**RESUME UNIDAD 3 4 5**

Joaquín García Téllez 1ºDAW B

**UNIDAD 3**

Un bucle es un tipo de estructura que contiene un bloque de instrucciones que se ejecutan de forma repetida.

Los bucles existentes son: while, do-while y for.

3.1. Bucles controlados por condición

El control de interacciones se lleva a cabo mediante una condición. Donde sí la condición es cierta el bucle realiza las interacciones.

3.1.1. While

Al igual que el if, la instrucción while depende de una condición. Con la única diferencia que las instrucciones de adentro se ejecutaran de forma repetida hasta que la condición deje de cumplirse.

While (condición) {}

3.1.2. do-while

Disponemos de un segundo bucle controlado por una condición: do-while. El cual se diferencia en que primero se ejecuta el bloque de instrucciones y después se evalúa la condición para decidir si se realiza de nuevo.

do {

instrucciones

}while (condición);

3.2. Bucles controlados por contador: for

El bucle for permite controlar un número de interacciones mediante una variable (contador)

for (inicialización; condición; incremento){}

-Inicialización: para indicar una variable, que va a controlar el bucle.

-Condición: la expresión por la que se va a regir el bucle para que se cumpla sí o no.

-Incremento: construcciones para modificar el controlador de la condición.

3.4. Bucles anidados

Es muy frecuente anidar bucles. Esto nos quiere decir es que dentro de un bucle puede haber otro.

3.4.1. Bucles Independientes

Un bucle anidado no tiene por qué depender de otro, a estos se les llama bucles independientes.

3.4.2. Bucles Dependientes

Un bucle anidado puede depender de otro, a estos se les llama bucles dependientes.

**UNIDAD 4**

Conforme aumenta la extensión y la complejidad, esto ocasiona copiar y por ello generar problemas de: duplicidad de código y dificultad de mantenimiento.

4.1. Conceptos básicos

La solución mas optima es etiquetar el código, más conocido como crear un método.

public static void main(String[] args){}

static void nombre(){}

Con esto lo que conseguimos evitar la duplicación del código y facilitar el mantenimiento del mismo.

4.2. Ámbito de las variables

En el cuerpo de una función podemos declarar variables locales. Las cuales no se puedes utilizar fuera de ellas.

4.3. Paso de información a una función

En ocasiones, una función necesita información externa para poder llevar a cabo su tarea.

4.3.1. Valores en la llamada

En la llamada a una función se puede pasar valores.

4.3.2. Parámetros de entrada

Una función puede definirse para recibir tantos datos como necesite. Cada dato utilizado será asignado a un parámetro de entrada.

Los parámetros toman el valor como una copia de la expresión o variable utilizada en la llamada. Los cambios de esta variable realizados dentro de esta llamada no se verán afectados fuera de ella.

4.4. Valor devuelto por una función

Hemos visto que es posible pasar información a través de los parámetros de entrada. También es posible hacerlo al contrario. Para realizar esta función es necesario que entre ambas funciones haya concordancia.

4.5. Sobrecarga de funciones

Java permite que dos o más funciones compartan el mismo identificador en un mismo programa. Esto se le llama sobrecarga de funciones. La forma de distinguirlos es mediante los parámetros utilizados.

Es muy común encontrar métodos sobrecargados, pero con distintos paramentos.

**UNIDAD 5**

En los programas escritos hasta ahora hemos utilizado los tipos primitivos.

5.1. Tipo primitivo char

De forma general un carácter simple se describe como un char.

5.1.1. Unicode

Unicode identifica caracteres mediante un número único, llamado code point.

A la hora de seleccionar un carácter es posible utilizar Unicode o el propio carácter.

5.1.2. Secuencias de escape

Un carácter precedido de una barra invertida se conoce como secuencia de escape.

5.1.3. Conversión char<->int

Cada code point no es más que un número entero, representando un decimal. Es posible asignar una variable tipo int a un tipo char

5.1.4. Aritmética de caracteres

La relación que existe entre un carácter y su representante Unicode, permite realizar operaciones asimétricas.

5.2. Clase Character

El tipo char es a todas luces insuficiente para realizar operaciones con caracteres. La clase Character amplía la funcionalidad y la simplicidad del trabajo.

5.2.1. Clasificación de caracteres

Un carácter puede clasificarse en: digito, letra, blancos u otros.

La clase Character se puede usar para devolver un booleano.

5.2.2. Conversión

Los métodos que realizan conversiones son aquellos que devuelven transformado el valor que se les pasa como parámetros. También existe los que realizan operaciones inversas. ToLowerCase/toUpperCase.

5.3. Clase String

Las cadenas, conjuntos secuenciales de carácter, se manipulan mediante la clase String, que funciona de forma dual.

Almacena una cadena de caracteres de caracteres.

5.3.1. Inicialización de cadenas

De forma análoga a como lo hacemos con la clase Scanner, podemos utilizar new para crear y asignar un valor a un String.

5.3.2. Comparación

Los operadores de comparación disponles para números y caracteres igual.

Si queremos comparar de forma igualitaria un tipo String utilizamos:

-Iguales= cad1.equals (cad2);

-Alfabética: otra forma de comparar es de forma alfabética.

5.3.3. Concentración

El operador + sirve para unir o concatenar dos cadenas.

5.3.4. Obtención de caracteres

Todos los caracteres que forman una cadena puede se identidad mediante la posición que ocupa, al igual que los elementos de una tabla. Char charAt (posición).

5.3.5. Longitud de una cadena

Para poder ver o utilizar la longitud de una cadena utilizamos: cad1.length ().

5.3.6. Búsqueda

Dentro de una cadena, entre los caracteres que la forman, e posible buscar un carácter o una subcadena.

IndexOf (): sirve para buscar ocurrencias desde el principio.

iastIndexOf: sirve para buscar ocurrencias desde el final.

5.3.7. Comprobaciones

Es posible de realizar ciertas comprobaciones con una cadena de caracteres. Por regla general, los métodos que realizan estas comprobaciones devuelven booleanos.

-Una cadena vacía, es aquella que no está formada por ningún carácter y se representa””.

-Para comprobar si una cadena contiene a otra utilizaríamos .contains.

-Para un prefijo o sufijo: .starsWith.

5.3.8. Conversión

Una cadena puede transformase sustituyendo todas las letras que la componen utilizando: toLowerCase/toUpperCase.

El método replace permite sustituir las ocurrencias.

5.3.9. Sedación en partes

Una cadena se puede descomponer en partes si utilizamos un separador: Split (String separador)

5.4. Cadenas y tablas de caracteres

Existe una innegable relación entre cadenas, clase String y las tablas de caracteres.

La forma de crear una tabla de caracteres es: toCharArray ().