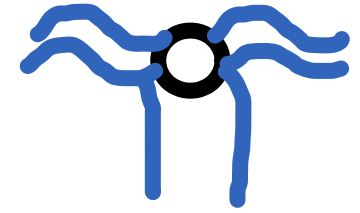


Capítulo III.I – Escribiendo simples programas

Entradas en el terminal de comando

Revisitemos la función *main*

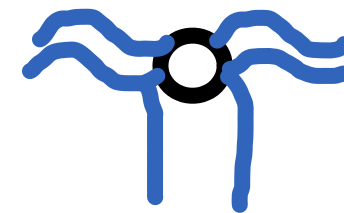


```
public static void main(String[] args)
```

Permite que nuestro programa sea más interactivo usando los comandos `javac` y `java`



Elige alguna función que hayas definido previamente y observemos cómo podemos usar `args`



Tomando arreglos como ingresos

¿Qué ocurre si mi función toma un arreglo como entrada?

Podemos procesar múltiples entradas

Elige alguna función que hayas definido previamente y observemos cómo podemos usar args

Desciframientos comunes (parsing) de `String[] args`

- `Integer miEntradaEntero = Integer.parseInt(args[0]);`
- `Double miEntradaDecimal = Double.parseDouble(args[0]);`
- `Boolean miVariableBooleana = Boolean.parseBoolean(args[0]);`



- ¿Cómo sería el desciframiento (parsing) para una variable carácter?
 - `char miVariableCaracter = args[0].charAt(0);`

```
"12" != 12;  
"42.0" != 42.0;  
"true" != true;  
"t" != 't';
```

Resumen

- Aprovecharse de *args* permite que mi programa sea más interactivo a través del terminal de comando.
- Cuidado con los tipos de variables: hay que descifrarlos para que estén empatados con los tipos especificados por mi función
- Temas recapitulados: bucles por, tipos de variables

A close-up, low-key photograph of a hand typing on a laptop keyboard. The lighting is dramatic, with the hand and keys highlighted against a dark, blurred background. The text is overlaid on the center of the image.

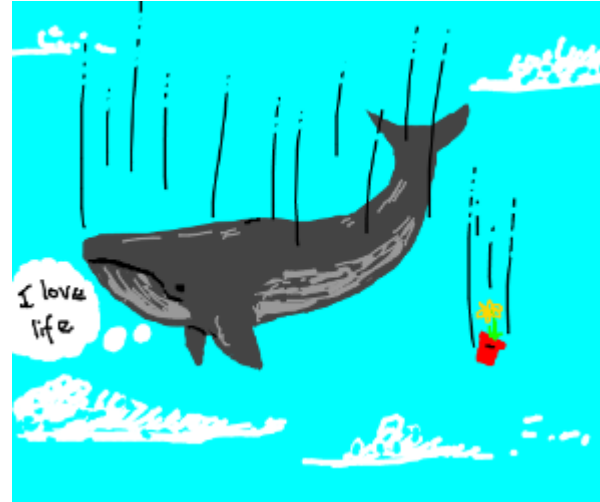
Capítulo III.II – Escribiendo simples programas

Entradas de usuario

La guía del autoestopista galáctico



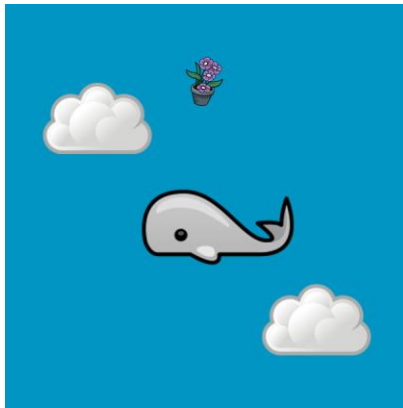
<https://hdwallpaperim.com/whale-clouds-petunias-the-hitchhikers-guide-to-the-galaxy/>



<https://drawception.com/game/aPLBhLpyQC/whale-in-space/>



<https://wallup.net/digital-art-illustration-nature-flying-whale-moby-dick-clouds-sky-fairy-tale-flowerpot-falling-the-hitchhikers-guide-to-the-galaxy/>



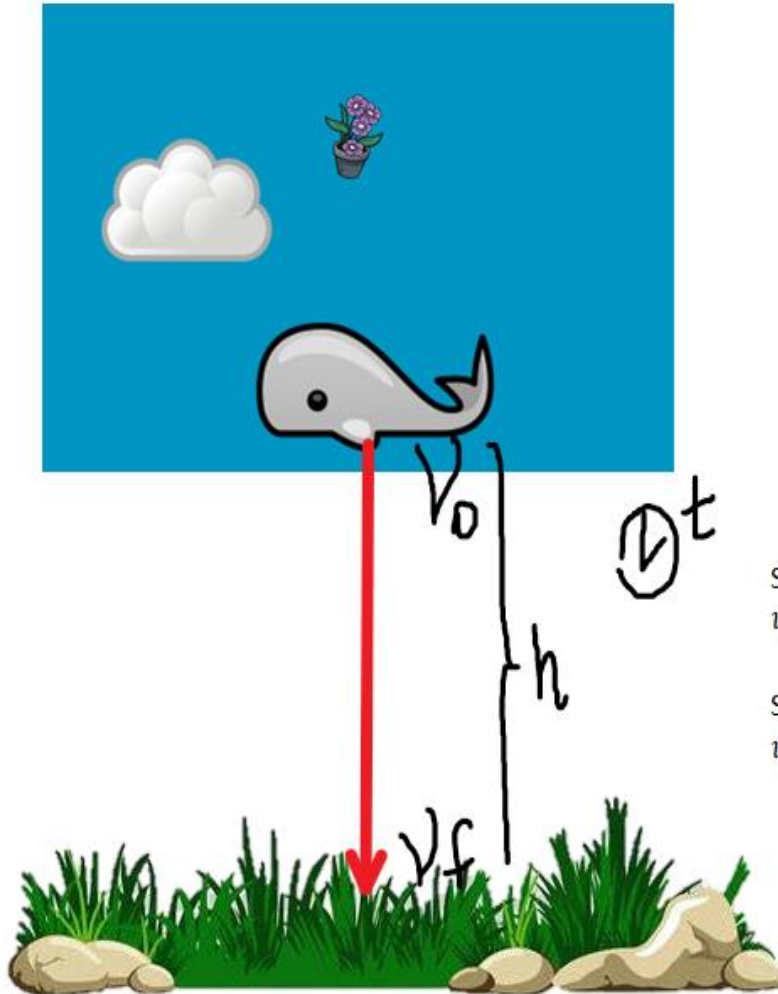
<https://recyclemeffree.org/the/35532-the-hitchhikers-guide-to-the-galaxy-whale-984-238.php>



<https://www.etsy.com/listing/481580281/hitchhikers-guide-to-the-galaxy-whale>


¿A qué velocidad se van a estrellarse en el suelo?

Explicación breve



Si medí el tiempo de caída, uso
 $v_f = v_0 + g * \text{tiempo}$

Si medí la altura de caída, uso
 $v_f^2 = v_0^2 + 2 * g * \text{altura}$

Fórmula	No incluye
$v_f = v_0 \pm gt$	Sin h
$h = v_0 t \pm \frac{gt^2}{2}$	Sin v_f
$h = \left(\frac{v_0 + v_f}{2}\right) \cdot t$	Sin g
$v_f^2 = v_0^2 \pm 2gh$	Sin t
$h_n = v_0 \pm \frac{g}{2}(2n - 1)$	

<https://matemovil.com/wp-content/uploads/2020/01/f%C3%B3rmulas-ca%C3%ADda-libre.jpg>

Suposiciones: Caída libre, m/s, $g = 9.81 \text{ m/s}^2$

Clase Scanner

```
1. Scanner entradaUsuario = new Scanner(System.in);
2. System.out.println("Mensaje orientativo sobre qué
   ingresar...");
```

- `entradaUsuario.next();` // cadena
- `entradaUsuario.next().charAt(0);` // caracter
- `entradaUsuario.nextDouble();` // decimal (similar a `ParseDouble`)
- `entradaUsuario.nextInt();` // entero (similar a `ParseInt`)
- `entradaUsuario.nextBoolean();` // booleana

- Código...

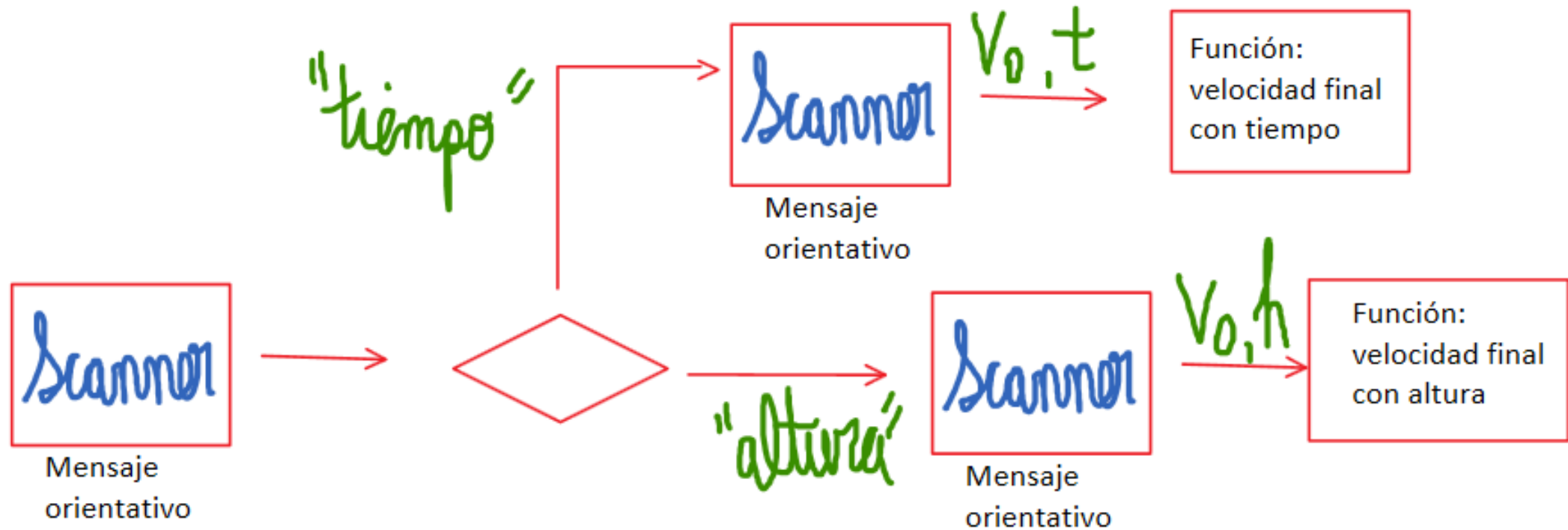
```
3. entradaUsuario.close(); // colgarlo
```



```
n = eval(input("Mensaje orientativo sobre qué
ingresar..."))
```

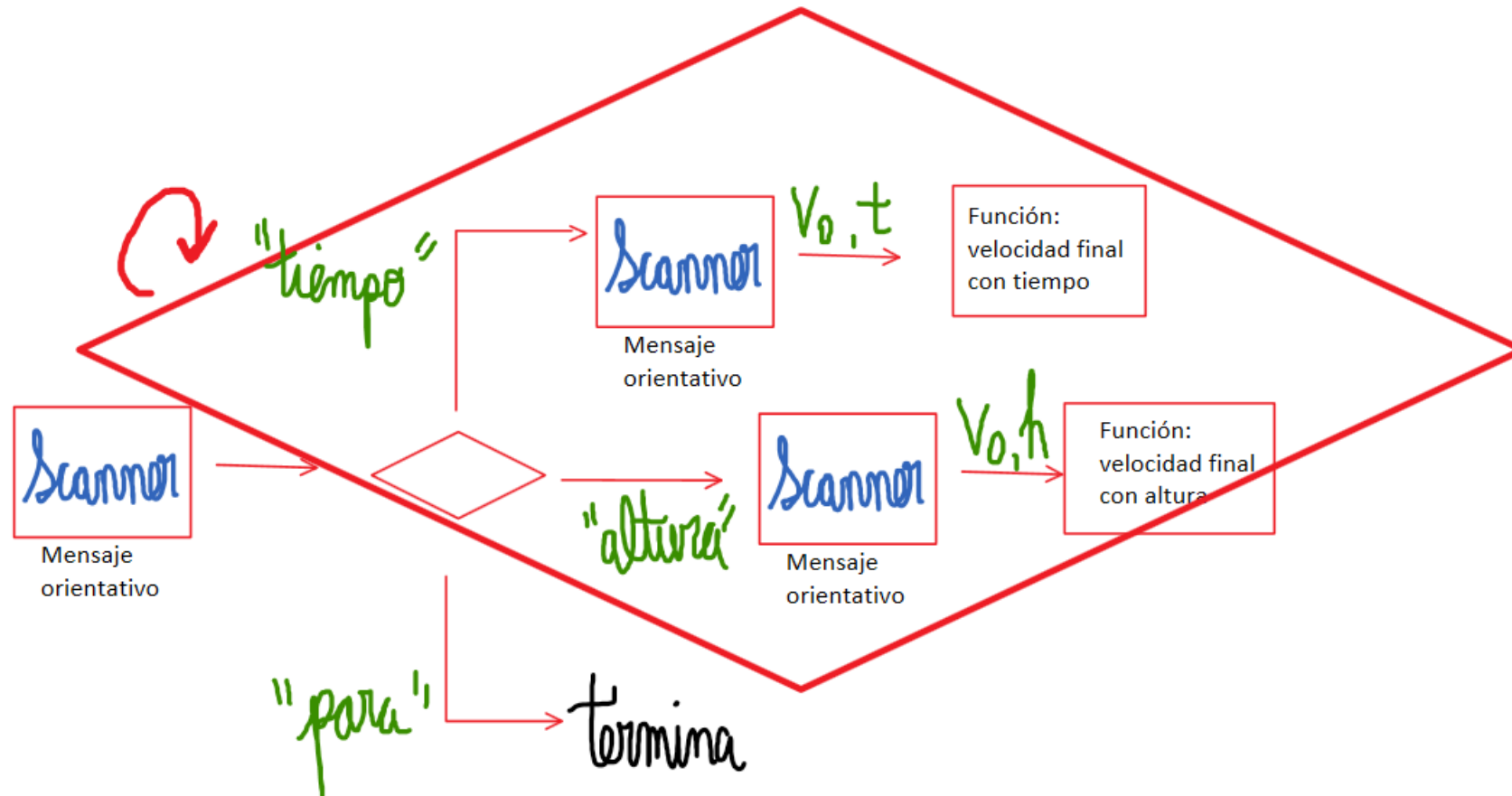
Posible esquematización

- Inténtalo

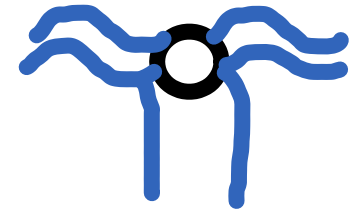


Leyendo múltiples entradas

- Podemos trabajar *en vivo* con múltiples entradas usando un bucle mientras (porque no sabemos cuántas entradas hay de antemano)



Comentarios finales



Puedes expandir este pequeño programa:

- ¿Qué pasa si quiero que devuelva un mensaje optimizado a 42 m/s?
- Si la gravedad no es 9.81 ms^{-2}
- Diferentes unidades (km/min)
- No existe solución “correcta”, puedes experimentar

Intenta con tu **propio** problema de la situación de la vida real:

- lista de compras calcular el precio de descuento, n entre otras tareas que puedan ser mecanizadas.

Tiempo	tiempo	altura	para
--------	--------	--------	------



¿Velocidad y
altura?



¿Velocidad y
altura?



¿Velocidad y
tiempo?

Tiempo 0 8	altura 9 25	tiempo 8 7	para
------------	-------------	------------	------



Calcular



Calcular



Calcular

Resumen

- La clase SCANNER es útil para leer entradas personalizadas del usuario
- Podemos ahora escribir programas más interactivos y orientativos
- Temas recapitulados: bucles mientras, estructuras de control, funciones predefinidas, variables

Adicional

¿Por qué Java es conocido como un Lenguaje de Programación orientada a objetos?



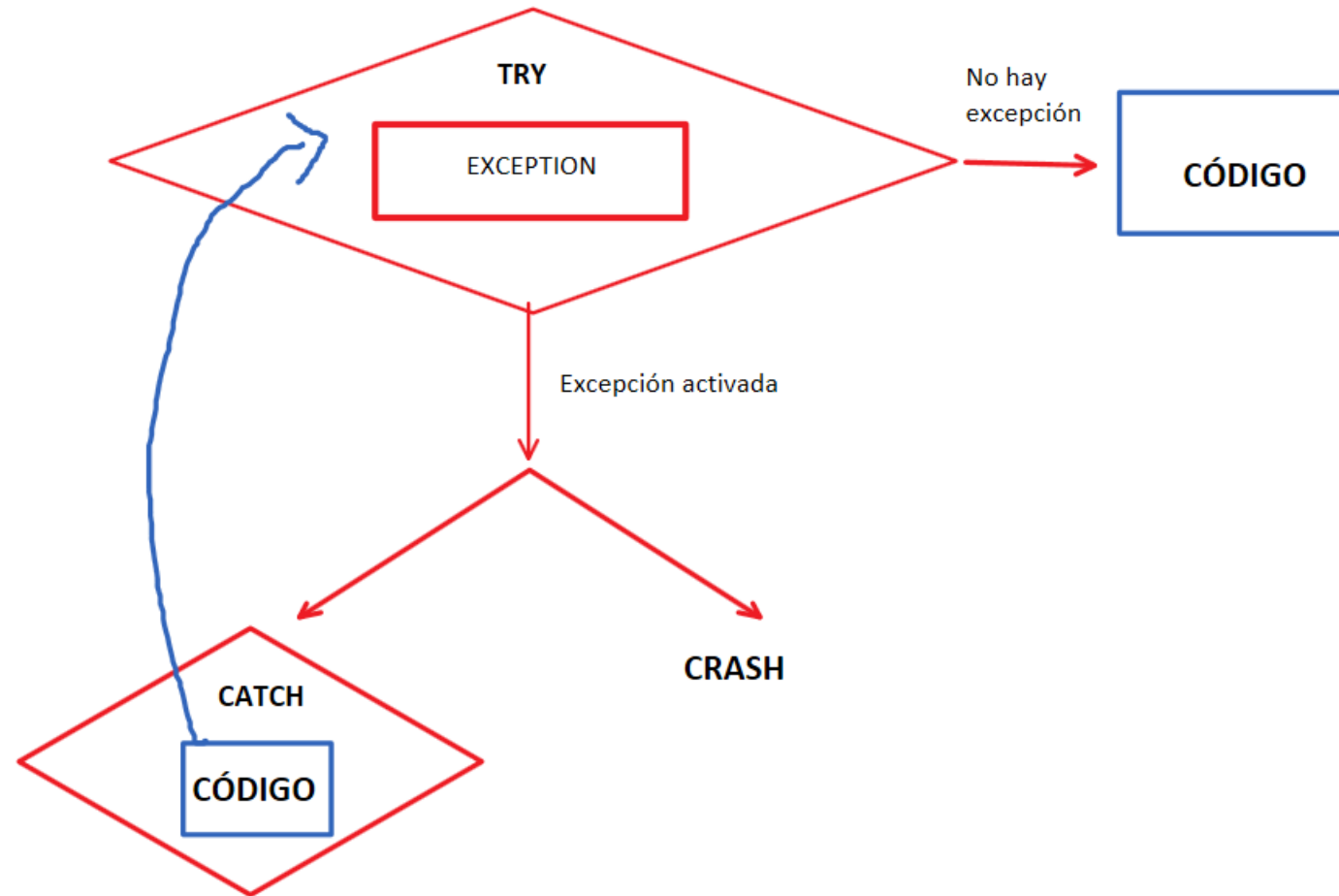
Capítulo III.III – Escribiendo simples programas

Código más robusto

Código robusto

- El programa es “tolerante” a argumentos ilegales: no crashea
- Observemos un error a través de un simple programa que divide dos decimales que son entradas del usuario

Esquema de try-catch

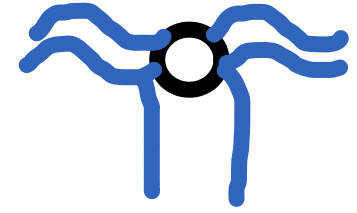


Excepciones



- Excepciones comunes
 - *ArrayIndexOutOfBoundsException*
 - *ArithmeticException*
 - *NullPointerException*
 - *IllegalArgumentException*
 - *InputMismatchException*
 - ...
- Excepciones personalizadas
 - `throw new IllegalArgumentException()`

Comentarios



Puedes editar funciones pasadas
para que sean más robustas



Desafío personal: no permitas que
tu función crashee

Resumen

- Esquematizar antes de programar con el fin de anticipar errores
- Podemos plantear nuestras propias excepciones, capturarlas y lidiar con estas usando try-catch.
- Código robusto también se trata de escribir un programa que no crashee y tolere argumentos ilegales.