Tema 2
Programación estructurada

Prof. José de la Torre López
(adaptación Antonio Hernández)

1º Desarrollo de Aplicaciones Multiplataforma
Módulo formativo - Programación

La caja negra

Reglas de la programación

1. Entender bien lo que se pide
2. Comprender cómo comienza el programa
3. Contemplar todos los tipos que de entrada que pueda haber en mi caja negra (el usuario es tonto)
4. ¿Qué pasos tengo que seguir para que con esa entrada le de a mi usuario su resultado deseado?
5. ¿Cómo y cuándo le voy a dar a mi usuario su resultado?

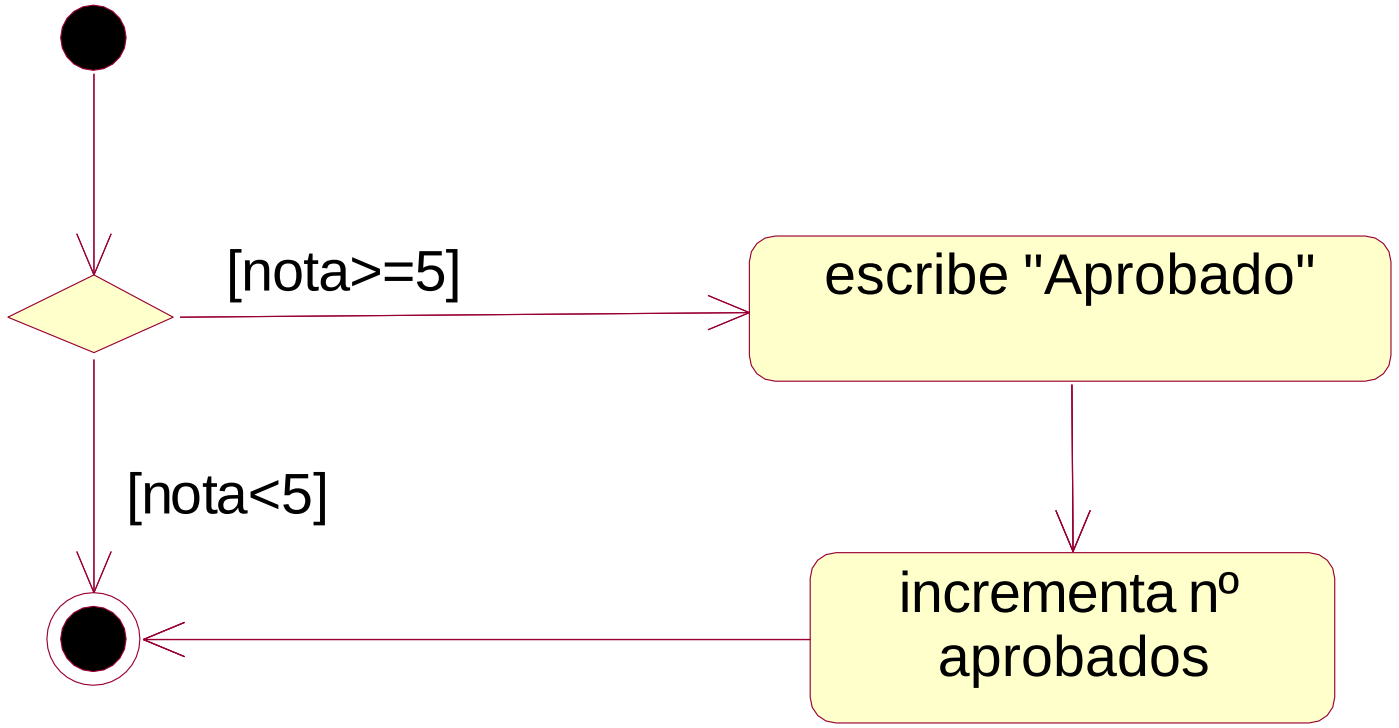
¿Qué sabemos hasta ahora?

- Sintaxis básica de Java
- Crear programas que hagan un solo propósito
- Los programas van línea a línea tal cual han sido programados

¿Qué vamos a ver?

- Condicionales: bifurcaciones lógicas
- Bucles: repetir trozos de código

Condicional simple - if



Condicional simple - if

```
if (expresiónLógica) {
    … código …
}
```

Condicional simple - if

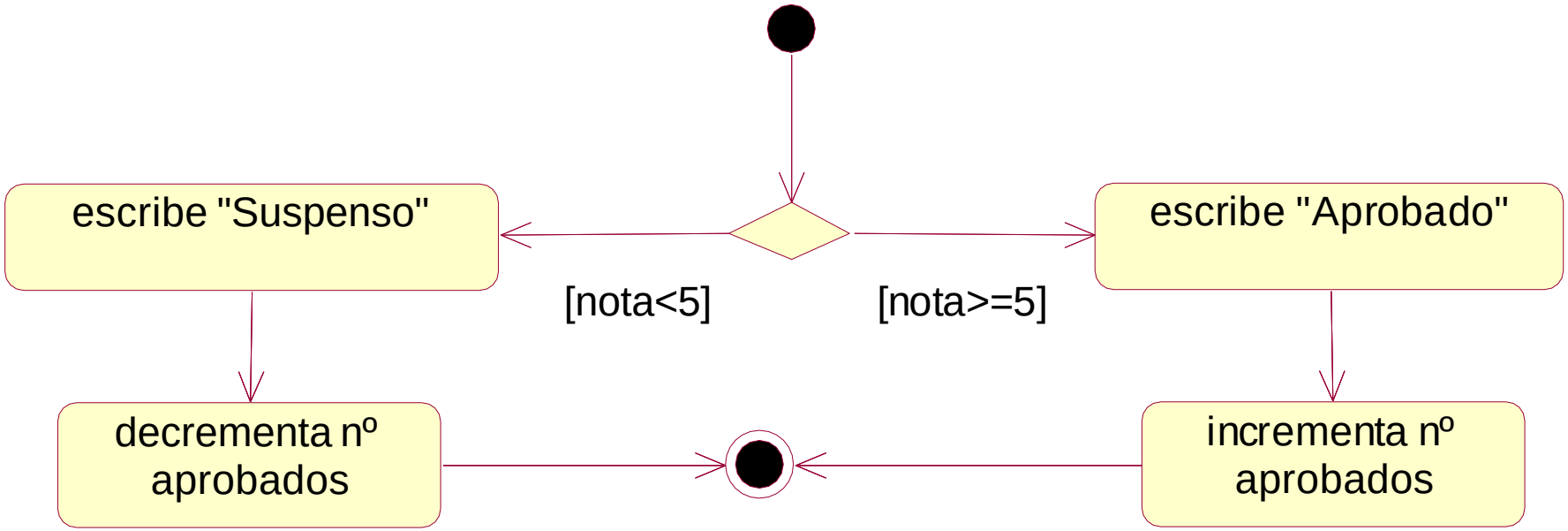
```
if (expresiónLógica)
    sentencia;

// No es recomendable
```

Condicional simple - if

```
if  (nota >= 5) {
    System.out.println(“Aprobado”);
    aprobados++;
}
```

Condicional simple - if



Condicional compuesta- if

```
if (expresiónLógica) {
    … código …
}
else {
    … código …
}
```

Condicional compuesta- if

```
if  (nota >= 5) {
    System.out.println(“Aprobado”);
    aprobados++;
}
else {
    System.out.println(“Suspenso”);
    suspensos++;
}
```

Condicional anidado

```
if (expresiónLógica) {
    … código …
}
else {
    if (expresiónLógica) {
    … código …
    }
}
```

Condicional anidado - Menú

```
if(x == 1) {
	//Código de la opción 1
}
else {
	if (x == 2) {
		//Código de la opción 2
	}
	else {
		if (x == 3) {
			//Código de la opción 3
		}
		else {
			//Código para cualquier otra cosa
		}
	}
}
```

Condicional anidado mejorado

```
if(x == 1) {
      //Código de la opción 1
}
else if (x == 2) {
      //Código de la opción 2
}
else if (x == 3) {
      //Código de la opción 3
} else {
      //Código para cualquier otra cosa
}
```

"==" != "="

Permitidme el lujo de recordaros que para comparar si algo es igual a otra cosa se hace con ==

El = es para asignaciones.

Al principio os equivocaréis por esto, pero seguro que lo aprenderéis.

Reglas de la programación

1. Entender bien lo que se pide
2. Comprender cómo comienza el programa
3. Contemplar todos los tipos que de entrada que pueda haber en mi caja negra (el usuario es tonto)
4. ¿Qué pasos tengo que seguir para que con esa entrada le de a mi usuario su resultado deseado?
5. ¿Cómo y cuándo le voy a dar a mi usuario su resultado?

Consejo para empezar*

Dividir el programa en 3 partes fundamentales:

1. Entrada de datos: Se da la bienvenida, se van pidiendo los datos que el usuario tiene que introducir y se validan.
2. Procesado de datos: Se hacen cálculos, decisiones y el propósito del programa.
3. Salida de datos: Se le muestra al usuario lo que ha pedido y se le despide.

* Esto NO es aplicable siempre, son los pasos recomendables para comenzar a programar. Más adelante veremos que el flujo del programa se puede complicar.

Ejercicio en clase

Un profesor requiere un programa que obtenga la nota cualitativa (valor textual de la nota) de un alumno según la nota cuantitativa (valor numérico de la nota) que tenga según la siguiente información:

Muy Deficiente ---------- 0 – 2.99
Insuficiente_____3 – 4.99
Suficiente_____5 – 5.99
Bien_____6 – 7.49
Notable_____7.5 – 8.99
Sobresaliente_____9 – 9.99
Matrícula de Honor ---------- 10

El programa debe mostrar: Nombre y Apellidos del Alumno, Asignatura que se evalúa, Nota Cuantitativa y Nota Cualitativa.

Mejoras al anterior ejercicio

1. ¿Y si introduce en la nota: “pollo frito”?
2. ¿Y si el nombre es obligatorio y le da al intro?
3. ¿Y si introduce un número muy muy grande?¿Cuál es el límite?

En este tema aprenderemos a controlar estos factores.

¿Lo recordáis?

```
if(x == 1) {
      //Código de la opción 1
}
else if (x == 2) {
      //Código de la opción 2
}
else if (x == 3) {
      //Código de la opción 3
} else {
      //Código para cualquier otra cosa
}
```

Condicional complejo – switch case

El switch case es un condicional que imita el comportamiento del bucle anidado internamente.

```
switch (variable){
	case valorVariable1:
		//instrucciones caso 1
		break;
	case valorVariable2:
		//instrucciones caso 2
		break;
	default:
		//instrucciones para cualquier otro caso
		break;
}
```

Son exactamente lo mismo

```
if (x == 1) {
	//Código de la opción 1
}
else if (x == 2) {
	//Código de la opción 2
}
else if (x == 3) {
	//Código de la opción 3
} else {
	//Código para cualquier
	//otra cosa
}
```

```
switch(x){
    case 1:
		//Código de la opción 1
		break;
    case 2:
		//Código de la opción 2
		break;
    case 3:
		//Código de la opción 3
		break;
    default:
		//Código para cualquier
		//otra cosa
		break;
}
```

Ventajas del switch

- En general es más legible y ordenado
- Es más mantenible
- Te da la opción por defecto implícitamente

Ejemplo claro:

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/switch.html

Tipos de variable del switch

- int
- short
- char
- byte
- enum (no os preocupéis ya lo veremos)
- String

Consejos con el switch

- Los case tienen un :
- Los case tienen un break; al final. Si no lo ponéis ejecutará todo el código hasta encontrar un break;

Hemos aprendido...

- If
- Else
- Anidados
- Switch Case

Pero también hemos hablando de que algo es más mantenible...

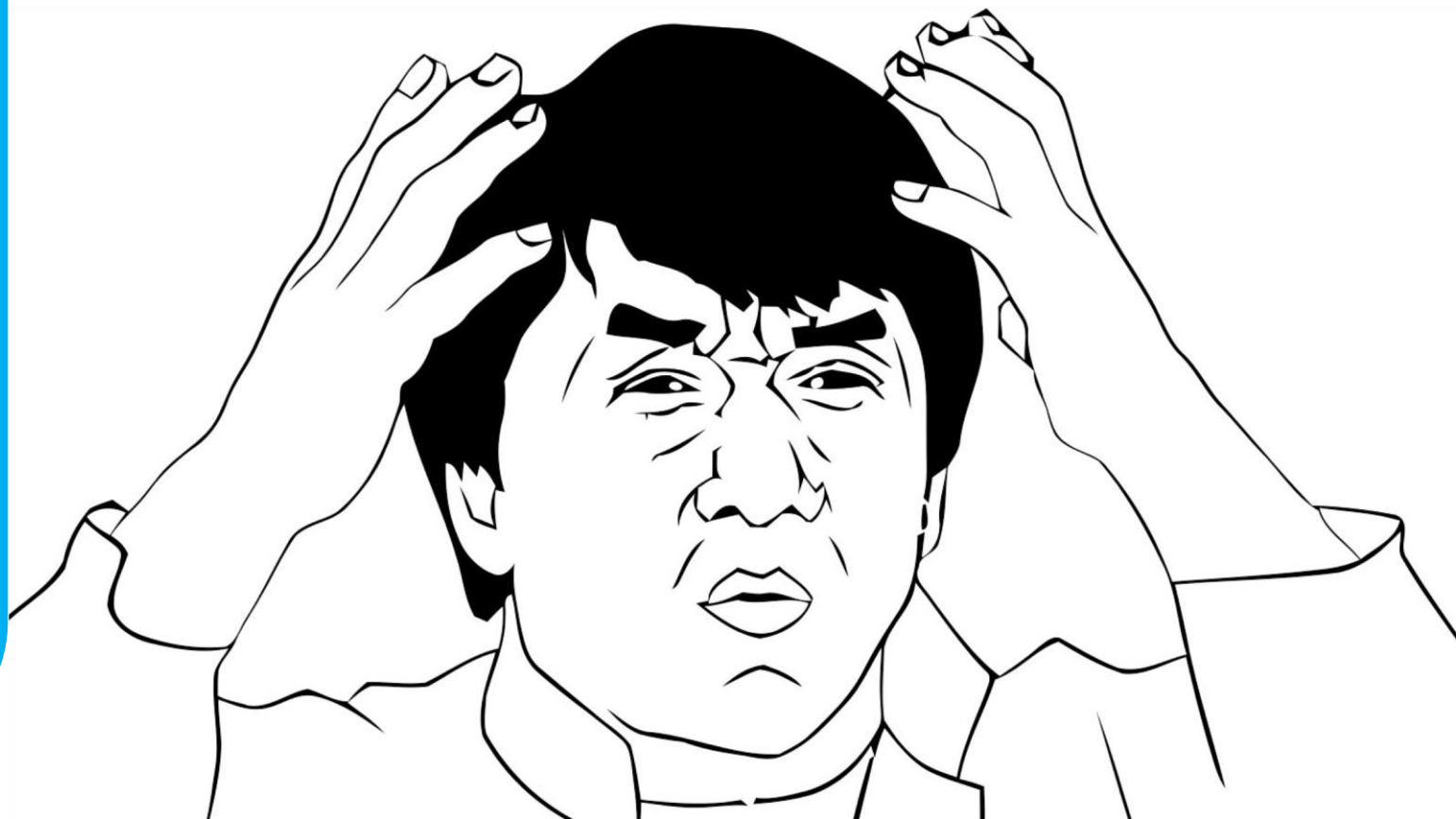
Buenas prácticas y batallitas

Cuando programamos comentamos el código porque el YO futuro no recordará qué narices está pasando cuando tenga arreglar un bug.

Buenas prácticas y batallitas

Preguntaréis:
¿Quién co_o ha escrito esta mi_rda?

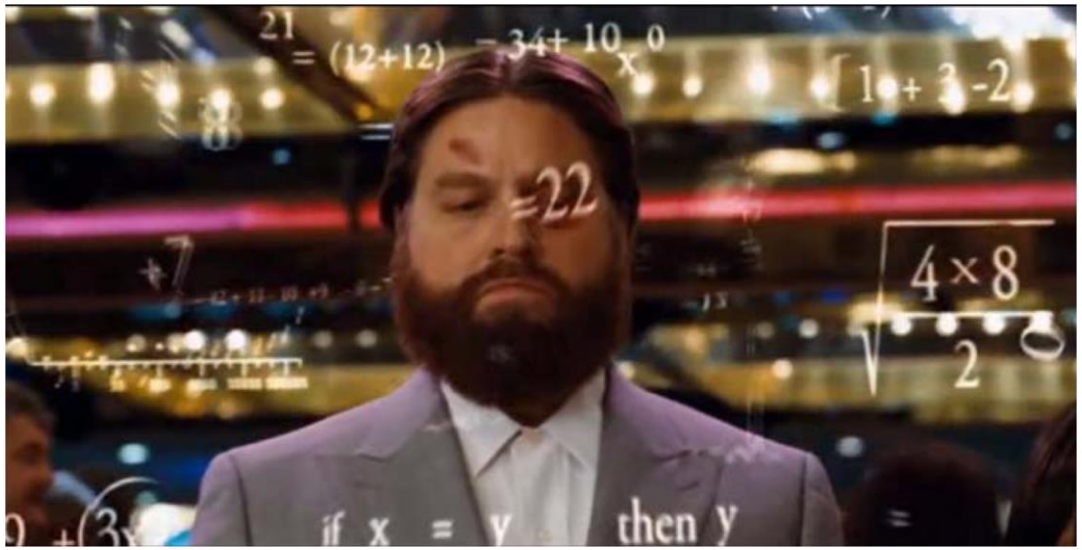
Y sí… fuiste tú.



Buenas prácticas y batallitas

Hay que ser ordenados, limpios, honestos con nosotros mismos, profesionales.

Porque te podría tocar el marrón de otro y creedme, no querréis descifrar lo que vuestro compañero escribió.



Cambiemos el if por el switch

Mismo ejercicio de las notas.

Ejercicio de clase

Escribir un programa que pida un entero entre 1 y 12, siendo cada número un mes del año. En función del mes elegido dirá si el mes es de primavera, verano, otoño o invierno.

Enero-marzo: invierno
Abril-junio: primavera
Julio-septiembre: verano
Octubre-Diciembre: otoño

Bucles

Para código que debe repetirse n veces o bien hasta que una condición lógica lo diga.

Bucles - while

```
while (expresiónLógica){
    //Repite este código
    //mientras la expresión
    //sea true
}
```

Bucles - while

```
int i = 1;
while (i <= 100){
    System.out.println(i++)
    ;
}
```

Bucle - while

1. Comprueba que la expresión lógica sea true.
2. Si es true, ejecuta el código dentro del while.
3. Vuelve al paso 1.

Bucles – expresiones lógicas

Tened en cuenta que una expresión lógica puede ser cualquier comparación

o

cualquier boolean

Bucles - while

```
boolean salir = false;
Scanner sc = new Scanner(System.in);

while (!salir){
        System.out.print("Pulsa 1 para salir del bucle: ");
        int x = Integer.parseInt(sc.next());
        if(x == 1){
              salir = true;
        } else {
              System.out.println("Aquí seguimos dentro…");
        }
}
System.out.println("Por fin he salido!!!");
```

Bucles infinitos

Mucho cuidado con los bucles infinitos. Todo bucle debe tener siempre una forma de acabarse.

Bucles infinitos

```
int i = 1;
while (i <= 100){
    System.out.println(i)
    ;
}
```

Bucles infinitos

```
int i = 1;
while (i <= 100){
        System.out.println(i);
}
```

```
int i = 1;
while (i <= 100){
    System.out.println(i--);
}
```

```
boolean salir;
while (!salir){
    System.out.println(“Holaaa”);
}
```

Ejercicio en clase

Escribe un programa que muestre un menú:

1. Suma
2. Resta
3. Multiplicación
4. División
5. Salir

Al elegir una opción 1-4 pedirá 2 números y mostrará el resultado de la operación. Tras ello, volverá a mostrar el menú. Se pide controlar errores así como que solo se puedan elegir opciones del menú.

Bucles – do while

```
do {
    //Repite este código
    //mientras la expresión
    //sea true
} while (expresiónLógica);
```

Bucles – do while

```
int i = 1;
do {
    System.out.println(i++)
    ;
} while (i <= 100);
```

Bucle - do while

1. Ejecuta el código dentro del do-while.
2. Comprueba si la condición lógica es true
3. Si es true, vuelve al paso 1

Bucles while y do while

WHILE

Primero comprueba
Después ejecuta

DO WHILE

Primero ejecuta
Después comprueba

Bucles – for

```
for(inicialización; condiciónPermanencia; incremento){
      //Repite este código
      //mientras la expresión
      //sea true
}

//Los 3 campos del for son OPCIONALES SIEMPRE
```

Bucles – for

```
int i;
for(i = 0; i<100; i++){
    System.out.println(i)
    ;
}
```

Bucle - for

1. Inicializa la variable de contador
2. Comprueba la condición lógica
3. Ejecuta el código del for
4. Incremento del contador
5. Vuelta al paso 2

Mucho cuidado con los bucles

- No toquéis dentro del bucle la variable contador a no ser que sepáis lo que se está haciendo.
- El for se utiliza cuando se sabe el número de veces que debe iterar.
- El while y el do while cuando no se sabe cuántas vueltas va a dar, lo sabe una condición booleana.

Bucles – for – declaración

```
for(int i = 0; i < 100; i++){
        System.out.println(i);
}
System.out.println(i);
```

La i no se verá fuera del for porque su ámbito es el for (donde se declara).

Aquí, la i sí que está declarada fuera del bucle, por lo que existe fuera de las llaves del for.

```
int i;

for(i = 0; i < 100; i++){
        System.out.println(i);
}
System.out.println(i);
```

Bucle – for - declaración

¿Dentro o fuera del for?

Donde mejor os parezca pero las buenas prácticas dicen que hay que declararla dentro del for siempre que no sea necesario utilizar esa variable fuera del ámbito del bucle.

Bucle – for - declaración

¿Dentro o fuera del for?

Para mí es más correcto dentro del for mientras no quede otro remedio.

Además, el garbage collector se encargará de limpiar memoria ☺

Mucho cuidado con los bucles

Cuidado con querer reutilizar variables lógicas o contador dentro del mismo ámbito.

Bucle – for - ejercicio

Crea un programa en Java que pida nombre y nota de 10 alumnos y que calcule la nota media.

Bucle – for - ejercicio

Además, dirá qué alumno tiene la nota más alta y qué alumno tiene la nota baja.

Bucles – Ejercicio típico

Crea un programa en Java que imprima los 100 primeros números pares empezando desde el 0.

Bucles – Ejercicio típico

Crea un programa en Java que imprima los 100 primeros números primos.

Número primo: es divisible entre sí mismo y la unidad.

Break, Continue, Goto

Todos rompen el flujo de los bucles.

break; → Obliga a salir del bucle inmediatamente.

continue; → Obliga a abortar la vuelta, saltar a la comprobación del resultado y volver a iterar.

goto; → Es una palabra reservada de Java, heredada de antiguos lenguajes de programación que está **prohíbida** dentro del gremio (puede llevar a muchos errores).

Ejercicios básicos de bucles

Hacedlos TODOS es por vuestro bien

Esto no se acaba aquí

A partir de ahora toca hacer MUCHOS programas

Tema 2
Programación estructurada

Prof. José de la Torre López
(adaptación Antonio Hernández)

1º Desarrollo de Aplicaciones Multiplataforma
Módulo formativo - Programación