UTN
Facultad Regional Córdoba

TUP

Tecnicatura Universitaria en Programación

LABORATORIO DE COMPUTACIÓN III

Trabajo Práctico Integrador
Estanciero

TPI
2° Año – 3° Cuatrimestre

CC BY ND NC

Índice

Trabajo Práctico Integrador 2

Introducción 2
Objetivos 3
Requerimientos funcionales. 4
- Contenido del juego físico 4
- Inicio del juego 4
- Movimiento en el tablero: 5
- Compra y venta de propiedades: 5
- Construcción de mejoras: 5
- Eventos y situaciones especiales: 5
- Gestión financiera: 6
- Finalización del juego: 6
- Perfiles virtuales 6

Requerimientos No Funcionales 7
1. Rendimiento y optimización: 7
2. Calidad del código y buenas prácticas de programación: 7
3. Pruebas unitarias y cobertura de código: 8
4. Usabilidad y accesibilidad: 8
5. Diseño de software y patrones de diseño: 8
6. Seguridad: 8

Forma de trabajo 9
Forma y fechas de entrega 9
- Seguimiento 10

Evaluación y Calificación: 11

Trabajo Práctico Integrador

Introducción

El juego del Estanciero es una adaptación argentina del clásico juego de mesa Monopoly. Fue creado en 1942, y desde entonces ha sido un juego de mesa popular en Argentina y otros países de habla hispana. El nombre "Estanciero" hace referencia al término utilizado en Argentina para referirse a los propietarios de grandes extensiones de tierra, también conocidos como estancias. Este juego recrea la dinámica económica de la compra, venta y desarrollo de propiedades, así como la gestión de recursos financieros. El objetivo del juego es acumular la mayor cantidad de dinero y propiedades posibles, mientras se intenta evitar la bancarrota. Los jugadores recorren un tablero que representa un mapa ficticio de Argentina, en el que pueden comprar diferentes tipos de propiedades, como estancias, campos, estaciones de tren, etc. Además, se enfrentan a diferentes eventos y situaciones que pueden afectar su economía, como impuestos, mejoras en las propiedades, alquileres, entre otros. Los jugadores compiten entre sí para adquirir propiedades estratégicas y construir sobre ellas para aumentar su valor y generar ingresos. El juego se desarrolla a lo largo de varias rondas, y el jugador que logre acumular la mayor cantidad de riqueza al final del juego es declarado el ganador.

En este contexto, como parte de la materia Laboratorio de Computación III, se presenta el desafío de desarrollar el juego “El Estanciero” funcional en Java. Este desafío no solo requiere conocimientos técnicos en programación, sino también una comprensión profunda de las reglas y la lógica del juego, así como la capacidad de aplicar conceptos aprendidos en las clases para crear un juego interactivo y totalmente funcional.

Objetivos

El objetivo principal de este trabajo integrador es aplicar los conocimientos adquiridos durante las clases de Laboratorio de Computación III para desarrollar el juego del estanciero completamente funcional en Java. Esto implica diseñar e implementar un programa que permita a un jugador competir contra jugadores autómatas, siguiendo todas las reglas del juego y permitiendo las funcionalidades esenciales, como la compra y venta de propiedades, el cálculo de alquileres, el manejo de dinero, entre otros aspectos.

El juego desarrollado también debe incluir características avanzadas que demuestren la comprensión y aplicación de los conceptos de programación aprendidos, como la implementación de patrones de diseño, la gestión eficiente de la memoria, la modularidad y la reutilización de código, la manipulación de estructuras de datos complejas, la implementación de algoritmos de búsqueda y evaluación de estados del juego. También deben aplicarse los conocimientos adquiridos sobre la gestión del código fuente, su versionado y manipulación.

El desarrollo del juego en Java requerirá la aplicación de conocimientos de programación orientada a objetos, manejo de estructuras de datos, algoritmos, manejo de interfaces gráficas de usuario, pruebas unitarias, y el uso de herramientas como Maven, JUnit, Mockito y JaCoCo, entre otros.

Este proyecto no solo permitirá a los estudiantes aplicar los conceptos y técnicas aprendidas en el aula, sino que también los desafiará a enfrentarse a situaciones reales de desarrollo de software, como la planificación, diseño, implementación, pruebas y entrega de una aplicación de software completa y funcional. Además, el uso de herramientas de control de versiones y la colaboración en un entorno de desarrollo colaborativo como GitHub les brindará una experiencia valiosa en el uso de herramientas profesionales ampliamente utilizadas en la industria.

Opcionalmente, los alumnos podrán desarrollar (en un proyecto aparte) una interfaz gráfica que se integre al proyecto de BackEnd mediante apis Rest y que sea visualmente atractivo y fácil de usar, que permita al jugador interactuar de manera intuitiva con los elementos del juego.

Requerimientos funcionales.

La siguiente lista de requerimientos debe cumplirse para que el trabajo se considere completo y correcto, **a excepción de los requerimientos opcionales, que serán considerados como parte de la nota normativa**.

Contenido del juego físico

- 1 tablero de juego
- 6 peones (uno de cada color)
- 29 tarjetas (escrituras)
- 32 tarjetas (suerte y destino)
- 32 chacras
- 12 estancias
- 2 dados
- billetes de diferentes valores y colores.

Inicio del juego

- El jugador puede iniciar la partida y elegir entre 3 niveles de dificultad (difícil, medio y fácil)
    a. El **modo difícil** incluye 4 jugadores virtuales extra cada uno con un perfil de juego diferente (1 agresivo, 2 moderados y 1 conservador).
    b. El **modo medio** incluye 3 jugadores virtuales extra, cada uno con un perfil de juego diferente (1 agresivo, 1 moderado y 1 conservador).
    c. El **modo fácil** incluye 2 jugadores virtuales extra, cada uno con un perfil de juego diferente (1 moderado y 1 conservador).
- Se debe asignar un orden de turno a los jugadores definida por el lanzamiento de los dados (se tiran los dados y quien saque el número más alto iniciará el mismo e irán jugando en orden descendente por el valor de los dados). Si se repitiera el número, se asignará el turno de manera aleatoria a alguno de ellos.
- Al inicio del juego el jugador puede elegir opcionalmente si se permite ganar el juego por cantidad de valores totales; si opta por esto, debe ingresar cual es el monto al que debe arribar un jugador para ganar la partida.
- Cada jugador inicia con 35.000 en billetes.
- Todos los jugadores inician desde la salida.

Movimiento en el tablero:

- Los jugadores deben poder lanzar los dados y moverse por el tablero según el resultado del lanzamiento. (Esto último debe suceder automáticamente)
- Si un jugador cae en una casilla de propiedad no adquirida, debe tener la opción de comprarla.
- Si un jugador cae en una casilla de propiedad adquirida por otro jugador, debe pagar el alquiler correspondiente.
- Si en los dados salen números iguales, (dos 1 o dos 4, etc.) una vez hecho el movimiento de adelantar y efectuadas las transacciones que indique la casilla a la que ha llegado, vuelve a tirar los dados, pudiendo esto repetirse dos veces, pero si sucediera una tercera tiene que ir a la cárcel.
- El pago de los alquileres será automático cuando un jugador caiga en una propiedad de otro jugador.

Compra y venta de propiedades:

- Los jugadores deben poder comprar propiedades cuando caigan en casillas de propiedad no adquirida.
- Los jugadores deben poder vender propiedades en su poder al banco.
- **Opcionalmente:** Los jugadores deben poder vender propiedades en su poder a otros jugadores.

Construcción de mejoras:

Los jugadores deben poder construir mejoras (Campos, Chacras y Estancias) en las propiedades que poseen, siempre y cuando cumplan con los requisitos necesarios.

La venta de Chacras y Estancias es exclusiva entre el Propietario y el Banco.

Eventos y situaciones especiales:

Los jugadores deben poder enfrentarse a eventos y situaciones especiales, como impuestos, premios, tarjetas de suerte, entre otros, que pueden afectar su economía.

Cuando un jugador llega a una casilla con la denominación Suerte o Destino tiene que levantar la tarjeta correspondiente y luego de cumplir lo que indica de manera automática.

Los Premios e Impuestos son cobrados o pagados por el Banco de manera automática.

Un jugador preso tiene que permanecer en la casilla comisaría. Si cae en la casilla no se le considera preso.

Al llegar a la casilla de descanso, el jugador tiene derecho, si así lo desea, a quedarse por dos turnos, siempre que no saque doble en los dados. Es condición avisar antes de tirar los dados los deseos de quedarse.

Gestión financiera:

Los jugadores deben poder gestionar su dinero decidiendo sobre la compra de propiedades y construcción de mejoras. El pago de alquileres, impuestos y premios será ejecutado automáticamente por el sistema.

Finalización del juego:

- El juego debe finalizar cuando un jugador acumule una cantidad determinada de dinero o cuando todos los demás jugadores hayan quedado en bancarrota.
- Se debe mostrar un mensaje indicando al jugador ganador al finalizar la partida.

Perfiles virtuales

- Perfil de jugador conservador:
    a. Este jugador tiende a ser cauteloso y prioriza la acumulación de propiedades de bajo costo. (Provincias prioritarias: Formosa, Río Negro y Salta)
    b. Este jugador comprará fuera de las provincias de preferencia 1 de cada 5 propiedades que compre.
    c. Construirá mejoras solo cuando el costo de la construcción no sobrepase el 30% de su dinero en cuenta.
- Perfil de jugador equilibrado:
    a. Este jugador busca un equilibrio entre la acumulación de propiedades y la construcción de mejoras. (Provincias prioritarias: Mendoza, Santa Fe y Tucuman)
    b. Este jugador buscará comprar la serie de Ferrocarriles.
    c. Este jugador comprará fuera de las provincias de preferencia 1 de cada 3 propiedades que compre.
    d. Construirá mejoras cuando el costo de la construcción no supere el 50% de su dinero en cuenta o cuando se hayan vendido más del 75% de las propiedades.
    e. **Opcionalmente**: Si se implementa la compra/venta entre jugadores, este perfil buscará comprar para completar sus provincias de preferencia y estará dispuesto a pagar hasta el 100% del valor original de la propiedad.

- Perfil de jugador agresivo:
    a. Este jugador busca maximizar el retorno de inversión rápidamente, incluso a costa de correr mayores riesgos. (Provincias prioritarias: Tucuman, Córdoba y Buenos Aires)
    b. Este jugador buscará comprar la serie de Ferrocarriles y Companias.
    c. Este jugador no comprará fuera de las provincias de preferencia a no ser que ya se hayan vendido a otros jugadores al menos una propiedad de todas sus zonas de preferencia o el mismo haya completado sus zonas; en dicho caso, comprará tantas propiedades como pueda.
    d. Priorizará la expansión rápida y construirá mejoras cada vez que pueda.
    e. **Opcionalmente**: Si se implementa la compra/venta entre jugadores, este perfil buscará comprar para completar sus provincias de preferencia y estará dispuesto a pagar hasta el 200% del valor original de la propiedad.

Requerimientos No Funcionales

La siguiente lista de requerimientos debe cumplirse para que el trabajo se considere completo y correcto, **a excepción de los requerimientos opcionales, que serán considerados como parte de la nota normativa**.

1. **Rendimiento y optimización:**
    a. El juego debe tener un rendimiento óptimo, con una respuesta rápida a las interacciones del usuario, incluso en condiciones de carga de trabajo intensa.
    b. Se debe realizar una optimización adecuada del código y del uso de recursos, considerando la eficiencia en la ejecución del juego y la gestión de memoria.
2. **Calidad del código y buenas prácticas de programación:**
    a. El código debe seguir las convenciones de codificación de **Java (en versión 11)** y mantener una alta calidad en términos de legibilidad, mantenibilidad y escalabilidad.
    b. Se debe aplicar el principio de responsabilidad única (Single Responsibility Principle) y otros principios de diseño **SOLID** para garantizar una arquitectura de software robusta y modular.
    c. Se debe utilizar **Maven** como herramienta de gestión de dependencias y construcción del proyecto, siguiendo las mejores prácticas de estructura de proyectos y gestión de versiones.

3. **Pruebas unitarias y cobertura de código:**
    a. Se deben implementar pruebas unitarias utilizando **JUnit** y **Mockito** para garantizar la calidad del código y su correcto funcionamiento.
    b. Se debe lograr una cobertura de código con **JaCoCo** de **al menos 80%**, asegurando que las pruebas cubran la mayoría de las funcionalidades del juego.
4. **Usabilidad y accesibilidad:**
    a. La interfaz de usuario debe ser intuitiva y fácil de usar, con una experiencia de usuario agradable.
    b. **Opcional**: En caso de hacer una interfaz web, se deben considerar las mejores prácticas de desarrollo web.
5. **Diseño de software y patrones de diseño:**
    a. Se debe aplicar un enfoque de diseño de software modular y bien estructurado, utilizando patrones de diseño apropiados para mejorar la calidad y mantenibilidad del código.

        Ejemplos de patrones de diseño que podrían ser aplicados en el desarrollo del juego incluyen el patrón de diseño de Strategy para gestionar los perfiles de usuario virtuales.

6. **Seguridad:**
    a. Se debe evitar el uso de librerías (dependencias) que poseen vulnerabilidades críticas/high conocidas.
    b. **Opcional**: Se deben implementar medidas de seguridad adecuadas para proteger la integridad del juego y la confidencialidad de los datos del usuario, si los hubiera.
    c. **Opcional**: Se deben evitar vulnerabilidades conocidas, como inyección de código, desbordamiento de búfer y otros ataques comunes.

Es importante que los alumnos consideren estos requerimientos no funcionales como parte integral del desarrollo del juego, para garantizar un software de alta calidad, eficiente y con características profesionales.

Forma de trabajo

Para este proyecto, **los alumnos trabajarán en grupos de 6 o 7 personas**, lo que fomentará el trabajo en equipo y la colaboración entre los miembros del grupo. Cada equipo será responsable de autogestionarse y organizarse de manera eficiente para llevar a cabo todas las etapas del desarrollo del juego, desde el diseño y la implementación hasta las pruebas y la entrega final.

Una parte fundamental de este proyecto será **el uso de herramientas de control de versiones, específicamente Git y GitHub, usando el workflow "GitFlow"** aprendido en clases. Cada equipo deberá utilizar Git como sistema de control de versiones para mantener un registro de los cambios realizados en el código fuente del juego y trabajar de manera colaborativa en un repositorio de GitHub. Además, **se creará un aula virtual (classroom) en GitHub específica para este proyecto**, donde los equipos podrán acceder a los recursos necesarios, realizar seguimiento del progreso del proyecto y entregar sus entregables.

El trabajo en equipo y la colaboración serán elementos clave en el desarrollo del proyecto. Los equipos deberán dividirse las tareas de manera equitativa, asignando responsabilidades a cada miembro en función de sus habilidades y conocimientos. Además, deberán comunicarse de manera efectiva para coordinar y sincronizar sus esfuerzos, asegurándose de que todos los miembros del equipo estén alineados y contribuyan activamente al progreso del proyecto.

Forma y fechas de entrega

La entrega del trabajo se realizará a través del aula virtual (classroom) específica para este proyecto en GitHub. Se establecerá una fecha límite de entrega, la cual deberá ser respetada por todos los equipos. Es importante tener en cuenta que todos los requerimientos funcionales y no funcionales obligatorios deberán estar cumplidos en la fecha de entrega para poder aprobar el proyecto. Cualquier entrega realizada después de la fecha límite será considerada como tardía y estará sujeta a penalizaciones en la evaluación final del proyecto.

Se debe trabajar con el branch model GitFlow, esto quiere decir que las ramas estables de main y develop estarán bloqueadas para hacer push directo, y solo podrán llegar a ellas por medio de un Pull Request (PR) que deberá estar aprobado por el docente. Para trabajar en esta forma, se requerirá que abra una rama por cada semana de trabajo y el pull request esté solicitado antes de la fecha de entrega pactada. Las correcciones se harán sobre el PR semanal.

La fecha de entrega será fijada hasta el día Sábado 22 de Junio del 2024 a las 12:00 Hs.

Seguimiento

Para un seguimiento constante del progreso de los alumnos en el desarrollo del trabajo, se establecerán fechas de entrega parciales que incluirán los entregables que deberán presentar. Se establece un día de entrega para así tener tiempo para corregir la entrega para la próxima clase. Aquí se deja un esquema de cómo se abordarán las entregas, en semanas a partir de la presentación del mismo.

- **Semana 1:** Entregar un documento en formato markdown en el repositorio, en la carpeta /docs, con todas las clases base detectadas como necesarias para resolver el problema, con sus atributos, métodos y responsabilidades. Entregar un modelo DER con todas sus entidades completas.
- **Semana 2**: Entregar un documento en formato markdown en el repositorio, en la carpeta /docs, con la explicación de la experiencia de usuario que se desarrollará, incluyendo los menús de navegación, acciones, eventos y presentaciones que se usarán.
- **Semana 3**: Entregar un documento en formato markdown en el repositorio, en la carpeta /docs, con todas las interfaces detectadas, que métodos proveerán y que abstracción pretenden representar.
- **Semana 4**: Entregar desarrolladas todas las clases base en el package “model” y los scripts .sql para la creación de las tablas y su población en el caso de tablas que deben estar pre cargas.
- **Semana 5**: Entregar un documento en formato markdown en el repositorio, en la carpeta /docs, con los patrones de diseño que se pretenden utilizar con su explicación de porqué seleccionaron dicho patrón y qué problema pretende resolver.
- **Semana 6:** Entregar todas las interfaces desarrolladas dentro del package “services”
- **Semana 7**: Entregar todas las clases que modelan la base de datos dentro del package “entities”.
- **Semana 8**: Entregar la implementación de cada interface desarrollada dentro del package “services.impl”. Incluir en la entrega los test unitarios de cada implementación con al menos un 80% de cobertura de código.
- **Semana 9 - 11**: Terminar de desarrollar toda la aplicación con sus respectivos Test. En esta entrega se debe completar el 100% del trabajo TPI.

IMPORTANTE: Este esquema indica lo mínimo requerido para cada semana, pero no limita que cada equipo puede avanzar más allá de este plan adelantando etapas.

Evaluación y Calificación:

Es importante destacar que cumplir con los requerimientos establecidos es un criterio mínimo para aprobar el proyecto, pero para obtener una evaluación sobresaliente se valorará también la calidad del código, la eficiencia del diseño y la implementación, la cobertura de pruebas, la claridad de la documentación y la capacidad de trabajo en equipo y colaboración demostrada durante el desarrollo del proyecto.

- Se evaluará la calidad, complejidad y presentación del trabajo como así también la oportunidad de entrega en fecha.
- Este práctico tiene carácter de obligatorio y deberá aprobarse con un 60% (al igual que los parciales) para regularizar la materia.
- La calificación alcanzada por el grupo de trabajo será la calificación individual de cada uno de los integrantes del equipo.

Atribución-No Comercial-Sin Derivadas

Se permite descargar esta obra y compartirla, siempre y cuando no sea modificado y/o alterado su contenido, ni se comercialice. Universidad Tecnológica Nacional Facultad Regional Córdoba (S/D). Material para la Tecnicatura Universitaria en Programación, modalidad virtual, Córdoba, Argentina.