Proyecto I – Dots

Instituto Tecnológico de Costa Rica
Área de Ingeniería en Computadores
Algoritmos y Estructuras de Datos I (CE 1103)
Segundo Semestre 2018
Valor 25%

Objetivo General

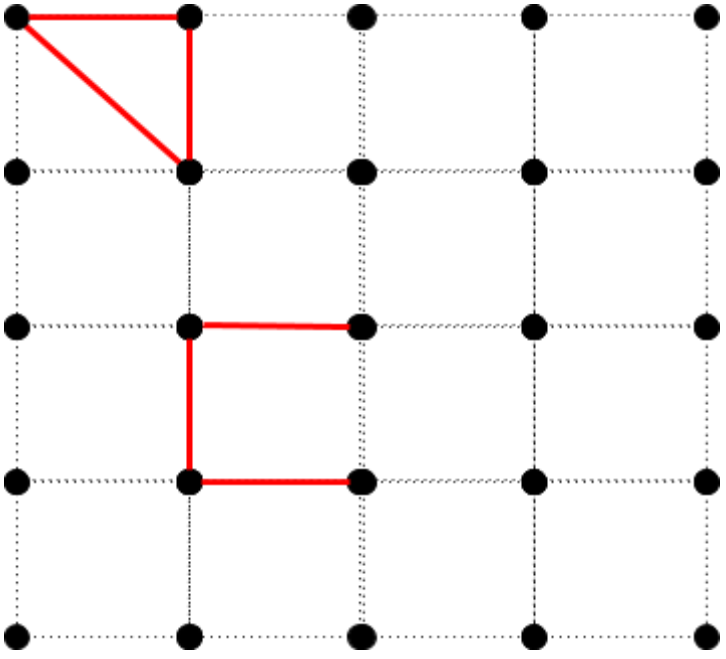
- ➔ Diseñar e implementar un juego multiplayer.

Objetivos Específicos

- ➔ Implementar listas enlazadas y algunas variaciones
- ➔ Investigar y desarrollar una aplicación en el lenguaje de programación Java
- ➔ Investigar acerca de programación orientada a objetos en Java.
- ➔ Aplicar **patrones de diseño** en la solución de un problema.

Descripción del Problema

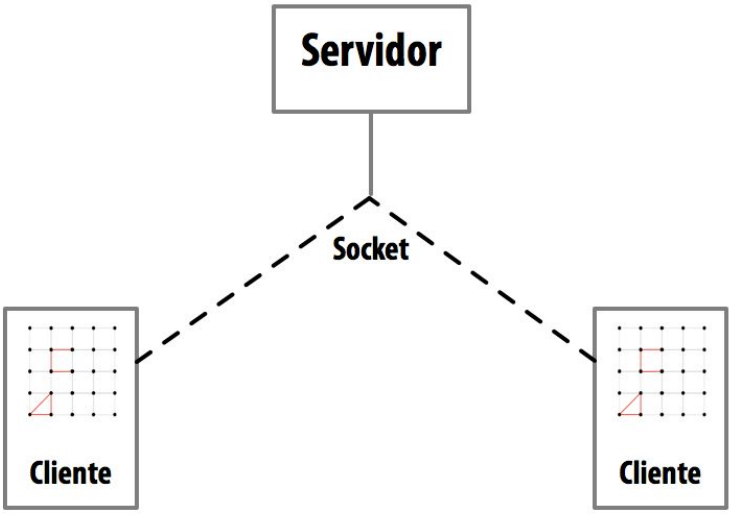
Dots es un juego multiplayer. El juego consiste en una malla de puntos donde el jugador puede unir dichos puntos mediante un segmento de línea por turno. El objetivo del juego es formar figuras geométricas para obtener puntos. El jugador que complete una o más figuras geométricas con el segmento de su turno, obtendrá los puntos según la figura formada y la cantidad de figuras.



Las figuras geométricas tienen diferentes puntajes según la cantidad de segmentos que las compongan y su forma. Cada segmento vale dos puntos. Cuando una figura se cierra, el área dentro de esta cambia de color y no puede utilizarse más (es decir ningún jugador puede unir puntos en el área interna).

Dots es un juego multiplayer. Hay un servidor central que consiste en una aplicación en Java que escucha las conexiones entrantes por un Socket. Cada aplicación cliente se ejecuta en una computadora y se conecta por socket al servidor. Cuando el juego va a iniciar, el servidor recibe la petición del cliente y lo ingresa en una

cola. En el momento que hayan dos o más jugadores, empieza el juego. Únicamente juegan dos jugadores a la vez.
El servidor lleva el control completo del juego. Los clientes únicamente grafican lo que el servidor les envían. Los clientes a su vez envían las acciones que realicen al servidor, el cual se encarga de mantener el estado del juego completo. Toda la comunicación entre cliente y servidor es en formato JSON.



Aspectos de implementación
Sobre el cliente

- No hay manejo de lógica del servidor.
- El cliente hace pull cada *n* segundos para obtener el estado del juego del servidor.
- El cliente toma la respuesta del servidor y la dibuja en la pantalla.
- El usuario interactúa con la malla de puntos y envía mensajes en JSON al servidor indicando la acción efectuada.
- No puede utilizar matrices ni ninguna clase de Java para modelar la malla del juego
- La malla (una vez que se carga posterior a la recepción del mensaje del server) se implementa como una lista de listas.
- Se implementa en Java.

Sobre el servidor

- La cola de jugadores se modela como una cola implementada por los estudiantes.
- No puede utilizar matrices ni ninguna clase de Java para modelar la malla del juego
- La malla es implementada por los estudiantes utilizando listas de listas.
- Recibe las acciones del cliente y actualiza la malla.
- Envía el estado de la malla cuando el cliente lo solicite.
- Se implementa en Java.
- Cuando el juego termina, debe indicarlo a los clientes y crear un nuevo juego para los jugadores en cola.

Tanto el cliente como el servidor pueden utilizar la biblioteca Jackson para serializar/deserializar JSON.

Documentación requerida

1. Internamente, el código se debe documentar utilizando JavaDocs y se debe generar el HTML de la documentación.
2. Dado que el código se deberá mantener en GitHub, la documentación externa se hará en el Wiki de GitHub. El Wiki deberá incluir:
    a. Breve descripción del problema
    b. **Planificación y administración del proyecto:** se utilizará la parte de project management de GitHub para la administración de proyecto. Debe incluir:
        ➔ Lista de features e historias de usuario identificados de la especificación
        ➔ Distribución de historias de usuario por criticalidad
        ➔ Plan de iteraciones que agrupen cada bloque de historias de usuario de forma que se vea un desarrollo incremental
        ➔ Descomposición de cada user story en tareas.
        ➔ Asignación de tareas a cada miembro del equipo.
    c. Diagrama de clases en formato JPEG o PNG
    d. Descripción de las estructuras de datos desarrolladas.
    e. Descripción detallada de los algoritmos desarrollados.
    f. Problemas encontrados en forma de bugs de *github*: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.

Aspectos operativos y evaluación:

1. **Fecha de entrega: De acuerdo al cronograma del curso**
2. El proyecto tiene un valor de 25% de la nota del curso.
3. El trabajo es **en grupos de 3 personas**.
4. Es obligatorio utilizar un GitHUB.
5. Es obligatorio integrar toda la solución.
6. El código tendrá un valor total de 75%, la documentación 20% y la defensa 5%.
7. De las notas mencionadas en el punto anterior se calculará la Nota Final del Proyecto.
8. Se evaluará que la documentación sea coherente, acorde a la dificultad/tamaño del proyecto y el trabajo realizado, se recomienda que realicen la documentación conforme se implementa el código.
9. La nota del código NO podrá exceder en 35 puntos la nota de la documentación, por lo cual se recomienda documentar conforme se programa.
10. La documentación se revisará según el día de entrega en el cronograma.
11. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
12. Los estudiantes pueden seguir trabajando en el código hasta 15 minutos antes de la cita revisión oficial
13. Aún cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones
    a. Si no se entrega documentación, automáticamente se obtiene una nota de 0.
    b. Si no se utiliza un manejador de código se obtiene una nota de 0.
    c. Si la documentación no se entregan en la fecha indicada se obtiene una nota de 0.
    d. Sí el código no compila se obtendrá una nota de 0, por lo cual se recomienda realizar la defensa con un código funcional.
    e. El código debe ser desarrollado en Java, en caso contrario se obtendrá una nota de 0.
    f. Si no se siguen las reglas del formato de email se obtendrá una nota de 0.
    g. La nota de la documentación debe ser acorde a la completitud del proyecto.

14. La revisión de la documentación será realizada por parte del profesor, no durante la defensa del proyecto. El único requerimiento que se consultará durante la defensa del proyecto es el diagrama de clases, documentación interna y la documentación en el manejador de código.
15. Cada estudiante tendrá como máximo 15 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cual se recomienda tener todo listo antes de ingresar a la defensa.
16. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final del proyecto.
17. Cada estudiante es responsable de llevar los equipos requeridos para la revisión, si no cuentan con estos deberán avisar al menos 2 días antes de la revisión a el profesor para coordinar el préstamo de estos.
18. Durante la revisión únicamente podrán participar el estudiante, asistentes, otros profesores y el coordinador del área.