

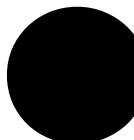
Java desde C

UNRN

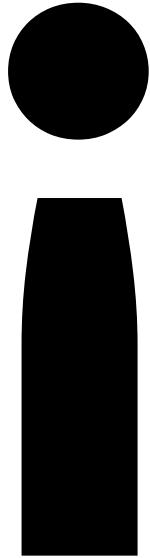
Universidad Nacional
de **Río Negro**



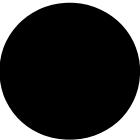
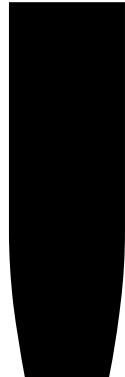
Dudas de la corrección del TP1



Funciona todo



Funciona todo





**Abran
hilo**

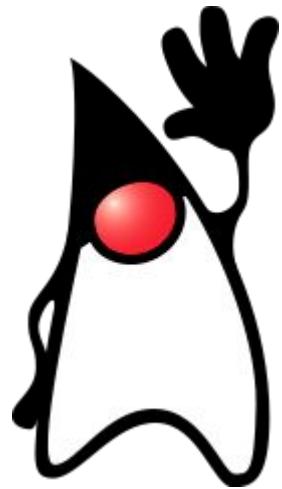
O contesten la corrección si tienen más dudas

Sobre el TP

Conan está en la versión 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



Java desde C



Análisis

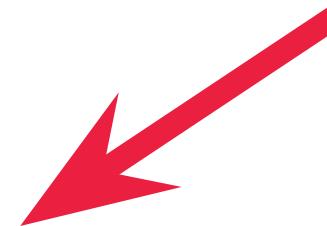
Veamos el HolaApp.java

HolaApp.java

```
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 temporalmente *mágicas*

```
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");  
    }  
}
```

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?

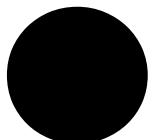
```
public class HolaApp {  
    public static void main(String[] args) {  
        System.out.printf("Hola %s!\n", "mundo");  
    }  
}
```

¡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**

**Y permite dos cosas
con el mismo nombre**

(quedan en contextos diferentes)

**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 [dub.sh/p2/jdk]



¿Preguntas?



Cuestiones de estilo

Los nombres de las class van en CamelloCase

Variables y argumentos van en dromedarioCase

**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

int
32-bits con signo

Números

short
16-bits con signo

long
64-bits con signo

Variables y literales

```
byte edad = 25;  
short anio = 2025;  
int poblacion = 15000;  
long distancia = 123456789L;
```

Decimales

float

punto flotante de 32-bits

double

punto flotante de 64-bits

Variables y literales

```
float temperatura = 36.5f;  
double pi = 3.141592653589793;
```

Lógicos

**boolean
true/false**

Variables y literales

```
boolean esValido = true;  
boolean esMayor = edad >= 18;
```

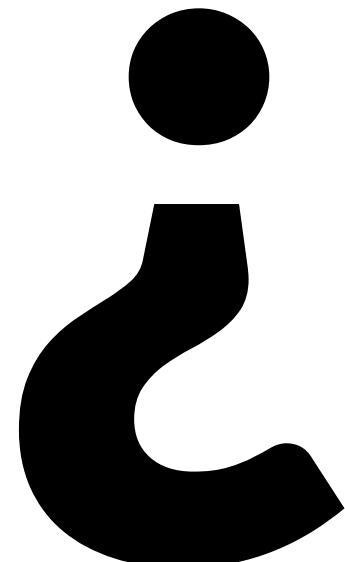
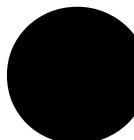
Carácteres

char

Carácter Unicode de 16-bits

Variables y literales

```
char letra = 'a'
```



¿Primitivos?

Como en
C

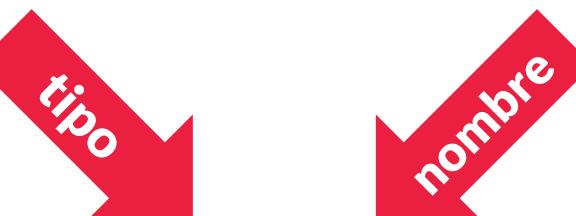
**Solo contienen el
valor**

[Ya vamos a ver cuáles no son primitivos]

Variables Locales

Declaración e inicialización

Igual que en C



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



Inferencia de tipo / tipo implícito

```
var i = 10;
```

SÍ O SI
inicializada

i sigue siendo
de un solo tipo

[Esto es más útil, más adelante]

Alcance de variables (scopes)

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

¿Cuanto vale i?

Alcance de variables (scopes)

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

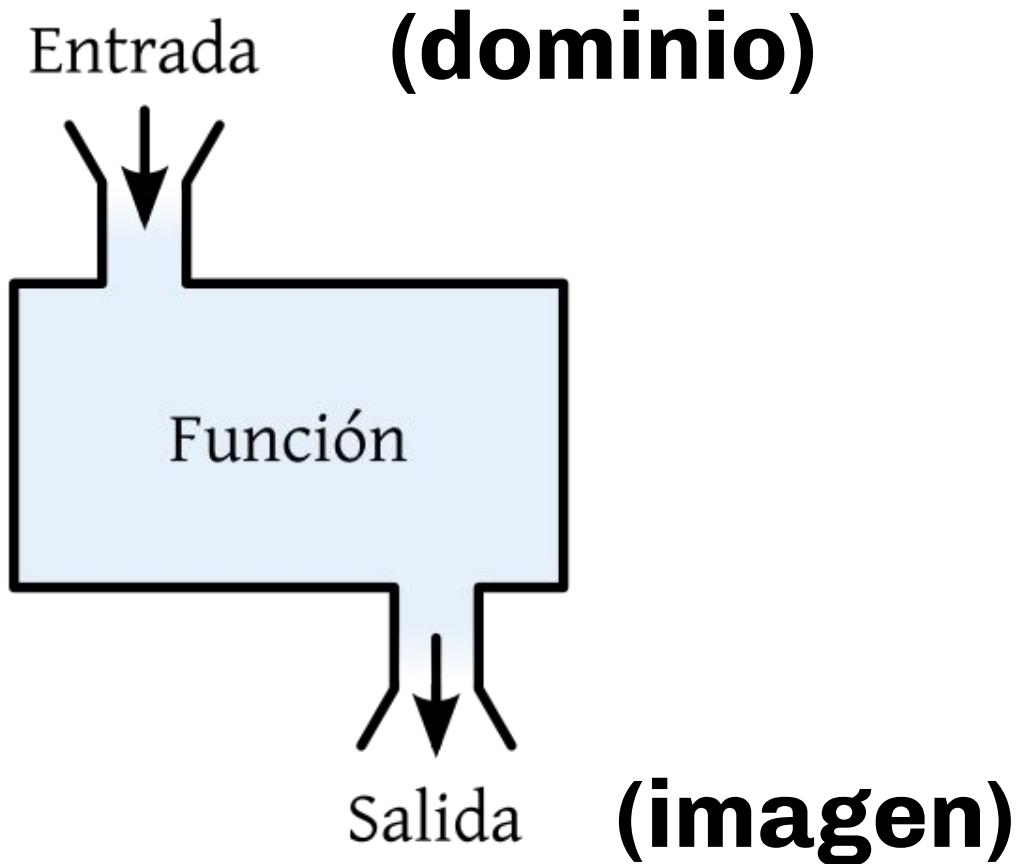
Una i

Otra i

i no es conocida en este alcance

“Funciones”

¿Como era una función?



Dada una función en C

Por
ejemplo

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

Su equivalente en Java

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



*Mucho no
cambia*

Y dentro de una
public class

La sintaxis general



Magia

```
public static tipoRetorno nombre(tipo identificador)
{
    return valorRetorno;
}
```

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

son

El ejemplo completo

```
class FuncionApp {  
    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;  
    }  
}
```

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

Cuestiones de estilo

Los nombres de las funciones van en dromedarioCase

**Los identificadores
deben ser
descriptivos*.**

Un solo return por función

**Todas las
funciones deben
estar
completamente¹
documentadas**

La forma base de un comentario de documentación

```
/**  
 * Descripción de una línea de lo que hace.  
 * @param identificador para que esta o que se hace con el  
 * @returns que resultados da la función  
 */  
public static tipoRetorno nombre(tipo identificador)  
{  
    return valorRetorno;  
}
```

A yellow cube with two large white question marks on its faces, resembling a power-up item from the Super Mario video game series.

¿Preguntas?



Entrada /Salida

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", 2025);
```

The diagram shows the Java code `System.out.printf("Hola %s, %d%n", "mundo", 2025);`. Three red arrows point from text labels to specific parts of the code:

- A red arrow pointing to the first placeholder `%s` has the text "Cadena de formato" written on it.
- A red arrow pointing to the second placeholder `%d` has the text "valores" written on it.
- A red arrow pointing to the new line character `\n` has the text "%n es el \n apropiado" written on it.

*ya que
estamos*

Cadenas Strings

Declaración e inicialización

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

¡como un 'struct'!

concatenació

¡Vamos a ver mucho más de este tema!

Literales de bloque

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

en linea propia

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

en linea propia

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);
```

Un printf a cadena

**¡A la
terminal!**



A yellow cube with two large white question marks on its faces, resembling a power-up item from the Super Mario video game series.

¿Preguntas?



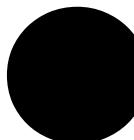
Tomando entrada del usuario

```
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);
        lector.close();
    }
}
```

nextInt
nextString
nextDouble
nextFloat

*Para otros
valores*



**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
igual**



Cuestiones de estilo

**Las funciones no
van con printf o
Scanner
[a no ser que sea
explícitamente su propósito]**

A yellow cube with two large white question marks on its faces, resembling a power-up item from the Super Mario video game series.

¿Preguntas?



Constantes

final tipo NOMBRE = valor;

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

Cuestiones de estilo

**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

Cuestiones de estilo

Un espacio antes y después de los operadores

Sin usar la asignación compuesta

(`+=`, `*=`, etc)

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;  
case 1:    ← ¡muy importante!  
    System.out.println("uno");  
    break;  
default:   ← ¡muy importante!  
    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;  
    }  
};
```



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

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



¿Preguntas?



Tipos no tan primitivos

¿Cuál es su
tipo
primitivo?

Ya tuvieron un
contacto con String

Tipos de Números



- byte** Byte
- short** Short
- int** Integer
- long** Long
- float** Float
- double** Double



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

Contiene cosas como:

`Integer.MAX_VALUE`

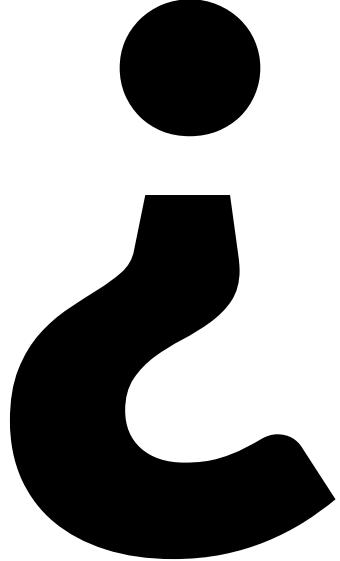
`Integer.valueOf(String numero)`

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

Conversión 'String'a'int'

Conversión 'int'a'String'

*Los otros
wrappers
son
similares*



pero...
¿por qué?



**No hay una
explicación oficial
al respecto**

SunSPOT



2007

https://en.wikipedia.org/wiki/Sun_SPOT



¿Preguntas?



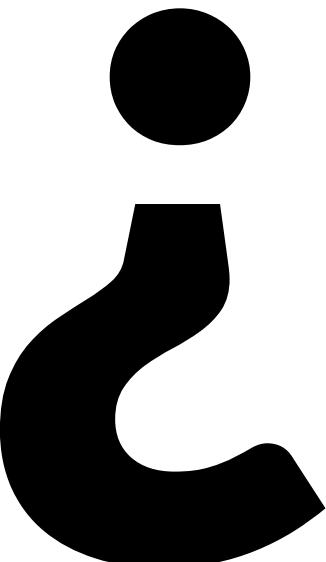
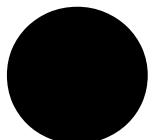
Testing base

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?



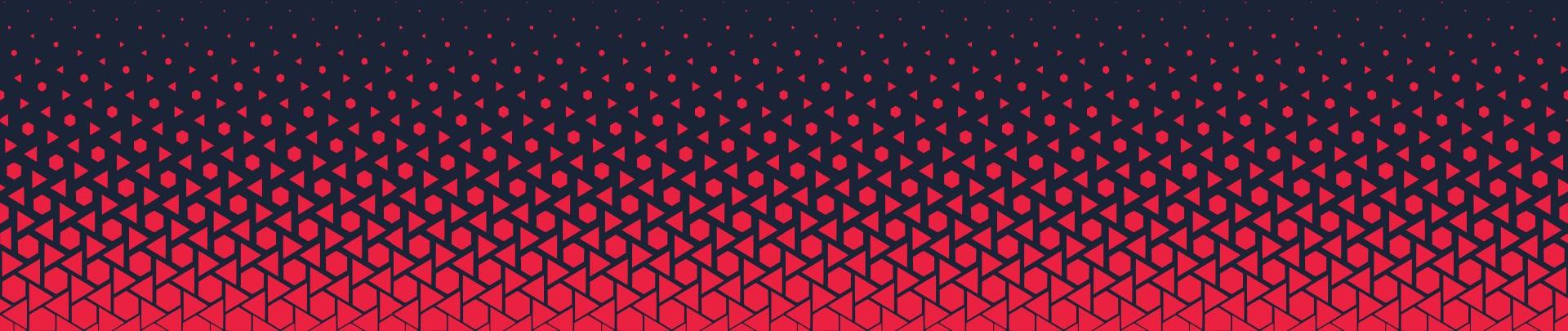
TP2

Desde C hacia Java



**Abran
hilo**

Hasta la próxima





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");  
    }  
}
```