

## 75.10 – Técnicas de Diseño

### Trabajo Práctico – Nikoli Games

### Iteracion 2

## Enunciado

- 1) Se deberá incorporar al motor de generación de juegos los juegos indicados.
- 2) Se deberá terminar de incorporar la actividad de Entrega 1.  
Se deberá respetar el JSON definido para las jugadas de entrada y salida para facilitar el armado de test de aceptación sobre los juegos.
- 3) Se debe poder **configurar** el juego (lo mas posible) de un archivo JSON, es decir, se debería poder crear un juego nuevo, definiendo en un archivo sus reglas, su board, y sus pistas iniciales. **Sin necesitar escribir código ni recompilar. Es decir se puede mezclar en la configuración las reglas existentes y definir su contenido, para armar o inventar una nueva variante.**

## Listado de Juegos

- Se deberán implementar los siguientes juegos que validan sus soluciones:
  - Country Road
  - Gokigen Naname
  - Slitherlink
  - Inshi no Heya (Actividad 1)

**Se puede mezclar en la configuración las reglas existentes y definir su contenido para armar o inventar una nueva variante.**

**Los requerimientos pueden (y van a) cambiar en cualquier momento.**

## Restricciones

- Trabajo Práctico en grupos de 6 alumnos implementado en java.
- Se deben utilizar las mismas herramientas que en el TP0.
- Todas las clases del sistema deben estar justificadas.
- Todas las clases deben llevar un comentario con las responsabilidades de la misma.
- El uso de herencia debe estar justificado. Se debe explicar claramente el porqué de su conveniencia por sobre otras opciones.
- Se debe tener una cobertura completa del código por tests.

## Criterios de Corrección

- Cumplimiento de las restricciones
- Documentación entregada
- Diseño del modelo
- Diseño del código
- Test unitarios
- **No se aceptarán TPs:**
  - **Con warnings.**
  - **Que no compilen en Travis-CI.**
  - **Con issues abiertos.**
  - **Que no se puedan utilizar.**
- **Que el TP no fue aceptado, significa que tiene esa entrega directamente desaprobada.**

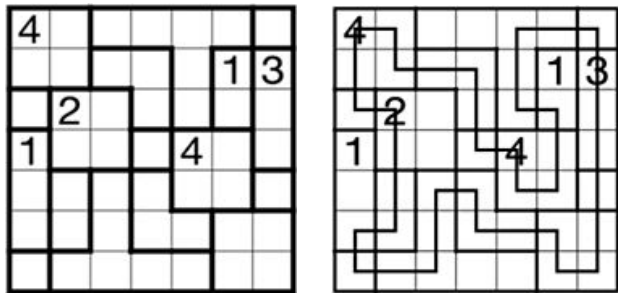
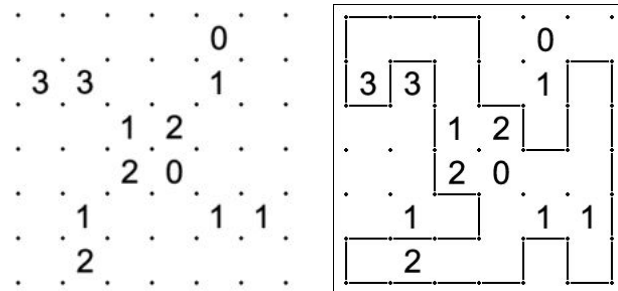
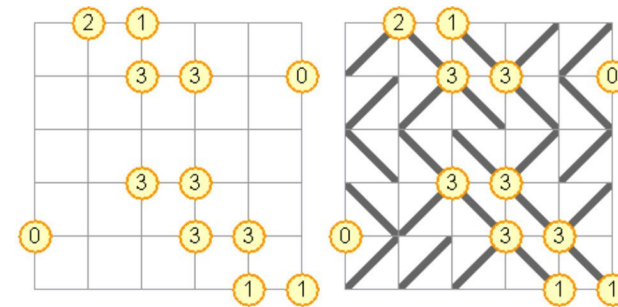
Se tendrán en cuenta también la completitud del tp, la correctitud, distribución de responsabilidades, aplicación y uso de criterios y principios de buen diseño, buen uso del repositorio y uso de buenas prácticas en general.

## Aclaraciones

- Para las consultas generales se va a utilizar el channel **#tp-grupal-channel**.
- Cada grupo va a tener su channel para consultas particulares y comunicación con su ayudante.
- Se debe usar el mismo toolset que para el TP0 (java 8, IntelliJ, gradle, pmd, cpd, checkstyle, findbugs).
- Se supone que todos están al tanto de las notificaciones en Slack. Esto incluye cambios de alcance, cambios en los requerimientos, restricciones adicionales, etc.
- Se podrán modificar la configuración del toolset sólo bajo aprobación de los ayudantes (#quejas-tooling).
- Cada grupo tendrá un repositorio en GitHub sobre el cual deberá trabajar en grupo.
- Se debe integrar el repositorio con Travis-CI.
- La documentación se realizará en la misma Wiki del repositorio en GitHub.
- El readme.md deberá tener un resumen con una breve reseña del propósito del proyecto y una explicación de como usarlo, y link a la wiki.
- En la fecha de entrega se realizará una demo del TP a uno o dos ayudantes. Esto implica que el TP se debe poder utilizar dentro de las restricciones de la entrega.
- No hay re-entrega. Es responsabilidad del grupo realizar las consultas durante las 2-3 semanas disponibles para realizar el TP.
- Si un alumno no puede concurrir a la demo, **deberá avisar con anticipación** y se coordinará una nueva fecha para que dicho alumno haga la presentación y defensa de la entrega.

- Durante la demo y posterior corrección se cargarán issues en GitHub que deben estar solucionados para la siguiente entrega.

## Juegos para esta Iteración

<b>COUNTRY ROAD</b>	<b>REGLAS</b>
	<p>Dibujar un circuito que vaya horizontal o verticalmente entre los centros de las celdas y que:</p> <ul style="list-style-type: none"><li>- Pase a lo sumo una vez por cada celda y cada región.</li><li>- Si una región tiene un número, pase por esa cantidad de celdas en ella.</li><li>- Si deja celdas contiguas sin visitar, estas pertenecen a la misma región.</li></ul>
<b>SLITHERLINK</b>	<b>REGLAS</b>
	<p>Dibujar un circuito sobre los bordes de las celdas, de modo que las celdas numeradas tienen tantos bordes en el circuito como su valor.</p>
<b>GOKIGEN NANAME</b>	<b>REGLAS</b>
	<p>Completar cada celda con líneas diagonales.</p> <p>Los números indican cuántas líneas tocan a ese número.</p> <p>Las líneas no pueden formar ningún circuito.</p>

## Herramientas a utilizar

- Java  $\geq$  1.8
- Gradle  $\geq$  2.6
- JUnit  $\geq$  4.11
- Git -> Github
- CheckStyle / PMD / Findbugs

## Cronograma tentativo (puede ir adaptándose)

8 de Septiembre	Armado de grupos
15 de Septiembre	Publicación 1 <sup>ra</sup> Entrega
22 de Septiembre	
29 de Septiembre	<b>Primera Entrega</b> - Publicación 2 <sup>da</sup> Entrega
6 de Octubre	Entrega de notas 1 <sup>ra</sup> Entrega
13 de Octubre	
20 de Octubre	<b>Segunda Entrega</b> - Publicación 3 <sup>ra</sup> Entrega
27 de Octubre	Entrega de notas 2 <sup>da</sup> Entrega
3 de Noviembre	
10 de Noviembre	<b>Tercera Entrega</b>
17 de Noviembre	Revision
24 de Noviembre	Publicación de notas finales