

Trabajo Práctico Especial

Objetivo

El objetivo del presente trabajo es implementar un juego en el lenguaje Java, con interfaz visual, aplicando los conceptos sobre diseño orientado a objetos adquiridos en la materia.

Descripción funcional

La aplicación a implementar es una variante del juego *Desktop Dungeons*¹. El juego consiste en un tablero por el cual el jugador se puede desplazar, donde hay enemigos con los que éste puede enfrentarse. Tanto el jugador como los enemigos tienen:

- **Salud:** es un número que determina la salud del personaje. Un personaje tiene una cantidad máxima de salud que es con la que comienza el juego. Durante el transcurso del mismo, esta va disminuyendo a medida que se enfrenta con otros personajes. Si este valor llega a 0, el personaje muere.
- **Fuerza:** es un número que determina cuánto daño le produce a sus oponentes en un ataque. El oponente pierde tantas unidades de salud como fuerza tiene el personaje que lo ataca.
- **Nivel:** es un número entre 1 y 3 que da una medida de cuán bueno es un personaje. El nivel determina la cantidad máxima de salud y la fuerza que tiene el mismo. Por ejemplo un jugador de nivel 1, tiene una salud máxima de 10 unidades y una fuerza de 5 unidades.
- **Experiencia:** es un número que registra el grado de evolución del jugador en su nivel actual.

El objetivo del juego es vencer al único enemigo de nivel 3 del tablero. El jugador inicialmente comienza con 10 unidades de salud (sobre un máximo también de 10) y 5 unidades de fuerza. Dado que estos atributos no le permiten vencer al enemigo de nivel 3, deberá aumentar su nivel para mejorarlos, y así poder vencerlo. Para ello, el jugador debe enfrentarse con enemigos más débiles a los que sí pueda vencer.

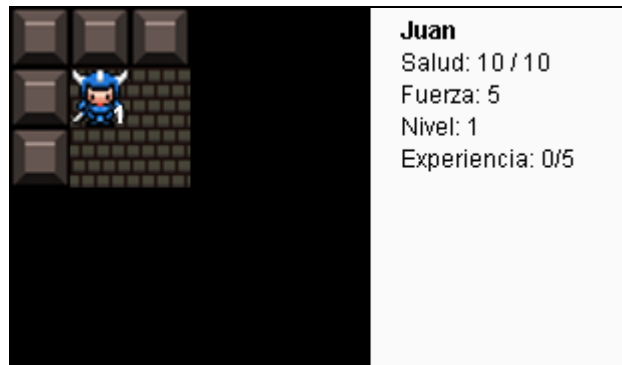
Inicialmente el tablero comienza mostrando únicamente la celda donde está ubicado el jugador y las ocho aledañas. Las restantes celdas del tablero se van descubriendo a medida que el jugador se desplaza por el mismo, siempre mostrando las ocho celdas que lo rodean.

El tablero puede contar con paredes que son elementos que impiden el paso del jugador. Si bien el tablero no necesariamente está rodeado por paredes, no se debe permitir que el jugador salga del mismo.

A la derecha del tablero se muestra el estado del jugador que consta de: nombre, cantidad actual y máxima de unidades de salud, fuerza, nivel y experiencia acumulada sobre la requerida para aumentar su nivel. Asimismo, sobre la imagen del jugador se muestra el nivel actual que tiene.

En la siguiente imagen se muestra el estado inicial de un tablero de 6 x 6 en donde se pueden observar los atributos del jugador y las celdas descubiertas:

¹ <http://www.qcfdesign.com/?cat=20>



Enfrentamientos

Para poder vencer a un enemigo es necesario enfrentarse a él. Se produce un enfrentamiento cuando el jugador se encuentra ubicado en una celda adyacente a la de un enemigo e intenta moverse hacia la misma. Cuando esto ocurre, el enemigo ataca en primer lugar al jugador, restándole tantas unidades de salud como fuerza él tenga. Si el jugador sobrevive, ataca al oponente de la misma manera (restándole tantas unidades de salud como fuerza tenga). En caso de que se agote la salud del enemigo, este muere y es reemplazado en el tablero por una mancha de sangre. Además se incrementa la experiencia del jugador tantas unidades como nivel tenga el oponente vencido. Notar que para que el jugador venza al enemigo puede que sea necesario más de un enfrentamiento.

Cuando el jugador se ubica en una celda adyacente a la de un enemigo (y por lo tanto está en condiciones de enfrentarse con él), se deben mostrar las características del enemigo (tipo, salud, fuerza y nivel). Esto también debe ocurrir cuando el usuario posiciona el mouse sobre algún enemigo que se encuentre visible y vivo.

La salud del jugador se recupera cuando este explora celdas aún no descubiertas del tablero. Por cada celda descubierta, se incrementa la salud del jugador y la de todos los enemigos vivos tantas unidades como nivel tenga cada uno.

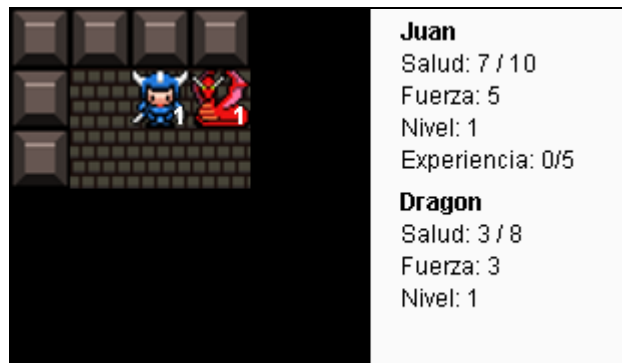
En el tablero de ejemplo de la sección anterior, como el jugador no se encuentra en una celda aledaña a un oponente no es posible que se produzca un enfrentamiento, y por lo tanto sólo se muestra su información a la derecha del tablero. Asimismo, sobre la imagen del jugador se muestra su nivel.

Si el usuario presiona la flecha derecha se descubren las 3 celdas aledañas, una de las cuales contiene un enemigo de nivel 1:



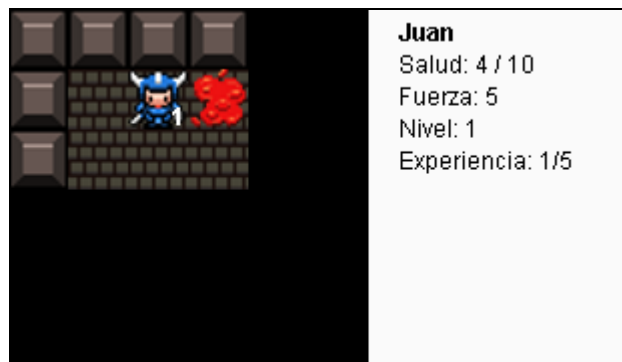
Como el jugador se encuentra en una celda vecina a un oponente, se muestran las propiedades del mismo en la barra de estado. Asimismo, sobre la imagen del enemigo se muestra su nivel.

Si el usuario presiona nuevamente la flecha derecha se produce un enfrentamiento. El enemigo ataca primero al jugador, reduciéndole su salud en 3 unidades. Luego, como el jugador permanece con vida, ataca al oponente quitándole 5 unidades de salud.

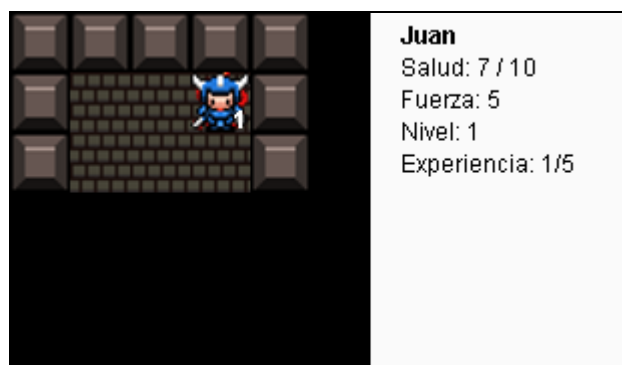


Para que el jugador venza al enemigo será necesario que peleen nuevamente.

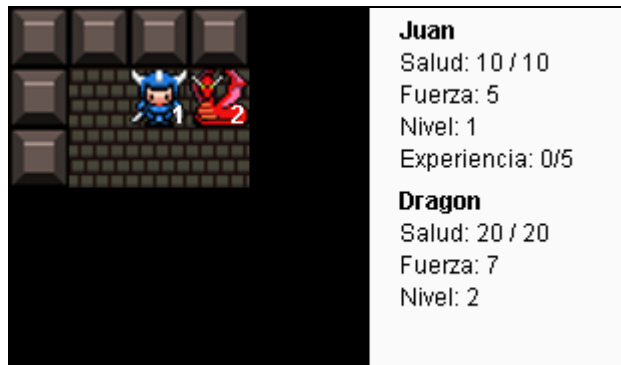
Si se vuelve a presionar la flecha derecha, se produce un nuevo enfrentamiento. En primer lugar, el jugador pierde 3 unidades de salud. Luego, al atacar al oponente lo vence debido a que su salud restante es menor a la fuerza del jugador. Cuando esto ocurre, acumula una unidad de experiencia (explicado más adelante en la sección “Niveles”). La imagen del enemigo se reemplaza por una mancha de sangre y a la derecha del tablero no se muestra más la información del enemigo.



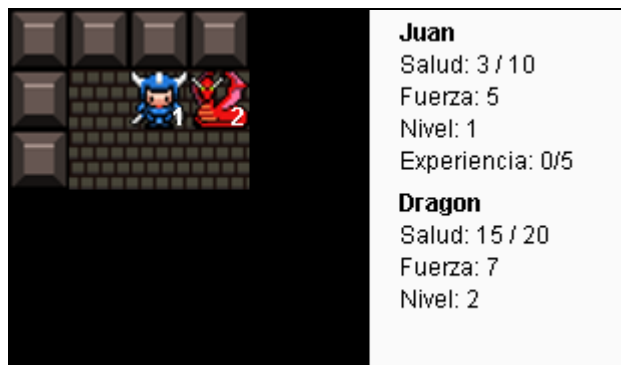
Luego de vencer al oponente, si el jugador presiona la flecha derecha, avanza una celda en dicha dirección. Cuando ocurre esto, se descubren 3 nuevas celdas y por lo tanto recupera 3 unidades de salud.



En este otro ejemplo, el jugador se encuentra en una celda vecina a la de un enemigo de nivel 2. Junto al tablero, se muestra la información del jugador y del oponente.



Cuando el jugador se enfrenta con el enemigo, pierde 7 unidades de salud. Asimismo el enemigo pierde otras 5.



En caso de que se produzca un nuevo enfrentamiento, el jugador muere debido a que la fuerza del enemigo es superior su salud actual. Notar que la salud del enemigo no se ve disminuida luego del enfrentamiento ya que el jugador muere antes de poder atacarlo.

Cuando el jugador muere, la imagen del mismo es reemplazada por una mancha de sangre.



Niveles

El nivel de un personaje determina su salud máxima y su fuerza. Para el jugador que se encuentra en el nivel N , su salud máxima es $10*N$ y su fuerza $5*N$ (más los bonus levantados que se explican en la sección "Bonus").

Para el caso de los enemigos estos valores dependen tanto de su nivel como de su tipo. Existen 3 tipos de oponentes: dragón, golem y serpiente.

La salud máxima y la fuerza de un enemigo de nivel N están dadas por:

Salud máxima: $\text{Piso} \{ [(N + 3)^2 - 10] * S \}$

Fuerza: $\text{Piso} [(N^2 + 5 * N) * 0.5 * F]$

donde F y S son constantes que dependen del tipo de enemigo según la siguiente tabla:

| Enemigo | F | S |
|-----------|-----|------|
| Dragón | 1 | 1.35 |
| Golem | 0.7 | 1 |
| Serpiente | 1 | 1 |

El nivel de los oponentes no cambia durante el desarrollo del juego. A medida que el jugador gana enfrentamientos, acumula unidades de experiencia que le permiten aumentar su nivel. La cantidad de experiencia que acumula está dada por el nivel del enemigo que vence. Cuando un jugador de nivel N acumula $5*N$ unidades de experiencia incrementa su nivel (aumentando 10 unidades su salud máxima y 5 su fuerza). Cuando esto ocurre, la cantidad de experiencia vuelve a 0 pero la salud se mantiene en el valor actual.

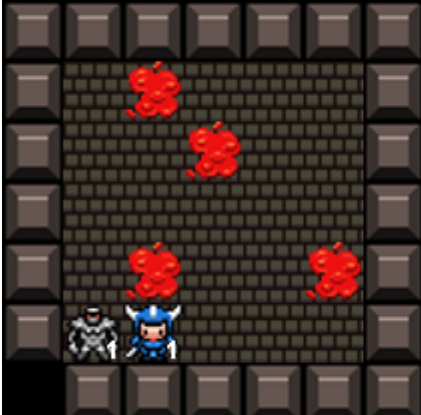
En el siguiente ejemplo, el jugador se encuentra en una celda vecina a la de un enemigo, luego de haber vencido a 4 oponentes de nivel 1. Por este motivo, su experiencia actual es de 4 unidades.



Juan
Salud: 10 / 10
Fuerza: 5
Nivel: 1
Experiencia: 4/5

Golem
Salud: 6 / 6
Fuerza: 2
Nivel: 1

Si el jugador se enfrenta con el golem, perderá 2 unidades de salud y le quitará 5 al enemigo.

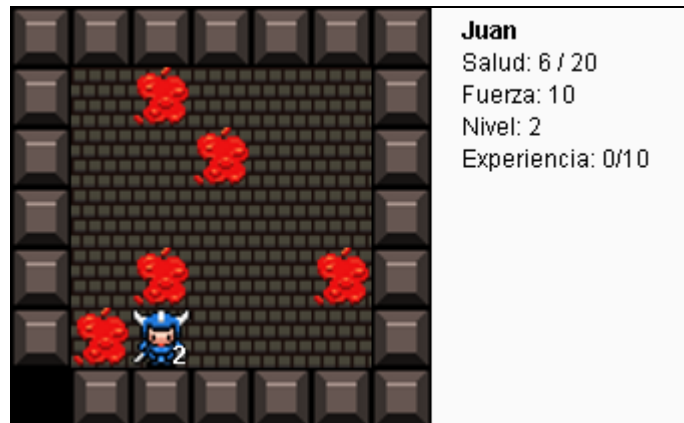


Juan
Salud: 8 / 10
Fuerza: 5
Nivel: 1
Experiencia: 4/5

Golem
Salud: 1 / 6
Fuerza: 2
Nivel: 1

Si se produce un nuevo enfrentamiento, luego de que el golem ataque al enemigo quitándole 2 unidades de salud, el jugador lo vencerá ya que su fuerza es superior al la salud actual del oponente. Cuando esto ocurre, acumula 1 unidad adicional de experiencia. Como alcanza el valor máximo, aumenta su nivel. En consecuencia, se incrementa la salud máxima en 10

unidades, la fuerza 5 unidades, y se establece la experiencia en 0. Asimismo, se actualiza la imagen del jugador para mostrar su nuevo nivel.

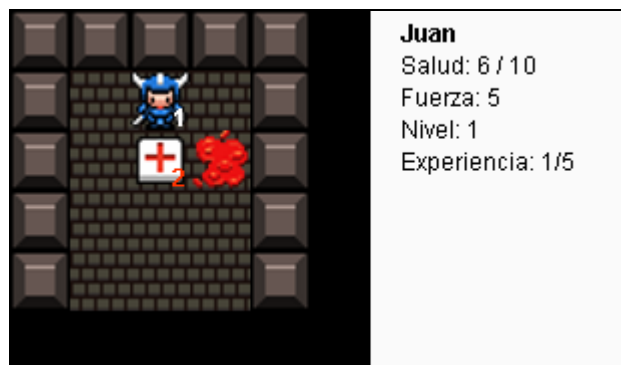


Notar que no es posible ganar el nivel anterior ya que no hay un enemigo de nivel 3 (por una falla de diseño del tablero).

Bonus

El tablero puede contener objetos que otorgan dos tipos de bonus: salud y fuerza.

El bonus de salud restaura tantas unidades de salud como éste lo indique. En el siguiente ejemplo, el jugador tiene 6 unidades de salud y se encuentra en una celda adyacente a un bonus de 2 unidades de salud:



Al desplazarse hacia abajo, levanta el bonus e incrementa 2 unidades su salud. El bonus desaparece del tablero.

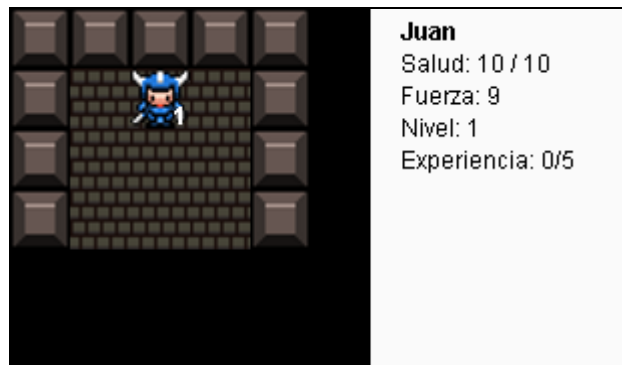


Si al sumar el valor de bonus con la salud actual del jugador se excede del máximo, se establece la salud en el máximo valor posible. Si el jugador ya posee la máxima salud posible, consume el bonus de todos modos (sin modificar su salud).

Existe otro tipo de bonus que incrementa la fuerza tantas unidades como se indica en la figura. En el siguiente ejemplo el jugador se encuentra en una celda adyacente a un bonus de fuerza 4.



Al desplazarse hacia la derecha, el jugador consume el bonus de fuerza incrementando así esta propiedad en 4 unidades.



Desarrollo del juego

Al ingresar a la aplicación, se le debe mostrar al usuario un menú principal con las siguientes opciones:

1. **Nuevo juego**
2. **Cargar juego**
3. **Salir**

Al seleccionar la opción “*Nuevo juego*” se comienza un nuevo juego. El jugador debe seleccionar de una lista de tableros, aquel que desea jugar. Los tableros se almacenan en archivos de texto con extensión “*.board*”. Estos archivos deben estar almacenados en un directorio llamado **boards**, que debe existir en el *path* actual donde se está ejecutando la aplicación.

Antes de mostrarle al usuario el tablero seleccionado el usuario debe ingresar su nombre. Luego, comienza el juego. El nombre elegido por el usuario se debe mostrar a la derecha del tablero. Una vez completado el tablero (al vencer al enemigo de nivel 3), se debe volver al menú principal.

Durante el transcurso del juego, el usuario debe poder reiniciarlo a través de un botón u opción del menú, así como cancelarlo y volver al menú principal. Asimismo, debe poder guardar el estado actual del tablero.

Formato de los archivos de tableros

El archivo del tablero es un archivo de texto que posee el siguiente formato:

- En la primera línea se especifican las dimensiones del tablero, en formato “filas, columnas”. El tablero mínimo es de 6 x 6 y el máximo de 30 x 30.
- En la segunda línea se especifica el nombre del tablero (que se muestra en el listado de tableros).
- Luego, por cada elemento del tablero se agrega una línea indicando su contenido. Estas líneas contienen 6 números enteros separados por coma que representan los siguientes campos:
 - **tipo de celda:** un entero entre 1 y 5 que indica el tipo de elemento del juego según el siguiente listado:
 1. Jugador
 2. Pared
 3. Enemigo
 4. Bonus de salud
 5. Bonus de fuerza
 - **fila, columna:** indica la fila y la columna de la celda.
 - **tipo de enemigo:** es un número entero que indica el tipo de enemigo según el siguiente listado:
 1. Golem
 2. Dragón
 3. Serpiente
 - Este valor solo se debe indicar para celdas de tipo 3. Para los restantes tipos de celdas debe valer 0.
 - **nivel del enemigo:** para el tipo de celda 3, indica el nivel del enemigo. Para los restantes elementos del tablero, este valor debe ser 0.
 - **valor:** para los tipos de celda 4 y 5, indica la cantidad de unidades de vida que restaura y la fuerza que aumenta respectivamente. Para los restantes elementos del tablero, este valor debe ser 0.

Adicionalmente, el archivo puede tener líneas de comentarios. Un comentario comienza con el símbolo “#” y dura hasta el final de la línea. Una línea puede ser todo un comentario, o se puede agregar un comentario al final de una línea válida. Las líneas en blanco, los espacios y las tabulaciones deben ser ignorados.

La aplicación debe validar el archivo del tablero. Ante un error de formato o de validación del tablero se debe indicar al usuario que el archivo es inválido, y luego salir de la aplicación.

A continuación se muestra un ejemplo de un archivo de tablero:

```
10, 10
ejemplotablero

#Paredes
2,0,0,0,0,0
2,0,1,0,0,0
2,0,2,0,0,0
2,0,3,0,0,0
2,0,4,0,0,0
2,0,5,0,0,0
2,0,6,0,0,0
```



```

2,0,7,0,0,0
2,0,8,0,0,0
2,0,9,0,0,0

2,9,0,0,0,0
2,9,1,0,0,0
2,9,2,0,0,0
2,9,3,0,0,0
2,9,4,0,0,0
2,9,5,0,0,0
2,9,6,0,0,0
2,9,7,0,0,0
2,9,8,0,0,0
2,9,9,0,0,0

2,1,0,0,0,0
2,2,0,0,0,0
2,3,0,0,0,0
2,4,0,0,0,0
2,5,0,0,0,0
2,6,0,0,0,0
2,7,0,0,0,0
2,8,0,0,0,0

2,1,9,0,0,0
2,2,9,0,0,0
2,3,9,0,0,0
2,4,9,0,0,0
2,5,9,0,0,0
2,6,9,0,0,0
2,7,9,0,0,0
2,8,9,0,0,0

2,1,6,0,0,0
2,2,6,0,0,0
2,3,6,0,0,0
2,5,6,0,0,0
2,5,5,0,0,0
2,5,4,0,0,0
2,5,3,0,0,0
2,5,2,0,0,0
2,5,1,0,0,0

# Jugador
1, 3, 3, 0, 0, 0

# Enemigos
3,4,6,1,1,0
3,2,5,2,1,0
3,1,3,3,1,0
3,4,4,1,1,0
3,3,2,2,1,0

3,5,8,1,2,0
3,3,8,2,2,0
3,2,7,3,1,0
3,6,2,3,2,0
3,6,4,2,2,0
3,7,8,1,2,0
3,8,7,3,2,0

3,8,1,2,3,0    # enemigo de nivel 3

# Bonus
    5,1, 7,0,0,3
    4, 7, 1, 0,0,5

```

En la siguiente imagen se muestra la representación del archivo anterior con todas las celdas descubiertas. Si bien al comenzar el juego sólo deberían estar visibles las 8 celdas aledañas al jugador, en este ejemplo se muestra el tablero completo para mostrar la ubicación de todos los elementos indicados en el archivo:



Almacenamiento de partidas

En cualquier momento del juego, el usuario puede guardar la partida para luego continuar jugando en otro momento. Al seleccionar esta opción, se le debe consultar el nombre y la ubicación del archivo a guardar (puede tener cualquier nombre y cualquier extensión, pero debe ser almacenado fuera del directorio *boards*).

Luego, cuando el usuario restaura una partida, deberá seleccionar el archivo que previamente guardó. La aplicación deberá cargar el tablero especificado en el archivo y continuar el juego desde allí. El formato de este archivo es libre, cada grupo deberá decidir qué información almacenar y con qué formato.

Consideraciones de implementación

La cátedra proporcionará un conjunto de clases para simplificar las operaciones visuales relacionadas con el tablero y el manejo de imágenes. **La aplicación debe funcionar con estas clases, y las mismas no se pueden modificar.**

Adicionalmente se proporcionarán imágenes para los tipos de celdas. Cada grupo puede optar por utilizar estas imágenes o crear las suyas.

El código fuente debe contener comentarios en formato *Javadoc*.

Se deberán implementar casos de prueba de *JUnit* para verificar el correcto funcionamiento de las clases de *backend*. No es necesario testear clases de *frontend*.

Se debe proporcionar un *buildfile* de *Ant* en donde el target default genere el *jar* ejecutable de la aplicación (archivo que debe llamarse *tpe.jar*), la documentación (*Javadoc*) y ejecute los tests.

Grupos

El trabajo deberá ser realizado por grupos de hasta 3 (tres) alumnos. **Cada grupo deberá enviar un correo electrónico a la cátedra informado quiénes son los integrantes.**

Se habilitará un repositorio de SVN para cada grupo, que puede ser utilizado durante el desarrollo del trabajo. La entrega final del trabajo se realizará a través de este repositorio (se considerará el último commit antes de la fecha y hora de entrega).

Material a entregar

El código fuente de la aplicación se deberá entregar a través del repositorio de SVN, el cual como mínimo deberá contener los siguientes directorios y archivos:

- **src**: directorio con los códigos fuente
- **resources**: directorio con las imágenes utilizadas y cualquier otro archivo necesario (propiedades, configuración, etc.)
- **boards**: directorio de tableros con archivos de prueba utilizados durante la implementación
- **build.xml**: buildfile de Ant para poder construir el proyecto

El repositorio **no** debe contener archivos de configuración de Eclipse ni código fuente compilado (no entregar archivos **.class** ni **.jar**).

Adicionalmente, se debe almacenar en el repositorio un informe que contenga: diagrama UML de las clases de *backend*, explicación de la jerarquía diseñada, problemas encontrados durante el desarrollo y decisiones de diseño tomadas.

Fecha de entrega

La fecha de entrega es el día **jueves 9 de junio antes de las 18hs**. Si el trabajo no estuviera aprobado se cuenta con una instancia de recuperación.

Existe la posibilidad de entregar en fecha tardía, el jueves 16 de junio a las 18hs. En este caso se aplicará una penalización de 2 (dos) puntos en la nota final del trabajo y no se contará con una instancia de recuperación.

Evaluación

Para la evaluación se tendrá en cuenta la funcionalidad, el estilo del código, el diseño de las clases implementadas, los testeos de unidad implementados y el contenido del informe. Se debe obtener una nota mayor o igual a 4 (cuatro) para aprobar.

Si se entrega un trabajo en fecha tardía que está aprobado, pero luego de aplicar la penalización pasa a estar desaprobado, el trabajo queda finalmente desaprobado.

Consultas

Las consultas se realizarán en el horario del laboratorio, o mediante correo electrónico a la dirección **poo@it.itba.edu.ar**