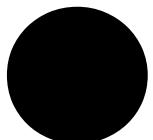


# Java desde C

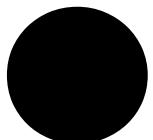
**UNRN**

Universidad Nacional  
de Río Negro





# **Dudas de la corrección del TP1**



# Funciona todo

# Sobre el TP

Conan está en la versión 0.1

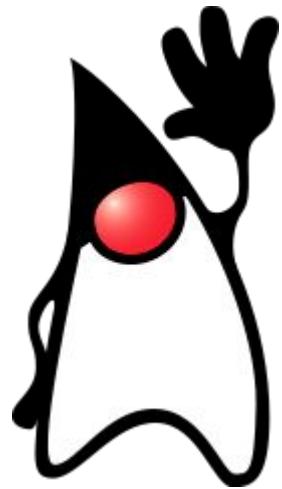


1. Cómo deben ser las entregas
2. Primer contacto con las herramientas
3. La mecánica de la corrección
4. Primer informe
5. Expectativas generales



**Abran  
hilo**

**O contesten la corrección si tienen más dudas**



# Java desde C



Análisis

# Veamos un HolaApp . java

# HolaApp.java

```
public class HolaApp {  
    public static void main(String[] args) {  
        System.out.printf("Hola %s!\n", "mundo");  
    }  
}
```

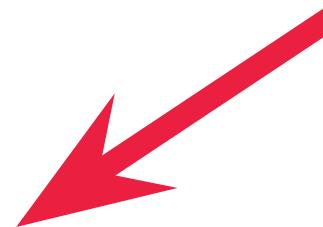
# HolaApp.java

Tienen que tener el mismo nombre

```
public class HolaApp {  
    public static void main(String[] args) {  
        System.out.printf("Hola %s!\n", "mundo");  
    }  
}
```

# Un ejemplo mínimo, con el punto de entrada

```
public class HolaApp {  
    public static void main(String[] args) {  
        System.out.printf("Hola %s!\n", "mundo");  
    }  
}
```



# Unas palabras tempralmente mágicas

```
public class HolaApp {  
    public static void main(String[] args) {  
        System.out.printf("Hola %s!\n", "mundo");  
    }  
}
```

# Esto se parece a algo de C

```
public class HolaApp {  
    public static void main(String[] args) {  
        System.out.printf("Hola %s!\n", "mundo");  
    }  
}
```

# Sin decirles nada, ¿a qué se parece?

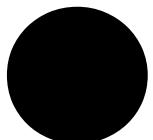
```
public class HolaApp {  
    public static void main(String[] args) {  
        System.out.printf("Hola %s!\n", "mundo");  
    }  
}
```

# ¡Hay printf!

```
public class HolaApp {  
    public static void main(String[] args) {  
        System.out.printf("Hola %s!\n", "mundo");  
    }  
}
```

**¡A la terminal!**





# **WTF es un package**

**Es una forma de  
agrupar partes  
del programa**

**\*Por el momento**

**Aunque de momento  
nuestros programas  
*no tienen* partes**

# **La plataforma de Java sí**

**Lo que está en otro  
paquete se trae con**

**import dirección.del.paquete;**



# Menos lo contenido en `java.lang.*`

*{Vamos a ver un poco de  
su contenido en un rato}*

La clave está en la Documentación del JDK



**¿Preguntas?**

# *Cuestiones de estilo*

**1**

# **Los nombres de los class van en CamelloCase**

**2**

# **Variables y argumentos van en dromedarioCase**

**3**

**Los identificadores  
válidos son solo con  
alfabéticos  
[azAZ]**

# *Comentarios*

```
/**  
 * comentario de documentación  
 */  
class HolaApp {  
    public static void main(String[] args) {  
        //comentario de linea  
        System.out.printf("Hola %s!\n", mundo);  
    }  
}
```

# Variables y tipos de datos

# Tipos de datos primitivos



**byte**  
**8–bits con signo**

**Números**

**int**  
**32–bits con signo**

**short**  
**16–bits con signo**

**long**  
**64–bits con signo**

*Decimales*

**float**

**punto flotante de 32-bits**

**double**

**punto flotante de 64-bits**

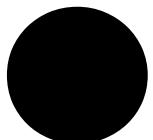
Lógicos

**boolean**  
**true/false**

**Carácteres**

**char**

**Carácter Unicode de 16-bits**



# ¿Primitivos?

Como en C...

**Solo contienen el  
valor**

**[Ya vamos a ver cuáles no son primitivos]**

# Variables Locales

# Declaración e inicialización

Igual que en C

tipo

nombre

```
int numero;  
int otro = 100;
```

inicialización

# Inferencia de tipo - tipo implícito

```
var i = 10;
```

Sí o sí inicializada

i sigue siendo  
de un solo tipo

[Esto es más útil, más adelante]

# Alcance de variables (scope)

```
{  
    int i = 5;  
}  
  
{  
    int i = 10;  
}  
System.out.println(i);
```

¿Cuanto vale i?

# Alcance de variables (scope)

```
{  
    int i = 5;  
}  
  
{  
    int i = 10;  
}  
System.out.println(i);
```

Una i

Otra i

i no es conocida en este alcance

# **“Funciones”**

**Más adelante vamos a ver que son realmente**

# ¿Como se tiene que llamar el archivo que contiene este ejemplo?

```
class FuncApp {  
    public static void main(String[] args) {  
        int resultado = suma(10,20);  
        System.out.printf("El resultado es %d!\n", resultado);  
    }  
  
    public static int suma(int a, int b) {  
        return a + b;  
    }  
}
```

\*Otra cosa que vamos a resolver después

# *Cuestiones de estilo*

# 4

# Los nombres de las funciones van en dromedarioCase

**5**

**Los identificadores  
deben ser  
descriptivos.\***

# 6

## Un solo return por función



**¿Preguntas?**



Para interactuar con nuestros programas

---

# **El “printf” de Java (igual que el de C)**

# Las cadenas, no son 'primitivas'

```
System.out.printf("Hola %s, %d%n", "mundo", 2024);
```

Cadena de formato

valores

%n es el \n apropiado

*ya que  
estamos*

# Cadenas Strings

# Declaración e inicialización

```
String micadena = "Hola mundo!";
int largo = micadena.length();
String otracadena = micadena + " estudiante";
```

¡como un 'struct'!

concatenación

¡Vamos a ver mucho más de este tema!

# Literales de bloque

```
String poema = """
```



```
    Ahora muchos lenguajes hablo y escribo  
    Pero muchos no entienden lo que digo  
    Solo un complejo circuito  
    Es capaz de razonar conmigo  
""";
```



# Para recuperar un carácter en una posición

```
String micadena = "Hola mundo!";  
char letra = micadena.charAt(0);
```

# Una cadena con formato hacia una variable

```
String cadena = String.format("Hola %s, %d", "mundo", 2024);
```

**¡A la terminal!**



# **Tomando entrada del usuario**

```
import java.util.Scanner;

class ScanApp {
    public static void main(String[] args) {
        Scanner lector = new Scanner(System.in);
        int var = lector.nextInt();
        System.out.printf("Hola %d!\n", var);
    }
}
```

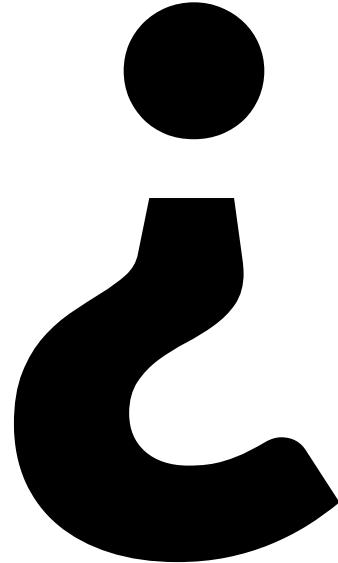
*Para otros  
valores*

**nextInt**

**nextString**

**nextDouble**

**nextFloat**



**Y si ingreso una  
cadena en lugar  
de un número**



¡BOOM!



¿Qué pasó?

Exception in thread "main" java.util.InputMismatchException  
...algunas líneas extras  
at ScannerApp.main(ScannerApp.java:9)

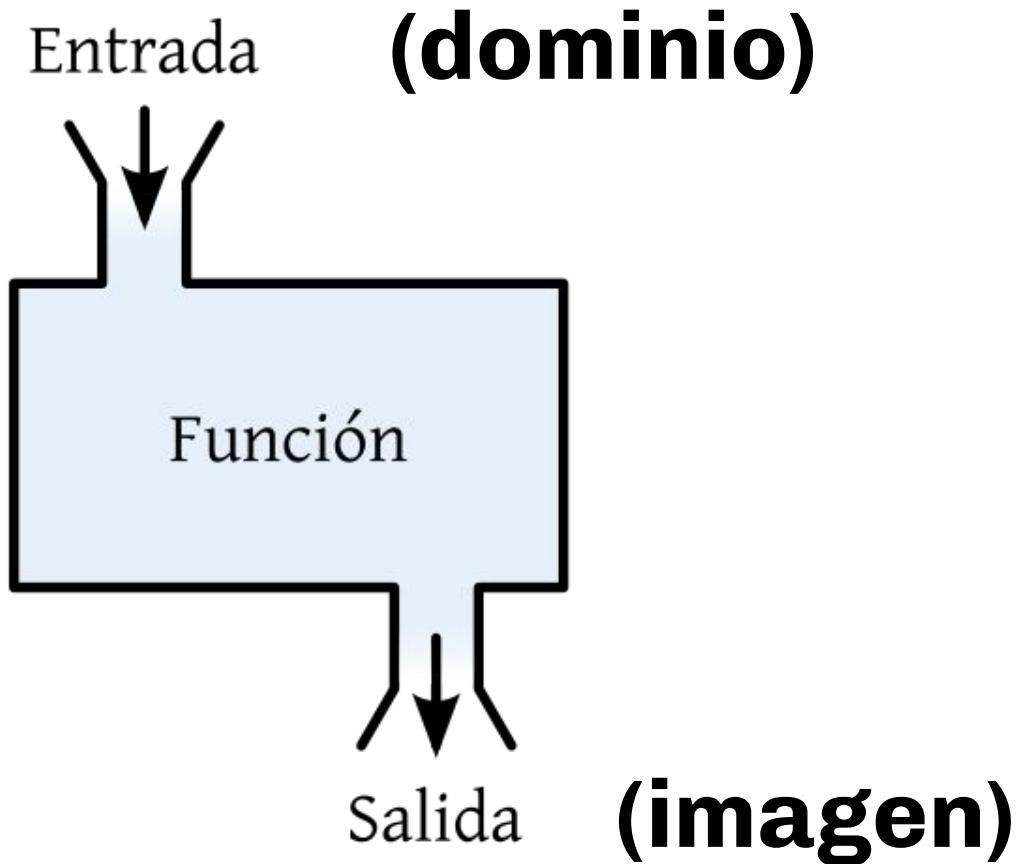
En qué lugar;  
archivo:linea

**Menos punteros y  
estructuras**

**El resto funciona  
*casi* igual**

# "Funciones"

# ¿Como era una función?



# Dada una función en C

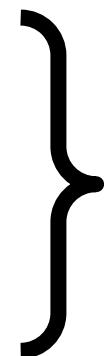
Por  
ejemplo

```
int suma(int op1, int op2)
{
    int suma = op1 + op2;
    return suma;
}
```

# Su equivalente en Java

*Mucho no  
cambia*

```
public static int suma(int op1, int op2)
{
    int suma = op1 + op2;
    return suma;
}
```



Y dentro de una  
**public class**

# La sintaxis general



```
public static tipoRetorno nombre(tipoArgumento identArgumento)
{
    return valorRetorno;
}
```

**Técnicamente no  
son funciones, ya  
vamos a ver que  
son**

# *Cuestiones de estilo*

# 7

# Las funciones no van con printf o Scanner

[a no ser que sea  
explícitamente su propósito]

**8**

**Todas las funciones  
deben estar  
completamente<sup>1</sup>  
documentadas**



**¿Preguntas?**

# Constantes

**final tipo NOMBRE = valor;**

```
final int MAXIMO = 10000;
```

# *Cuestiones de estilo*

9

**Las constantes van  
en mayúsculas, con  
SNAKE\_CASE**

# Operadores

# asignación

```
variable = valor;
```

# Posfijos

expresión++  incremento  
expresión--  decremento

# Lógico

a && b  Operador Y

a || b  Operador O

!  negación lógica

# aritméticos

-a       negación de signo

a + b     suma

a - b     resta

a \* b     multiplicación

a / b     división

a % b     resto de la división

# relacionales

$a > b$   mayor que

$a < b$   menor que

$a \geq b$   mayor o igual que

$a \leq b$   menor o igual que

$a == b$   igual que

$a != b$   distinto que

# Estructuras de control

# Variaciones sentencia condicional

```
if (condicion) {  
    bloque;  
}
```

```
if (condicion) {  
    bloque;  
} else {  
    bloque;  
}
```

```
if (condicion) {  
    bloque;  
} else if (otra) {  
    bloque;  
} else {  
    bloque;  
}
```

# Selección switch

```
int cuenta = 3;  
System.out.print("la variable cuenta es ");  
switch (cuenta) {  
    case 0:  
        System.out.println("cero");  
        break; ← ¡muy importante!  
    case 1:  
        System.out.println("uno");  
        break; ← ¡muy importante!  
    default:  
        System.out.println("negativo o mayor a uno");  
        break;  
}
```

# Switch mejorado (>JDK 17)

```
String dia = "lunes";

int numeroDia = switch (dia) {
    case "lunes" -> 1;
    case "martes" -> 2;
    // ... case's para los demás días
    default -> 0; // y este opcional para los desconocidos
};
```

Para transformación de valores

# También admite código dentro



```
String dia = "lunes";  
  
int numeroDia = switch (dia) {  
    case "lunes" -> 1;  
    case "martes" -> 2;  
    // ... resto de los case's  
    default -> {  
        System.out.println("día desconocido");  
        yield 0;  
    }  
};
```



'produce' 0

**yield es parecido a un return**

—

# Estructuras de repetición

# Lazos indefinidos

```
int i = 0;  
while (i < 50) {  
    System.out.printf("%d, ", i);  
    i++;  
}
```

# Lazos definidos

```
for (int i = 10; i > 0; i++) {  
    System.out.printf("%d, ", i);  
}
```

**Hay una variación de este lazo que vamos a ver más adelante**

# Lazos definidos

```
for (int i = 10; i > 0; i++) {  
    System.out.printf("%d, ", i);  
}  
System.out.printf("\n %d ", i);
```

¿Qué muestra este programa?

**Igual que en C  
(menos el repeat)**

# *Cuestiones de estilo*

**10**

# **Un espacio antes y después de los operadores**

**11**

**Sin break y continue  
en su lugar, usen banderas**

**12**

**Sin usar la  
asignación  
compuesta  
 $(+=, *=, \text{etc})$**



**¿Preguntas?**

# Tipos no tan primitivos

¿Cuál es su  
tipo primitivo?

Ya tuvieron un  
contacto con String

# Tipos de Números



- |               |                                                |
|---------------|------------------------------------------------|
| <b>byte</b>   | <input type="checkbox"/> <u><b>Byte</b></u>    |
| <b>short</b>  | <input type="checkbox"/> <u><b>Short</b></u>   |
| <b>int</b>    | <input type="checkbox"/> <u><b>Integer</b></u> |
| <b>long</b>   | <input type="checkbox"/> <u><b>Long</b></u>    |
| <b>float</b>  | <input type="checkbox"/> <u><b>Float</b></u>   |
| <b>double</b> | <input type="checkbox"/> <u><b>Double</b></u>  |



**Agregan código y  
son en realidad  
*otra cosa*  
que vamos a ver más adelante**

# Contiene cosas como:

`Integer.MAX_VALUE`

`Integer.valueOf(String numero)`

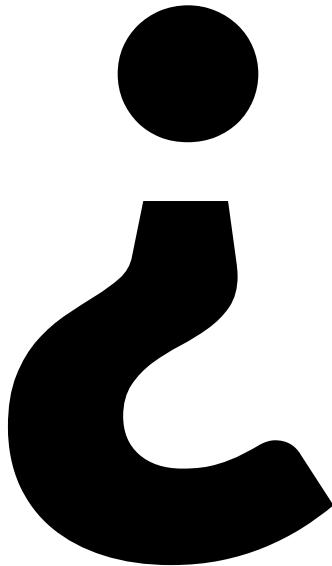
`Integer.toString(int valor, int base)`

*Los otros  
wrappers  
son similares*

*Conversión 'String' a 'int'*

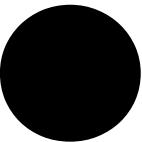
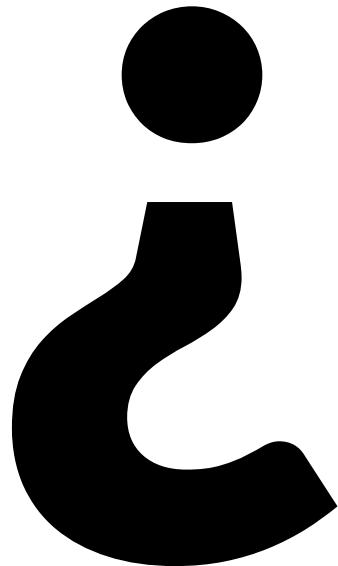
*Conversión 'int' a 'String'*

[Javadoc base /java.lang.Integer](#)



**pero...  
¿por qué?**





# SunSPOT



2007

[https://en.wikipedia.org/wiki/Sun\\_SPOT](https://en.wikipedia.org/wiki/Sun_SPOT)



**¿Preguntas?**

# arrays y matrices

**Una secuencia *mutable*\* y  
de elementos de *un tipo*  
con un largo fijo**

**tipo[] identificador;**



```
int[] arr = {1, 2, 3};
```

**Tamaño automático**

```
int[] arr = new int[10];
```

Tamaño manual, inicializado en cero

---

# Con información adicional

```
int[] arr = new int[100];  
arr.length
```



---

# Uso con un lazo definido

```
int[] arr = new int[100];
for (int i = 0; i < arr.length; i++){
    arr[i] = i;
}
```

# Lazo 'for-each'

```
int[] arr = new int[100];
for (int valor : arr) {
    System.out.println(valor);
}
```

**Pero solo permite leer del arreglo (y sin la posición)**

# Admite inferencia de tipo

```
int[] arr = new int[100];
for (var valor : arr) {
    System.out.println(valor);
}
```

# Declaración e inicialización de matrices

```
int[][] matrixUno;
int[][] matrixDos = {{1,2},{3,4}};
var matrixTres = new int[3][4][8]; //j3D!
```



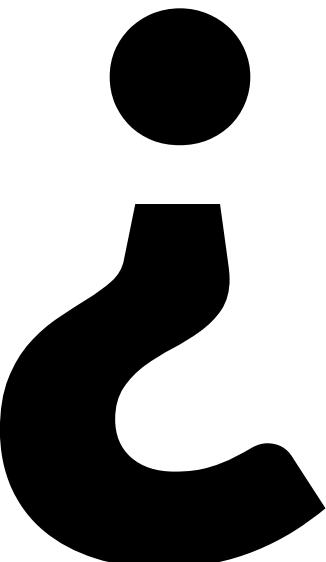
**¿Preguntas?**

# A probar nuestro código

(el primer enfoque)

# La estructura de una función de pruebas

```
public static void test_suma_positivos(){
    int argumento1 = 10;
    int argumento2 = 20;
    int esperado = 30; // argumento1 + argumento2 esta bien
    int resultado = suma_lenta(argumento1, argumento2);
}
```



**¿Cuántas  
funciones hacen  
falta?**



**Suficientes para  
ejercitar todas las  
decisiones (if y lazos)**



**¿Preguntas?**

# TP2

## Desde C hacia Java



**Abran  
hilo**

**unrn.edu.ar**

**UNRN**

Universidad Nacional  
de Río Negro



| unrnionegro



# Con gradle es más difícil

```
class ArgsApp {  
    public static void main(String[] args) {  
        for (int i=0; i < args.length; i++) {  
            System.out.printf("%d-%s\n", i, args[i]);  
        }  
        System.out.println("al final");  
    }  
}
```



# Una observación con los package / paquetes

**Lo que está en `java.lang.*` esta disponible sin import**

```
class ArgsApp {  
    public static void main(String[] args) {  
        int valor;  
        for (int i=0; i < args.length; i++) {  
            System.out.printf("%d-%s\n", i, args[i]);  
            valor = Integer.decode(args[i]);  
        }  
        System.out.println("al final");  
    }  
}
```