

# Algoritmos y estructuras de datos

## Cátedra Ing. Schmidt

### Trabajo práctico 2: TATETI

Nombre del grupo: Timebomb

Fecha de entrega: 15/11/24

Integrantes:

- Abel Tomas Modesti Funes
  - Padrón: 111932, e-mail institucional: [amodesti@fi.uba.ar](mailto:amodesti@fi.uba.ar).
- Rafaela Pilar Arias
  - Padrón: 112272, e-mail institucional: [rparias@fi.uba.ar](mailto:rparias@fi.uba.ar).
- Tomás Bautista Conti
  - Padrón: 111760, e-mail institucional: [tconti@fi.uba.ar](mailto:tconti@fi.uba.ar).
- Joaquín Daniel Ejberowicz
  - Padrón: 110741, e-mail institucional: [jejberowicz@fi.uba.ar](mailto:jejberowicz@fi.uba.ar).
- Ezequiel Matías Gacitua
  - Padrón: 112234, e-mail institucional: [egacitua@fi.uba.ar](mailto:egacitua@fi.uba.ar).
- Lucas Schvarzbok
  - Padrón: 111309, e-mail institucional: [lschvarzbok@fi.uba.ar](mailto:lschvarzbok@fi.uba.ar).

## Índice

Cuestionario.....	2
Informe del trabajo realizado.....	5
Manual de usuario.....	6
Manual del programador.....	12

## Cuestionario

1. ¿Qué es un SVN ?
2. ¿Qué es una “Ruta absoluta” o una “Ruta relativa”?
3. ¿Qué es Git?

## Respuestas

1. Un SVN o Subversion es una herramienta de control de versiones open-source. Se basa en un repositorio central (conocido como Main o Master) y diferentes ramas donde los contribuidores pueden subir sus cambios sin afectar a la rama principal. Permite además ver versiones previas y hacer un seguimiento de cambios. SVN fue lanzado en los 2000, y es el programa en el que se basa Git.
2. La “Ruta absoluta” y la “Ruta relativa” son formas de navegar a través de archivos, pero se diferencian en cómo son interpretadas por una computadora: La absoluta parte desde el directorio raíz (root) hasta el archivo al que se hace referencia. Un ejemplo de esta sería C:\Users\user\Escritorio\Nueva carpeta (en Windows, Linux utiliza otro formato).

Por otro lado, la ruta relativa es una ruta a un archivo o directorio que es relativo al directorio actual. Solo especifica la ubicación del archivo con respecto al directorio actual, por lo que, a diferencia del absoluto, no comienzan en el directorio raíz y son más cortos. Un ejemplo sería si se desea acceder desde el escritorio a un archivo dentro de Nueva carpeta: Nueva Carpeta\archivoDeTexto.txt.

Uno de los beneficios de utilizar una ruta absoluta es que es muy útil para referirse a archivos o directorios que están en una ubicación específica en

muchos dispositivos. Además, no es ambigua. Permite brindar la locación exacta del archivo o directorio.

En cambio, la ruta relativa es muy usada cuando se trabaja en directorios específicos que no varían. Por esta razón no es exportable a otros contextos, ya que la ubicación podría modificarse y volverse una ruta inválida.

3. Git es un sistema de control de versiones distribuido open-source que se destaca por ser el estándar de control de versiones mundial. Se encuentra en casi todos los entornos de desarrollo y sistemas operativos actuales. Fue lanzado en el año 2005 y permite crear repositorios locales para trabajar de forma remota fácilmente y luego sincronizarlos con el servidor general. Cada vez que un programador confirma un cambio, Git actualiza esos archivos modificados y guarda una referencia a esa copia instantánea (con un *commit*). Esto significa que cada vez que se hace un *commit*, Git guarda una representación de cómo está el proyecto en ese momento. La utilización de Git en el ámbito de la programación permite navegar por la historia de un proyecto de manera fácil, viendo cada "instantánea".

Para usar Git correctamente, se debe sincronizar el código con el servidor antes de crear nuevas versiones para evitar trabajar sobre una versión desactualizada. Además, en caso de ser necesario, se puede volver a cualquier versión anterior.

## Bibliografía utilizada

*What is subversion? SVN explained.* (2023, noviembre 3). Perforce Software; Perforce. Recuperado de

<https://www.perforce.com/blog/vcs/what-svn#what-is-subversion-svn>.

*Control de versiones con Subversion, Revision 4849.* (s/f). Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato. Recuperado de <https://www.unc.edu.ar/sites/default/files/SVN-BOOK-GENERALIDADES.pdf>

Waqas, M. (2023, marzo 6). *Difference between absolute path and relative path in linux*: Linkedin.com. Recuperado de

<https://www.linkedin.com/pulse/difference-between-absolute-path-relative-linux-wa-gas-muazam#:~:text=Absolute%20paths%20always%20start%20with,relative%20to%20the%20current%20directory>

Pérez, A. (2021, marzo 31). *Ruta absoluta, todo lo que hay que saber*. OBS Business School. Recuperado de

<https://www.obsbusiness.school/blog/ruta-absoluta-todo-lo-que-hay-que-saber>

*Understanding relative file paths*. (s/f). Codingrooms.com; Coding Rooms.

Recuperado de <https://www.codingrooms.com/blog/file-paths>

*¿Qué es Git?* (s/f). Microsoft.com. Recuperado de

<https://learn.microsoft.com/es-es/devops/develop/git/what-is-git>

*Git - Fundamentos de Git*. (s/f). Git-scm.com. Recuperado de

<https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Fundamentos-de-Git>

Atlassian. (s/f). *What is Git*. Atlassian. Recuperado de

<https://www.atlassian.com/git/tutorials/what-is-git>

## **Informe del trabajo realizado**

Para comenzar a trabajar en equipo de manera clara y ordenada creamos una hoja de cálculos de Google. Luego, incluimos cada TDA a realizar junto con sus atributos y métodos, imitando el UML visto en clase. El trabajo se dividió en partes iguales y planteamos entre todos las ideas principales del TDA Tablero. Consideramos necesario comprender la estructura y el funcionamiento del mismo ya que es el TDA que engloba a los demás. Por ejemplo, es en el Tablero donde se ven reflejadas todas las jugadas, y donde se define si algún jugador ganó.

En varias ocasiones distintas hicimos llamadas, ya que era lo más sencillo y cómodo para trabajar en grupo. Además, creamos nuestro repositorio privado de Github y nos aseguramos de que todos supieran utilizar la herramienta correctamente, haciendo énfasis en la utilización de ramas.

A pesar de la división de tareas comentada anteriormente, no fuimos demasiado estrictos. En general, todos participamos en las diversas Clases del trabajo práctico. Por ejemplo, luego de que un compañero terminaba un método o una Clase, otra persona leía el código con el fin de ver que era comprensible y funcional. De ser necesario, incluso proponía (e implementaba) mejoras. En otras ocasiones se dio que una persona codificaba una parte de cierta Clase, y otro decidía continuar el desarrollo porque tenía tiempo libre en ese momento. Así, intentamos alcanzar un código limpio, claro y con documentación completa.

A nivel grupal enfatizamos la aplicación de conceptos vistos en clase, como la estructura de los TDA's, la implementación de tests y la utilización de técnicas como herencia y polimorfismo.

## **GLOSARIO:**

En ambos manuales se utilizan los siguientes términos, por lo que es crucial diferenciarlos y comprender su significado.

Turno: Se define como cuando es el momento de jugar de un jugador.

Ronda: Se define como el momento de jugar de todos los jugadores. Una ronda se divide en cada turno correspondiente, y siempre comienza con el primer jugador. Se dice que finaliza cuando todos los jugadores terminaron sus turnos.

## **Manual de usuario**

### *Configuración antes de comenzar a jugar:*

Al iniciar el juego se le solicitará a los usuarios seleccionar con qué configuración desean jugar al Tateti. Se les pedirá ingresar por consola lo siguiente:

- El tamaño del tablero para jugar que está entre 3 y 99 (Inclusive).
- La cantidad de jugadores con los que se desea jugar.
- El nombre, ficha, y color de cada Jugador

```
Ingrese el tamaño del tablero con el quiere jugar (entre 3 y 99): 3
Ingrese el número de jugadores (entre 2 y 3): 2
Ingrese el nombre del jugador número 1: Juan
Elija la ficha de la lista que desea utilizar CUADRADO CIRCULO TRIANGULO CRUZ RECTANGULO ESTRELLA ROMBO CORAZON: cuadrado
Elija el color de la lista que desea utilizar ROJO VERDE AZUL AMARILLO ROSA CELESTE GRIS NEGRO: rojo
Ingrese el nombre del jugador número 2: pepe
Elija la ficha de la lista que desea utilizar CIRCULO TRIANGULO CRUZ RECTANGULO ESTRELLA ROMBO CORAZON: triangulo
Elija el color de la lista que desea utilizar VERDE AZUL AMARILLO ROSA CELESTE GRIS NEGRO: azul
```

Se les asignará a cada uno una cantidad de fichas en base al tamaño del tablero, y una cantidad máxima de cartas. El juego comienza sin cartas e irán tomándolas de un mazo cada vez que sea su turno. Para tomar cartas, se lanzará un dado y según el número que salga agarrarán (o no) cierta cantidad.

Previo a comenzar el juego, se mostrará por pantalla las características de los jugadores:

```
Nombre: Pepe  
Identificacion: 2  
Cantidad de fichas: 4  
Cantidad de cartas guardadas: 0  
Cantidad de cartas total: 1  
Ficha: ▲
```

*Qué hace el jugador en su turno:*

- Robar cartas: Cada vez que inicia el turno de un jugador éste tira un dado y roba la cantidad de cartas indicada por el mismo siempre y cuando éste no haga que el jugador supere la cantidad máxima de cartas. En ese caso, no se roba ninguna carta. Además, si el mazo está vacío no se roba ninguna carta. Y el jugador tampoco roba carta cuando este pierde su turno ocasionado por la carta Perder Turno.

Si no pudo robar cartas:

```
El jugador Juan no pudo robar cartas ya que el número que salió de tirar el dado  
más las que ya tiene supera la cantidad máxima.
```

Si logró robar cartas:

```
Pepe roba 2 cartas.
```

- Colocar fichas: Mientras el jugador tenga fichas disponibles se le pedirá colocarlas en una casilla del tablero que esté disponible. La disponibilidad se define evaluando si ya está ocupada por otra ficha o si está bloqueada.
- Mover fichas: Una vez que un jugador coloca todas sus fichas disponibles, deberá moverlas por el tablero. Solo se admiten movimientos adyacentes, incluyendo los movimientos dentro de una misma capa y entre dos capas distintas.

Para ingresar las coordenadas, debe tenerse en cuenta que:

X: Sube hacia la derecha, Y: Sube hacia abajo, Z: Sube hacia abajo.



```

Juan indique en que coordenadas desea colocar su ficha.
X (entre 1 y 3): 1
Y (entre 1 y 3): 1
Z (entre 1 y 3): 1

■ - -
- - -
- - -

- - -
- - -
- - -

- - -
- - -
- - -

```

- Jugar cartas: Una vez colocada/movida la ficha, en caso de tener cartas en mano éstas se pueden jugar.

Si el jugador posee cartas pero opta no jugar ninguna, puede ingresar 0 por consola. Si no, se ingresa el número correspondiente al de la carta.

```



1 - Carta Cambiar Ficha.
2 - Carta Anular Casillero.
Ingrese el número de la carta que desea utilizar, 0 para no usar ninguna (entre 0 y 2): 2


X (entre 1 y 3): 3
Y (entre 1 y 3): 1
Z (entre 1 y 3): 1
El casillero fue anulado correctamente.


```

En la próxima sección se especificará qué función tiene cada carta.

Tras finalizar la ronda completa (luego de que jugaron todos los jugadores una vez), se exportará en el directorio donde se está ejecutando el juego una imagen del tablero. La imagen mostrará el estado en el que quedó el tablero tras finalizar la ronda.

	0 1	0 2	0 3
0 1			
0 2			
0 3			

	0 1	0 2	0 3
0 1			
0 2			
0 3			

	0 1	0 2	0 3
0 1			
0 2			
0 3			

### *Cartas:*

La cantidad de cartas presentes en el mazo los jugadores se determina por la cantidad de jugadores, siete cartas por cada jugador.

Cada jugador tiene un límite de cartas a tener en mano, que es la mitad del tamaño del tablero redondeado hacia arriba.

Tipos de cartas:

- Carta Anular Casillero: Esta carta anula un casillero del tablero elegido por el usuario. Para que el jugador pueda jugar esta carta debe elegir un casillero que no esté bloqueado y que se esté vacío (que no tenga una ficha en él).
- Carta Bloquear Ficha: Esta carta bloquea a una ficha de otro jugador que ya estaba previamente colocada en un casillero. Para que el jugador pueda jugar esta carta debe elegir un casillero que no esté bloqueado y que tenga una ficha en él.
- Carta Perder Turno: Esta carta permite bloquear el turno de otro jugador por la duración de la ronda. Para que un jugador pueda jugar esta carta debe elegir un jugador distinto a él mismo.
- Carta Robar 2 Cartas: Esta carta permite que el jugador robe dos cartas del mazo. El jugador debe tener en cuenta que al robar dos cartas no supere el

máximo de cartas, porque sino no las agarrará y la carta desaparecerá de su mano.

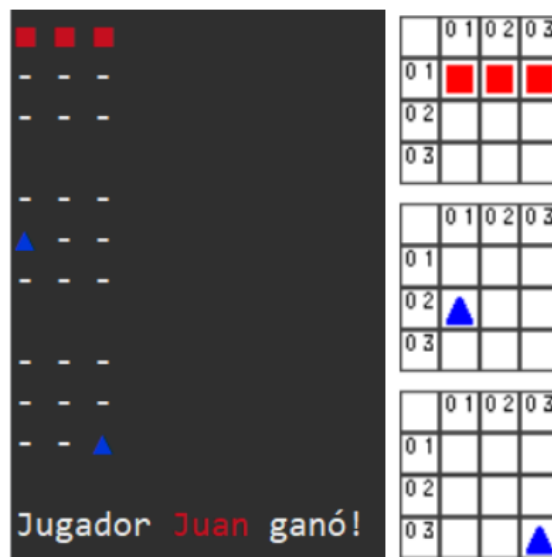
- Carta Bomba: Esta carta destruye las fichas dentro de un rango de 3x3x3 casilleros centrado en el casillero elegido. Si las casillas de este rango estaban bloqueadas o anuladas se revierten a una casilla común vacía. Las fichas se devuelven a cada jugador, pudiendo colocarlas nuevamente.
- Carta Cambiar Fichas: Esta carta cambia el color y símbolo de una ficha de otro jugador a los que corresponden al jugador que la juega.
- Carta Volver Turno: Esta carta regresa una ronda atrás. Se revierten todas las acciones de los jugadores a las de la ronda anterior, empieza la ronda nuevamente el primer jugador de la ronda.

#### *Cómo alcanzar la victoria:*

- Una forma de ganar es dentro de una misma capa como en un tateti común,
- Otra alternativa es aprovechar al máximo que el tablero posee tres dimensiones, ganando poniendo fichas entre capas.

En ambos casos la cantidad de fichas a colocar para ganar depende de la longitud de un lado. Además, al ganar se exportará un tablero con el nombre “Tablero Final”, mostrando el fin del juego con imágenes.

#### **Ganando en una misma capa:**



Ganando entre capas:

▲	■	-
-	-	-
-	-	-
-	-	-
■	▲	-
-	-	-
■	-	-
-	-	-
-	-	▲
Jugador Pepe ganó!		

	0	1	0	2	0	3
0	1	▲	■			
0	2					
0	3					

	0	1	0	2	0	3
0	1					
0	2	■	▲			
0	3					

	0	1	0	2	0	3
0	1	■				
0	2					
0	3				▲	

## Manual del programador

### Finalidad del programa:

La pieza de software desarrollada en Java está destinada a proponer un juego novedoso: un Tateti en formato 3D, agregándole cartas con diversas funcionalidades.

### Paquetes en los que se divide el programa:

La pieza de software se divide en diversos paquetes: **tateti**, **cartas**, **utilidades**, **tests** e **imágenes**.

- **Paquete tateti:** En éste se hallan las clases relacionadas con el juego a programar. En total son nueve: Casillero, Colores, Fichas, Imagen, Jugador, Mazo, Tablero, Menú y JuegoTateti.
- **Paquete cartas:** Al tratarse de múltiples cartas que emplean herencia y polimorfismo, fueron separadas del paquete Tateti.. Las clases incluidas son: Carta, CartaAnularCasillero, CartaBloquearFicha, CartaBomba, CartaCambiarFicha, CartaPerderTurno, CartaRobar2Cartas, CartaVolverTurno.
- **Paquete utilidades:** Contiene clases relacionadas con las diversas estructuras lineales utilizadas, así como otras que efectúan validaciones simples y de uso frecuente. Las clases que lo componen son: Herramientas, Lista, Nodo, Pila y Teclado.
- **Paquete tests:** Los testeos fueron llevados a cabo con JUnit5. Las clases correspondientes a este paquete son: TestTablero, TestCasilleroYFicha, TestJugador y TestCartas.
- **Paquete imágenes:** Dentro de éste se hallan las imágenes en formato .bmp utilizadas para mostrar el tablero.

### Profundizando los paquetes:

#### *Paquete tateti:*

**Clase Casillero:** Cada casillero es inicializado con la correspondiente posición X, Y, Z. Con los métodos dentro de la clase se puede: asignar un jugador

al casillero, desasignar a un jugador, bloquear el casillero y desbloquear el casillero. Respecto al manejo de excepciones, en general se valida que las posiciones no sean menores a cero.

**Clase Imagen:** Crea imágenes .bmp, las manipula de varias formas (recolorizar, añadir borde, juntar), y las exporta.

**Clase Jugador:** Cada jugador es inicializado con el nombre, la cantidad de fichas máximas, la cantidad de cartas máximas a tomar, una selección del enum Fichas, una selección del enum ColoresDisponibles y una Lista de Cartas. Los métodos a destacar dentro de la clase permiten: hacerle perder su turno y robar cartas de un mazo.

**Clase Mazo:** Representa el mazo de cartas del juego, se construye un mazo que recibe como parámetro un entero positivo representando la cantidad de cartas que habrá por cada tipo. Se optó por utilizar una Pila para desarrollarlo y se inicializa con todas las cartas mezcladas. Se proporciona un método para tomar una carta del mazo (que la elimina de éste elimina y la devuelve). Respecto a las validaciones, no se pueden tomar cartas si éste está vacío y se lanza una excepción

**Clase Tablero:** Crea el tablero donde se va a jugar al tateti. Se lo inicializa con un entero que representa al tamaño. El tablero siempre será cúbico. Dentro de los métodos se encuentran: imprimir el tablero por consola, exportar el estado del tablero como una Imagen .bmp, colocar una ficha en un casillero, mover una ficha a otro casillero, y revisar si un movimiento resultó en victoria. Respecto a las validaciones, al inicializar se valida que los tamaños no sean menores a 3 ni mayores a 99. Y al momento de colocar/mover fichas, se verifica la validez de los movimientos (tanto las posiciones x,y,z como los casilleros a los que se desea ir/colocar).

**Clase JuegoTateti:** Cumple la función de main, encargándose de ejecutar el juego de Tateti, comienza con la inicialización del Menu, seguido de los métodos principales: el Tablero, los Jugadores y el Mazo. Luego, inicia el ciclo del juego, una vez que un jugador gane, finaliza el programa mostrando por pantalla el nombre del ganador. Además, posee otro método que aumenta en uno un

contador. Se utiliza en el main con el objetivo de contabilizar las rondas y los turnos. El número de turno es utilizado para establecer adecuadamente cuándo inicia una nueva ronda, ya que cuando el número de turno es mayor o igual a la cantidad de jugadores, se lo vuelve a establecer con valor 1 y se aumenta en uno a la ronda. Por otro lado, el número de ronda es utilizado para generar las imágenes de los tableros por ronda, mientras que el número de turno

**Clase Menu:** En esta clase se crean todos los métodos auxiliares al main. Entre ellos se encuentran: Inicializar Tablero, Jugadores, y Menú, robar cartas del mazo, colocar y mover fichas, y usar las cartas. En cuanto a las validaciones en Menu, son mínimas ya que las validaciones en los distintos métodos a lo largo del código le dieron robustez para que en cualquier caso de uso siga funcionando de la mejor manera y no permita el mal uso de los mismos.

**Enum Fichas:** Contiene las diversas opciones en cuanto a fichas: cuadrado, círculo, triángulo, cruz, rectángulo, estrella, rombo y corazón. Al inicializarse el jugador, debe incluirse el tipo de ficha con el que se va a jugar y lo verá reflejado en el tablero 3D. Dentro de este enum hay una estructura switch-case donde, según lo que se eligió para jugar, se establece un char que es el que verá en el tablero por consola.

**Enum Colores:** Define los colores disponibles: rojo, verde, azul, amarillo, rosa, celeste, gris y negro. Cada color está ligado a un valor int que representa ese color en formato RGB, se incluye una estructura switch-case que, según el color seleccionado por el jugador, devuelve su valor RGB. Además, hay otro método que devuelve su ANSI Escape Code para utilizar con printf. Al inicializar un jugador, es necesario asignar qué color lo representará, este color se verá reflejado en el tablero 3D dentro de la consola.

### *Paquete cartas:*

**Clase Carta:** Se optó por utilizar herencia y polimorfismo al crear las cartas, por lo que esta clase es abstracta y las demás cartas heredan de la misma.

**Clase CartaAnularCasillero:** Esta carta anula un casillero del tablero elegido por el usuario. Al utilizarse se debe ingresar por consola un casillero y se llama al método de casillero que alterna el bloqueo. Respecto a las validaciones antes de su uso, verifica que dicho casillero esté vacío y que no esté bloqueado. Tras ser anulado, ninguna ficha podrá ser colocada o movida sobre el mismo.

**Clase CartaBloquearFicha:** Esta carta bloquea a una ficha de otro jugador que ya estaba previamente colocada en un casillero. Al utilizarse se llama al método de casillero que alterna el bloqueo. Respecto a las validaciones antes de su uso, se verifica que el casillero no esté vacío ni bloqueado. Tras ser bloqueado la ficha no se la podrá mover.

**CartaBomba:** Utiliza el entorno del casillero para tener un acceso rápido a los casilleros que rodean al elegido para poner la bomba. Verificar si existen, están bloqueados y si tiene fichas. Al explotar la bomba, deja todo el entorno (incluido el casillero donde se la colocó) en un estado inicial, devolviendo las fichas que se levantan a cada jugador.

**CartaCambiarFicha:** Se solicita ingresar un casillero no vacío ni bloqueado. Cambia el propietario de un casillero, cambiando la ficha y el color que estaban anteriormente al del usuario que la juega.

**Clase CartaPerderTurno:** Se pide ingresar un jugador (por identificación), el cual perderá el turno. Si ya usaron esta carta en él dentro de la misma ronda, se deberá elegir a otra persona. No está permitido bloquearse a uno mismo.

**CartaRobarCartas:** Se roban dos cartas del mazo. Si el jugador no puede llevar a cabo la acción porque supera el máximo de sus cartas posibles, la carta se elimina y el jugador no roba nada, malgastando la carta.

**CartaVolverTurno:** Al utilizar esta carta el tablero vuelve al estado que tenía cuando jugó el primer jugador (la ronda anterior), levantando las fichas puestas por el primer jugador en adelante. Se lleva a cabo creando un tablero auxiliar tras el final de cada ronda.



### *Paquete utilidades:*

**Clase Herramientas:** Contiene múltiples métodos genéricos que se utilizan frecuentemente a lo largo del programa. Entre ellos se encuentran: validar si un número es positivo (estricto y no estricto), devolver un dígito según la posición, validar un RGB dado y reiniciar el color de la terminal.

**Clase Lista:** Representa una estructura lineal de lista genérica, que es utilizada en varias clases distintas. Contiene los métodos básicos de una lista: agregar un elemento, quitar un elemento, ver si está vacía, iniciar el cursor, avanzar el cursor, obtener la longitud y contar las ocurrencias de un valor. Respecto a las validaciones, existen dos métodos validarPosicionParaAgregar y validarPosicionEnLista que verifican las posiciones y se utilizan en varios métodos de la clase mencionados.

**Clase Nodo:** Inicializa un nodo y cuenta con los métodos para obtener el dato dentro de ese nodo, obtener la dirección al siguiente nodo y modificar ambos datos. Esta clase es utilizada en las clases Lista y Pila.

**Clase Pila:** Inicializa una pila que cuenta con los métodos de agregar un elemento al tope, obtener el tope, evaluar si está vacía, quitar un elemento del tope e invertir la pila.

**Clase Teclado:** Contiene métodos genéricos utilizados para pedirle inputs al usuario. Entre ellos se hallan: pedir un String, pedir un número y pedir un número entre un intervalo. Una particularidad de estos métodos es que todos reciben un String mensaje, el cual puede ser incluido o no (puede ser nulo). En caso de estar vacío, se reemplaza al mensaje por uno genérico.

### *Paquete tests:*

**Clase TestTablero:** Los testeos de esta clase se centran en probar si las validaciones al inicializar el tablero funcionan correctamente, haciendo énfasis en el entorno del tablero.

**Clase TestCasilleroYFicha:** Verifica todas las alternativas que pueden surgir al colocar y mover una ficha en un casillero, además de corroborar que los casilleros se bloqueen correctamente.

Al utilizar el método para colocar fichas, se testea su funcionalidad si éste ya está ocupado por otra ficha o si está bloqueado.

Al mover fichas, además de controlar que no se puedan mover sobre casilleros ocupados/bloqueados, se testea mover una ficha bloqueada. Otra implementación que se verifica es la de mover la ficha a lugares adyacentes, incluyendo el movimiento entre capas. También se intenta mover la ficha en los extremos del tablero: por ejemplo, si está en el extremo derecho, se ve qué sucede si se intenta mover a la derecha (que es una posición inválida).

Además, en este apartado se verifica si el programa indica correctamente cuándo se gana una partida, tanto dentro de una capa como entre distintas capas del tablero.

**Clase TestJugador:** Se prueba inicializar el jugador con sus distintos parámetros inválidos y se verifica las funcionalidades relacionadas con el mazo de cartas general y las cartas del usuario.

Al robar cartas se verifica que dicho método funcione, que robar carta no exceda el máximo de cartas establecido, que no se pueda robar cero cartas y que no se pueda robar de un mazo inválido.

**Clase TestCartas:** Se prueba utilizar todas las cartas del juego en un contexto simple y conciso. En Menu y JuegoTateti se utilizan muchos métodos que solicitan un input del usuario, por lo que para el correcto funcionamiento de los testeos se especifica mediante prints qué es lo que debería ingresarse para el test de cada carta.