Proyecto I – Connect dots

Instituto Tecnológico de Costa Rica
Área de Ingeniería en Computadores
Algoritmos y Estructuras de Datos I (CE 1103)
II Semestre 2023
Valor 20%

Objetivo general

- ➔ Implementar un juego que permita la aplicación del Paradigma Orientado a Objetos mediante la utilización de estructuras de datos lineales.

Objetivos específicos

- ➔ Diseñar una solución que permita resolver el problema descrito en esta especificación aplicando patrones de diseño y utilizando el estándar UML.
- ➔ Implementar una solución que permita resolver el problema descrito en esta especificación utilizando Programación Orientada a Objetos en Java y haciendo uso de estructuras de datos lineales.
- ➔ Elaborar la documentación correspondiente a la solución implementada para la evidencia del trabajo desarrollado utilizando estándares de documentación técnica y herramientas de gestión de proyectos

Atributos relacionados

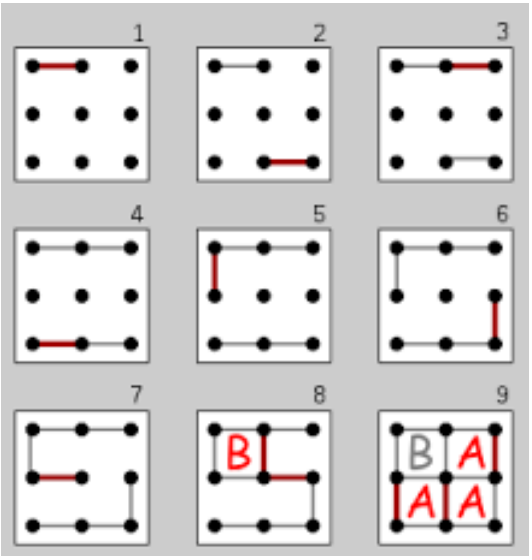
A continuación se describen los atributos del graduado que se pretenden abordar con el desarrollo del proyecto.

Diseño (DI)

Diseña soluciones creativas para problemas de ingeniería complejos y diseña sistemas, componentes o procesos para satisfacer las necesidades identificadas con la consideración adecuada para la salud y la seguridad públicas, el costo total de la vida, el carbono neto cero, así como las consideraciones de recursos, culturales, sociales y ambientales según sea necesario.

Descripción del problema

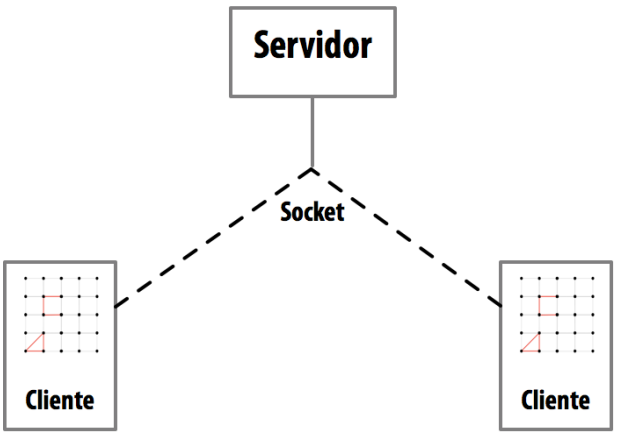
Connect dots es un juego multiplayer de *n* jugadores. El juego consiste en una malla de puntos donde el jugador puede unir dichos puntos mediante líneas por turno. Si el jugador logra cerrar un cuadrado en su turno, puede seguir jugando hasta que agregue una línea que no cierre un cuadrado. El objetivo del juego es cerrar cuadrados para obtener puntos. El jugador que logre cerrar más cuadrados al final de la partida, será el ganador.



Todos los cuadrados tienen el mismo valor. Cuando un jugador cierra un cuadrado, a este se le asignará algún identificador que permita saber cuál fue el jugador que lo cerró.

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. Pueden jugar varios jugadores a la vez.

El servidor lleva el control completo del juego. Los clientes únicamente grafican lo que el servidor les envía. 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.



Adicionalmente, un cliente tendrá un control elaborado por los estudiantes que permitirá al jugador seleccionar dónde colocará las líneas.

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.
- 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, así como cualquier biblioteca para manejo de la interfaz gráfica.

Sobre el control

- Debe ser implementado en Arduino.
- El control permitirá al usuario moverse en las cuatro direcciones (arriba, abajo, derecha e izquierda), así como seleccionar un punto.
- Se sugiere hacer el control con cinco botones.
- El grupo debe realizar la maqueta del control para interactuar con la aplicación. No se permite presentar el circuito únicamente en la protoboard.
- Se sugiere el uso de las bibliotecas jSerialComm o RXTX para la comunicación serial con Arduino.

Documentación requerida

1. Internamente, el código se debe documentar utilizando JavaDocs y se debe generar el HTML de la documentación.
2. La documentación externa se hará en Latex (preferiblemente, pero también puede ser elaborado en alguna otra herramienta), y deberá contener al menos estas secciones.
    a. Descripción del problema
    b. Diagrama de clases

    c. Descripción de las estructuras de datos desarrolladas.
    d. Corridas de ejemplo con capturas de pantalla y explicaciones en prosa.
3. Se deberá planificar el proyecto en Microsoft Azure Devops o en Jira.
    a. Plan de iteraciones que agrupen cada bloque de historias de usuario por sprint, de forma que se vea un desarrollo incremental. Se deberán de crear tres sprints.
    b. Asignación de tareas a cada miembro del equipo.
4. Documentación de diseño: En un documento por aparte deben presentar las siguientes secciones:
    a. **Listado de requerimientos del sistema:** Cada grupo deberá identificar las necesidades y los requerimientos de un problema complejo de ingeniería considerando la salud y la seguridad pública, el costo total de la vida, el carbono neto cero, así como aspectos relacionados con recursos, culturales, sociales y ambientales según sea necesario.
    b. **Elaboración de opciones de solución al problema:** Para el problema planteado deberán documentarse al menos dos opciones de solución. Cada solución deberá ser acompañada de algún tipo de diagrama. **Estas opciones de solución no deben ser fácilmente descartables y deben llevar un análisis objetivo con base en criterios técnicos o teóricos.**
    c. **Valoración de opciones de solución:** Se deberán valorar alternativas de solución para un problema complejo de ingeniería que cumplan con necesidades específicas, considerando la salud y la seguridad pública, el costo total de la vida, el carbono neto cero, así como aspectos relacionados con recursos, culturales, sociales y ambientales según sea necesario.
    d. **Selección de la propuesta final:** Se deberá seleccionar una propuesta final de las opciones de solución, de acuerdo con los criterios de comparación.
    e. **Diseño de la alternativa seleccionada:** Se deberá documentar completamente el diseño final seleccionado considerando la salud y la seguridad pública, el costo total de la vida, el carbono neto cero, así como aspectos relacionados con recursos, culturales, sociales y ambientales según sea necesario. Para el caso de este proyecto esto incluye (en caso de que aplique): descripción del protocolo utilizado, diagrama de bloques del sistema, diagramas propios de diseño de sofware aplicables (de flujo, clases, composición, UML, patrones de diseño, etc).
    f. **Validación del diseño:** se deberá validar el diseño final de acuerdo con los requerimientos, la salud y la seguridad pública, el costo total de la vida, el carbono neto cero, así como aspectos relacionados con recursos, culturales, sociales y ambientales según sea necesario.

Aspectos operativos y evaluación

1. **Fecha de entrega: De acuerdo al cronograma del curso**
2. El proyecto tiene un valor de 20% de la nota del curso.
3. El trabajo es **en grupos de 3 personas**.
4. Es obligatorio utilizar un GitHub para el manejo de las versiones. Se debe evidenciar el uso de Commits frecuentes.

5. Deben entregar en el TEC Digital un documento con el link del repositorio de GitHub, Azure DevOps y el PDF de las documentaciones. Para ambas herramientas deben dar acceso al correo del profesor.
6. Es obligatorio integrar toda la solución.
7. La presentación funcional tendrá un valor de 70%, la documentación externa 15% y la documentación de diseño 15%.
8. De las notas mencionadas en el punto anterior se calculará la Nota Final del Proyecto.
9. 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.
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. La nota de la documentación debe ser acorde a la completitud del proyecto.
    g. Si alguna persona integrante del proyecto no se presenta a la revisión se le asignará una nota de cero en la nota final 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 grupo tendrá como máximo 30 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.