

# Cuestiones de estilo

**UNRN**

Universidad Nacional  
de Río Negro



**2025**



# Cuestiones de estilo



**Para  
tenerlas a  
mano**

**y referenciarlas  
en las  
correcciones.**

# Aunque algunas las vé

Dredd



0x0000

*Cuestiones de estilo*

**Sin errores de  
ortografía**

**IntelliJ viene con corrector  
ortográfico**



0x0001

*Cuestiones de estilo*

**Los nombres de las  
class van en  
CamelCase**

# Variables, funciones y argumentos van en dromedarioCase



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

**Los nombres de las  
*funciones* van en  
dromedarioCase**

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

0x0006

*Cuestiones de estilo*

# Un solo return por función

0x0007

*Cuestiones de estilo*

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

0x0008

*Cuestiones de estilo*

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

0x0009

*Cuestiones de estilo*

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

0x000A

*Cuestiones de estilo*

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



0x000B

*Cuestiones de estilo*

**Sin usar la  
asignación  
compuesta**

**(+=, \*=, etc)**

0x000C

*Cuestiones de estilo*

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

0x000D

*Cuestiones de estilo*

**Debe tener el  
mismo nombre que  
la `class` con `Test` al  
final**

# La estructura de un test individual debe ser:

**preparar / ejecutar / comprobar / limpiar**  
**Con identificación y mensajes**

0x000F

*Cuestiones de estilo*

# Una llamada a función en cada caso de prueba\*\*

\*salvo que estén *fuertemente* conectadas  
como en DivisionLenta

**No atajar la  
excepción si no es  
posible tomar una  
decisión**

**El main de un  
programa no debe  
dejar pasar  
excepciones de tipo**

**Qué familia de  
excepciones se  
eligió debe de  
estar  
documentada**



**Usar argumentos  
como variables  
solo si no cambia  
su significado**

**No atajar una  
excepción lanzada  
en el mismo bloque**

**No convertir  
excepciones con  
tipo a sin tipo**

**Todas las  
excepciones que  
lancemos deben de  
estar documentadas  
@throws**

**Las excepciones de  
tiempo de  
ejecución deben  
documentar como  
evitar su  
lanzamiento**

**Sean específicos  
con lo que atajan,  
no está permitido atajar  
Exception o RuntimeException**

# **La inicialización de los atributos va en el constructor**

# Las clases van en CamelloCase y sus atributos en DromedarioCase



**{ Los atributos `private`  
o  
`protected` con justificación  
}  
y *nunca* `public`**

**Los paquetes  
deben comenzar  
en  
ar.unrn e ir en  
minúsculas**

**Implementar  
equals requiere  
implementar  
hashCode**

**Al extender,  
sobreescribir solo  
para llamar a super  
no es correcto\*\***

*Excepto con el constructor*

0x002F

*Cuestiones de estilo*

# Minimizar el código duplicado

# **Las clases, atributos y métodos llevan documentación**

**Los métodos  
get / set no pueden  
ser usados para la  
lógica del  
problema**

**La utilización de  
atributos estáticos  
debe de estar  
justificada**



**No hacer**  
**import paquete.\***  
**Solo traer lo que**  
**necesitamos**

# No apilen líneas

**Todos los bloques llevan sus  
llaves, y no encadenar más de  
~dos llamadas a métodos en  
una línea**

0x0000

*Cuestiones de estilo*

# Los nombres de las interfaces

terminan en dor,

able o ible

Comparador

Ordenable

{verbo-sustantivo /  
verbos-adjetivo}

0x0000

*Cuestiones de estilo*

**Al usar una  
Collection, la  
variable\* del tipo  
de la Interfaz**

```
List lista = new LinkedList()
```

**unrn.edu.ar**

**UNRN**

Universidad Nacional  
de **Río Negro**



| **unrionegro**

**Los argumentos y  
retornos deben ser  
del tipo más simple  
posible**

**Atajar solo para un  
print no es  
gestionar la  
excepción**



**Abran  
hilo**

**Podemos debatir todas y cada una de ellas**