



Indice

[1 Music Guide](#)

[1.1 Entrega 0 - Para el curso de los miércoles a la noche](#)

[1.2 Conceptos a aplicar](#)

[1.3 Material de consulta](#)

[1.4 Casos de prueba a implementar](#)

[1.5 IMPORTANTE: Herramientas de versionado](#)

[1.5.1 Cómo crear un usuario en github](#)

[1.5.2 Qué servicios da github](#)

[1.5.3 Herramienta Classroom](#)

[1.5.4 Cómo unirse al Classroom](#)

[1.5.5 Me equivoqué de grupo y no me deja cambiar, ¿qué hago?](#)

[1.5.6 Objetivos que deben cumplir](#)

1 Music Guide

1.1 Entrega 0 - Para el curso de los miércoles a la noche

Estamos por cerrar un interesante proyecto para músicos. Mientras nuestro equipo de ventas termina de ajustar los detalles, necesitamos que el grupo técnico vaya calentando motores, para lo cual se pide:

- Modelar un objeto que represente al estadio Luna Park
 - su capacidad es de 9.290 personas para cualquier día
- Modelar un objeto que represente al local "La Trastienda"
 - su capacidad es de 400 en planta baja y 300 en el primer piso (el primer piso se habilita los sábados únicamente).

1.2 Conceptos a aplicar

En esta entrega queremos que apliques estos conceptos

- definición de objetos conocidos (wko)
- polimorfismo
- manejo de referencias
- definición de casos de prueba o tests simples

Y tangencialmente queremos que aprendas a trabajar en grupo utilizando herramientas profesionales

- para compartir código
- para manejar versionado de código
- y eventualmente para coordinar el trabajo en conjunto de diferentes personas que colaboran en un proyecto de software

1.3 Material de consulta

Recomendamos leer estos apuntes

- [Módulo 1](#): Objeto. Mensaje. Métodos.
- [Módulo 2](#): Referencias. Estado. Compartir objetos. Identidad.
- [Anexo A](#): Testing
- [Anexo D](#): Tutorial de Git en Wollok.

1.4 Casos de prueba a implementar

Test (con precondiciones)	Resultado esperado
Queremos saber la capacidad de La Trastienda para el 05 de agosto del 2017	700 (porque es sábado)
Queremos saber la capacidad de La Trastienda para el 08 de agosto del 2017	400 (porque no es sábado)
Queremos saber la capacidad del Luna Park para el 05 de agosto del 2017	9.290
Queremos saber la capacidad del Luna Park para el 08 de agosto del 2017	9.290

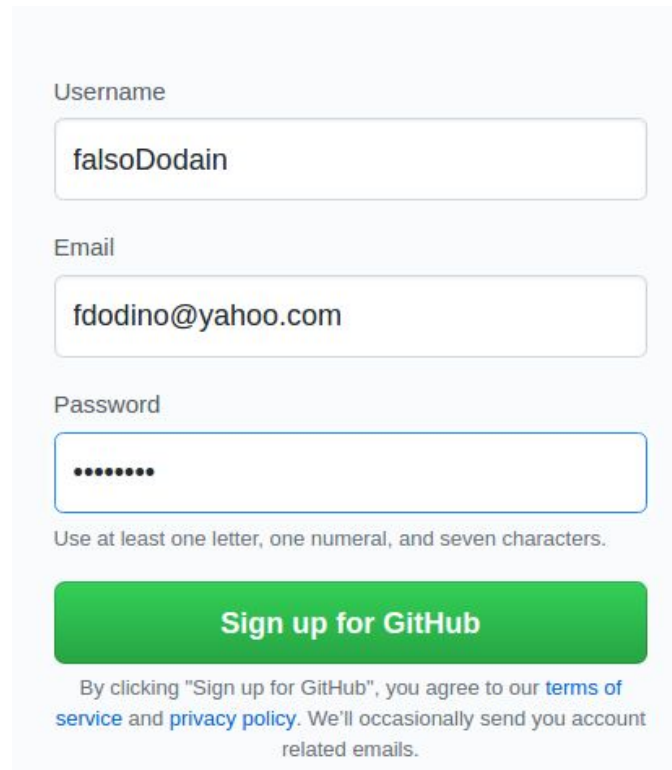
1.5 IMPORTANTE: Herramientas de versionado

Para compartir código vamos a utilizar la infraestructura que provee github, en particular

- cada persona (alumno o tutor) debe tener un usuario de github
- cada grupo debe tener un repositorio en github

1.5.1 Cómo crear un usuario en github

- Ingresar a <https://github.com/>
 - crear un nombre de usuario (en lo posible evitar nombres raros, como pepeElGroso o pepeKpo, ya que es muy posible que este mismo usuario sea usado posteriormente en el ámbito profesional).
 - una dirección de mail
 - una contraseña (revisen las condiciones que debe cumplir)

A screenshot of the GitHub sign-up form. It features three input fields: 'Username' with the text 'falsoDodain', 'Email' with 'fdodino@yahoo.com', and 'Password' with seven dots. Below the password field is a note: 'Use at least one letter, one numeral, and seven characters.' A green button labeled 'Sign up for GitHub' is positioned below the form. At the bottom, a line of text states: 'By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We'll occasionally send you account related emails.'

1.5.2 Qué servicios da github

- Infinitos repositorios (públicos) por usuario, donde ustedes pueden subir proyectos propios. También tiene disponible un [student pack](#) donde brinda servicios y herramientas usualmente pagas de forma gratuita para estudiantes.
- Con su usuario ustedes pueden unirse a repositorios públicos o a otros repositorios privados, o colaborar en repositorios de terceros (donde el aporte deberá ser aprobado por el dueño del repositorio).
- Los repositorios pueden pertenecer
 - a un usuario
 - o bien a una organización (por ejemplo, a esta materia o a un curso de esta materia)
- Cada repositorio tiene muchas funcionalidades: permite almacenar el historial de código y visualizarlo en la web, pero también se provee un issue tracker (para registrar bugs y nuevos features) y una wiki (para subir documentación online), entre muchas otras características.

Pueden ingresar a <https://github.com/wollok/pepita-game> y navegarlo para ver un ejemplo de un repositorio. Incluso el entorno Wollok es un proyecto de código abierto, que pueden ver en <https://github.com/uqbar-project/wollok>

1.5.3 Herramienta Classroom

Github ofrece una herramienta educativa que facilita todos los procesos burocráticos de un curso, en particular

- define una organización común para un curso (ej: curso miércoles a la noche de Paradigmas de Programación)
- descentraliza la creación de los repositorios por grupo, ya que
 - el primero que ingresa (alumno o tutor) crea el grupo
 - los siguientes se suman al grupo
 - el único tema es cuando se suscriben erróneamente a otro grupo, pero eso lo manejaremos internamente (no es lo mismo hacer esto para los 50 casos que para un 10% que se equivoque).
- y por último permite crear asignaciones o tareas, para que se agrupen de distinta manera. En nuestro caso vamos a tener una única asignación: "TP Objetos" hasta el final de la cursada, ya que tendremos grupos estables durante todo el ciclo de vida del trabajo práctico. Los cambios serán excepcionales y los manejaremos como tales (en forma manual).

1.5.4 Cómo unirse al Classroom

- Se fijan en la planilla cuál es su número de grupo
- Ingresan a este [link](#)
- Deberán loguearse a github con el usuario creado en el paso 1.3.1
- Si no existe un grupo con su número, lo crean con el formato "Grupo n"
- Por ejemplo:


Create a new team

Grupo 1

+ Create team

-
- Si algunos de sus compañeros de grupo ya lo creó, simplemente tienen que unirse a ese:

Join an existing team

Grupo 1 1 student	Join
	

Una vez que hagan eso les va a aparecer el link al repositorio de su grupo. Cada tutor se unirá al grupo automáticamente, no es necesario hacer ninguna tarea administrativa más.

1.5.5 Me equivoqué de grupo y no me deja cambiar, ¿qué hago?

Mandale un mail a tus tutores, y ellos lo arreglarán.

1.5.6 Objetivos que deben cumplir

1. Clonarse el repositorio en una máquina diferente por integrante (clone)
2. Cada integrante debe subir un cambio al repositorio remoto (checkout - pull - commit - push). La recomendación es que cada uno modele un objeto diferente en un archivo distinto.
3. Los integrantes deben crear un conflicto con el versionador y resolverlo satisfactoriamente. Nuestra recomendación es
 - a. En un momento t_1 uno de los integrantes hace un commit + push de su cambio. Como consejo, pueden crear un test específico para este punto

```
test "resolver el conflicto en git" {  
  assert.that(true)  
}
```

- b. En otra máquina, el otro de los integrantes baja el cambio (pull).
- c. En un momento t_2 , uno de los integrantes agrega un comentario en dicho test:

```
test "resolver el conflicto en git" {  
  // hecho por XXX (integrante XXX)  
  assert.that(true)  
}
```

- d. Luego lo sube al repositorio (commit + push)
- e. En un momento t_3 , el otro integrante agrega su comentario en la versión (desactualizada) del test conflictivo:

```
test "resolver el conflicto en git" {  
  // hecho por YYY (integrante YYY)  
  assert.that(true)  
}
```

- f. Cuando quiere hacer un commit + push le aparece un conflicto. Se debe resolver el conflicto de manera de tener ambos comentarios (no deben perderse los cambios de ninguno de los integrantes, pueden hacerlo por línea de comando o bien con el plugin eGit como se muestra en el [Anexo D](#)).

```
test "resolver el conflicto en git" {  
  // hecho por XXX (integrante XXX)  
  // hecho por YYY (integrante YYY)  
  assert.that(true)  
}
```

- 4. **Además los integrantes deben implementar los objetos que pide la entrega y los casos de prueba asociados.**
- 5. Al finalizar la entrega se debe generar un tag "Entrega 0" en el repositorio del grupo.

IMPORTANTE: Deben evitar hacer push -f al subir sus cambios, eso pisa cualquier cambio de sus compañeros que no hayan sincronizado previamente (en un equipo de desarrolladores, hacer eso implica traer facturas).