



GUÍA PARA APROBAR

EL TP 1



HOLA!

Bienvenid@ a la guía de recomendaciones

Es importante leer esta guía **ANTES** y **DURANTE** la realización del trabajo práctico 1 de Algoritmos III



CONDICIONES DE REALIZACIÓN Y APROBACIÓN

Click en el ícono





EL INFORME

Empecemos a ver cómo es presentar un buen informe.
Es decir transmitir con **claridad** a otro desarrollador (en este caso un docente) cómo es la solución.

L^AT_EX



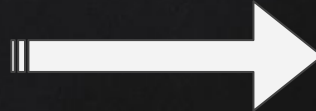
Plantilla

Overleaf

Preparamos una plantilla en LaTeX disponible en Overleaf para facilitar el armado del informe.

No es obligatorio su uso pero es una buena oportunidad para familiarizarse con LaTeX si todavía no lo usaron:

Click en el ícono



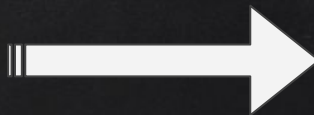


¿Qué diagramas tengo que presentar?
¿Cuántos?



ANTES DE SEGUIR LEER “UML SÍ O SÍ”

Click en el ícono





LA HERRAMIENTA

¡¡No usar el **PAINT** !!





LA HERRAMIENTA

Existen muchas herramientas diseñadas para realizar diagramas UML con la notación correcta. Bajar alguna y utilizarla.



Plant UML



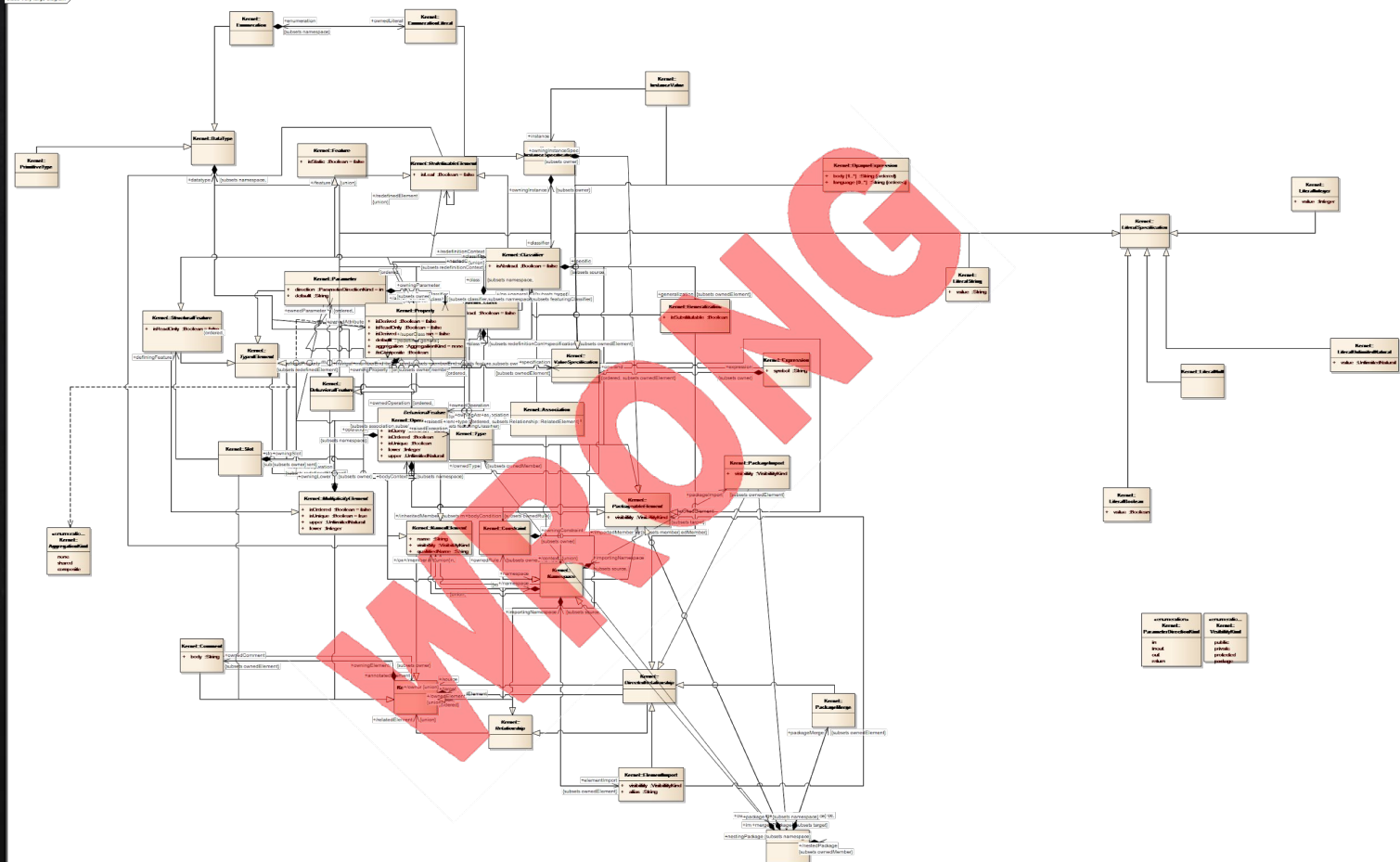


DIAGRAMA DE CLASES



En ingeniería de software, un diagrama de clases en Lenguaje Unificado de Modelado (UML) es un tipo de diagrama de estructura **estática** que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos.

class VeryLargeDiagram





DIAGRAMAS DE CLASES

- ✗ **No** intentar expresar la solución incluyendo TODAS las clases en un **único** y gran diagrama. Separlo. Presentar varios diagramas de clases que muestren distintas “partes” de mi solución de ser necesario.
- ✗ Tampoco conviene que en los diagrama de clases estén absolutamente tooodos los atributos y tooodos los métodos de nuestras clases. Que sea representativo de lo más importante, de lo que quiero comunicar con ese diagrama.
- ✗ Deben ser claros: Evitar que se crucen muchas líneas, debería poder leerse a simple vista sin tener que hacer “zoom”.



DIAGRAMA DE SECUENCIA



El diagrama de secuencia es un tipo de diagrama usado para modelar **interacción** entre objetos en un sistema según UML. En inglés se pueden encontrar como "sequence diagram", "event-trace diagrams".



DIAGRAMAS DE SECUENCIAS

- ✗ No entregar solamente 1. Hay muchos escenarios posibles donde mostrar la interacción entre los objetos de su solución. En particular **cada** test entregado por la cátedra es un buen candidato a ser mostrado en un diagrama de secuencia.
- ✗ No mostrar un escenario trivial, un getter o setter.
- ✗ Seguro conviene también mostrar algún escenario relacionado con lo que expliquen en “detalles de implementación”



DÓNDE PONER LOS DIAGRAMAS

- ✗ Todos los diagramas deben estar embebidos en el informe.
- ✗ No entregar archivos sueltos de diagramas (ni jpg, ni png, ni asta)
- ✗ Cada diagrama debe entrar en el ancho de una hoja A4 con un tamaño de fuente legible (no debe ser necesario hacer zoom). Si no entra entonces el diagrama es demasiado grande y debe descomponerse en varios diagramas.



SUPUESTOS

Los supuestos tienen que explicar las decisiones que han tomado al resolver el trabajo práctico cuando la consigna no haya sido explícita y clara y llevaron a que ustedes tengan que determinar cómo se tiene que comportar la aplicación.

Deberían poder ser leídos como una extensión del enunciado **sin decir detalles de implementación.**

Por ejemplo, para un trabajo práctico del juego Gálaga: “*Al llegar al borde de la pantalla y querer seguir avanzando, la nave del jugador rebota en el sentido opuesto*”



DETALLES DE IMPLEMENTACIÓN

Aquí deben detallar y explicar cómo han decidido resolver los aspectos más complejos, más picantes del trabajo práctico. Eso quedará a criterio del alumno, pero siempre en todo trabajo práctico se destacan situaciones más complejas que requieren de explicaciones detalladas.



EXCEPCIONES

- Deben listar las excepciones que hayan creado
- Explicar por qué las han creado
- Qué significado tienen
- Y dónde se utilizan
- Debe haber pruebas unitarias para cada una de ellas

3.

El Código

```
<td align="center" style="font-size: 20px; line-height: 20px;">&nbsp;</td></tr>  
<tbody><tr>  
  <td align="left" style="color: #FFFFFF; font-size: 12px; font-weight: bold; line-height: 22px; letter-spacing: 1px;" class="title_color">  
    <!-- ===== section text ===== -->  
    <div style="line-height: 22px;">
```



RECOMENDACIONES DEL CÓDIGO

- Que la solución haga pasar las pruebas del enunciado es **condición necesaria pero no suficiente** para la aprobación.
- El objetivo es que la solución sea lo más **orientada a objetos** posible aplicando los temas vistos en la materia hasta ahora.
- La clase AlgoFix debe tener pocas responsabilidades. Se espera que que le **delegue responsabilidades** a las entidades adecuadas.
- Las clases creadas no deben ser simples contenedores de datos. Es decir, no deben tener solamente getters y setters, sino alguna responsabilidad. Hacer que tenga getters y setters únicamente implica que se viola el encapsulamiento en alguna parte (“**Tell, don’t ask**”).



RECOMENDACIONES DEL CÓDIGO

- Si se usa una relación de **herencia** entre dos clases debe estar debidamente justificada. **La herencia es una relación muy fuerte** y para usarla se debe cumplir la regla del “**es un**” y además debe haber código que tenga sentido pasarlo a una clase padre. Al ser Smalltalk un lenguaje de tipado dinámico se puede hacer polimorfismo sin herencia, por lo que se debe ser más cuidadoso todavía con el uso de herencia.
- Chequear que no haya código repetido.
- Chequear la responsabilidad asignada a cada objeto. Si un objeto tiene muchas responsabilidades seguro se podrá separar en varios otros.



RECOMENDACIONES DEL CÓDIGO

- No reinventar la rueda, investigar mensajes propios de los objetos que provee Pharo para evitar implementar cosas ya realizadas y probadas.
- **Evitar el uso de `ifTrue/ifFalse`**. Tratar de encontrar una solución polimórfica antes que estructuras de control.
- Medir la **cobertura de las pruebas** con el Test Runner. Si realizan TDD de manera estricta la cobertura será del 100% pero, con cierto criterio, alcanza con un 80% por lo menos.
- Una cobertura del 100% tampoco garantiza que las pruebas estén bien. Es necesario que sean unitarias y hayan sido creadas de manera incremental.



MANEJO DE FECHAS EN PHARO

Se pueden usar la clase `DateAndTime` de Pharo para manejar las fechas:

```
| fechaPublicacionTP1 fechaEntregaTP1 tiempoResolucionTP1 |
```

```
fechaPublicacionTP1 := DateAndTime year: 2018 month: 9 day: 16 hour: 22 minute: 0.
```

```
fechaEntregaTP1 := DateAndTime year: 2018 month: 10 day: 1 hour: 15 minute: 55.
```

```
fechaEntregaTP1 := '2018-10-01T00:00:00-03:00' asDateAndTime.
```

```
tiempoResolucionTP1 := (fechaEntregaTP1 - fechaPublicacionTP1).
```

```
Transcript show: tiempoResolucionTP1; cr.      " Duracion total "
```

```
Transcript show: (tiempoResolucionTP1 days). " Duracion en dias "
```

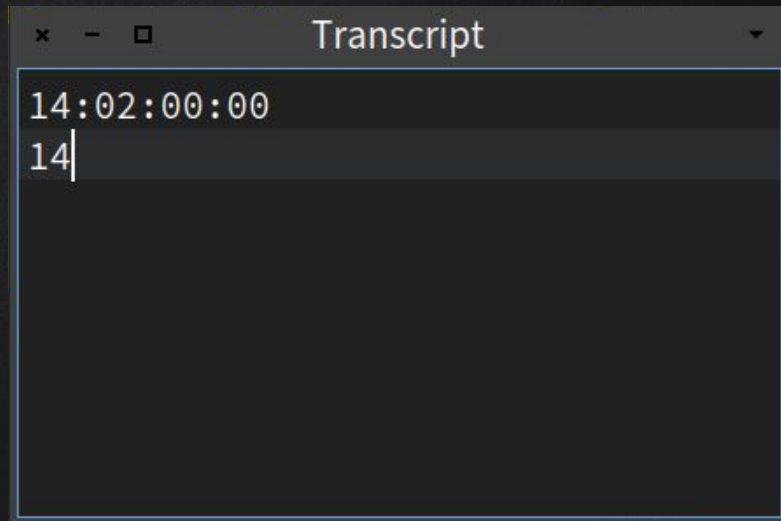


MANEJO DE FECHAS EN PHARO

El código anterior muestra lo siguiente en el Transcript:

En el primer caso se muestra la duración total expresada en DÍAS-HORAS-MINUTOS-SEGUNDOS y en el segundo caso se muestran solamente los días.

Análogamente se pueden mostrar solamente las horas con el mensaje hours.



```
14:02:00:00
14|
```



MANEJO DE FECHAS EN PHARO

Para poder sumar y restar fechas teniendo en cuenta los cambios de meses y de año se puede usar la clase `Duration`:

```
| fecha1 fecha2 fecha3 |  
fecha1 := '2018-10-01T00:00:00-03:00' asDateAndTime.
```

```
fecha2 := fecha1 + (Duration days: 7).  
fecha3 := fecha2 + (Duration month: 1).
```

```
Transcript show: fecha1;cr.  
Transcript show: fecha2;cr.  
Transcript show: fecha3.
```

A screenshot of a 'Transcript' window with a title bar containing standard window controls (close, minimize, maximize) and a dropdown arrow. The window displays the output of the code execution, showing three ISO 8601 timestamps on separate lines: '2018-10-01T00:00:00-03:00', '2018-10-08T00:00:00-03:00', and '2018-11-08T00:00:00-03:00'. The text is white on a dark background.

```
2018-10-01T00:00:00-03:00  
2018-10-08T00:00:00-03:00  
2018-11-08T00:00:00-03:00
```




!!! Y RECUERDEN !!!



UN GRAN PODER

CONLLEVA UNA GRAN RESPONSABILIDAD



CONSULTAS

- Utilizar la opción de búsqueda del campus para ver si su duda no fue evacuada anteriormente.
- Consultas generales en el campos o en Slack → #consultas-tp1 (no compartir soluciones)
- Consultas privadas → @Ayudante Virtual



PERO ADEMÁS

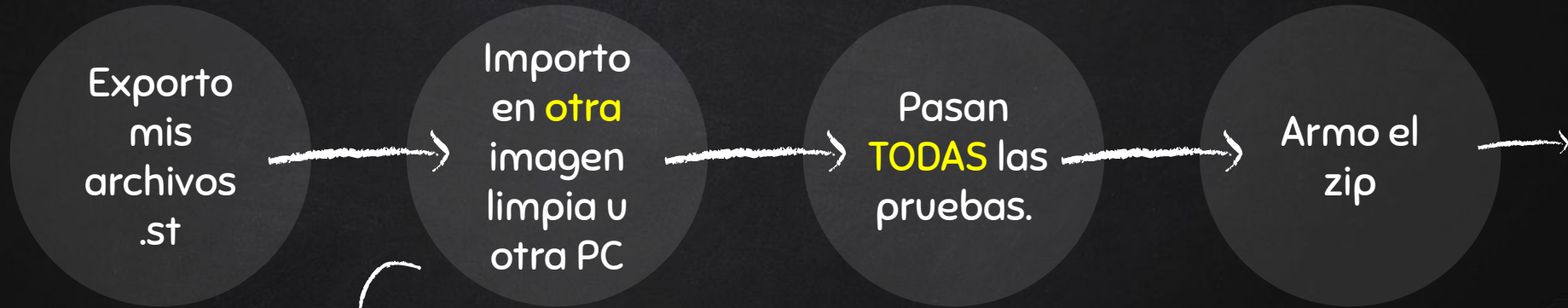
Una entrega es válida si posee:

- Código de la solución que haga pasar las pruebas del enunciado
- Código de las pruebas
- Informe con por lo menos un diagrama de clases y por lo menos dos diagramas de secuencia.

Si **no** contiene cada uno de esos elementos => se **desaprueba** automáticamente sin corregir.



ANTES DE SUBIR EL ZIP AL CAMPUS 1/3



Desde acá es lo que hacen los docentes al corregir su TP. ¿Por qué no hacer lo mismo nosotros antes de entregar?



ANTES DE SUBIR EL ZIP AL CAMPUS 2/3

→ Me fijo
que el zip
tenga
todos los
archivos

→ Me fijo
que el zip
se pueda
“unzip”

→ Me fijo
que el zip
se pueda
“unzip” en
otra PC

→ Me fijo
que el zip
se pueda
“unzip” en
otro SO →



ANTES DE SUBIR EL ZIP AL CAMPUS 3/3



¡ Entrego
el zip !



GRACIAS!

