WHILE:**

Con un while se pretende crear un bucle a partir de una condición. En cada iteración, la condición debe variar para encontrar la solución requerida.

int contador = 1;

while (contador <=15) {

System.out.println(contador);

contador++;

}

Sin variar la condición dentro del while, es muy probable que se obtenga un bucle infinito. En el ejemplo, si no existiera contador++, siempre se estaría ejecutando.

* **Estructura repetitiva DO-WHILE:**

Se trata de un bucle parecido al while pero que siempre ejecutará, al menos, una vez su contenido. Posteriormente, comprobará la condición.

int contador = 1;

do {

// haz lo que sea

System.out.println("Estoy dormido");

contador++;

}while (contador<=4);

* **Arrays (estáticos):**

Una lista de elementos del mismo tipo. Para crear un array, tenemos 2 opciones:

1. Lo creamos vacío añadiendo la capacidad:

Tipo [ ] nombre = new Tipo [capacidad];

Por ejemplo: String [] contactos = new String[6];

1. Lo creamos con los elementos incluidos

Tipo [ ] nombre = {añadir los elementos separados por comas};

int [ ] edades = {19, 20, 26, 21, 18, 22};

* Para obtener un valor de un array, la sentencia sería: edades[posicion];

edades[4] 🡪 Obtendrá el valor 18

* Para recorrer un array, generalmente se usa for: desde la posición 0 (inicial) a la posición longitud total (final):

for(int i=0; i<edades.length; i++) {

System.out.println(edades[i]);

}

* **ArrayList (dinámicos):**

Una lista de elementos del mismo tipo sin necesidad de darle capacidad. No va a ser necesario establecer un límite, puesto que al ser un array dinámico se adapta al contenido establecido. Lo creamos de la siguiente manera:

ArrayList<Tipo> nombre = new ArrayList<Tipo>();

Por ejemplo: ArrayList<String> lista = new ArrayList<String>();

El método add permite añadir un elemento al ArrayList

El método remove seguido de la posición permite eliminar el elemento de dicha posición.

El método size permite conocer la longitud del ArrayList (equivalencia a length de los arrays estáticos).

* **CLASES Y OBJETOS**
* Una clase representa una estructura de algo que pretendemos implementar. Va a estar formada por atributos y métodos, fundamentalmente.