[75.07 / 95.02]

Algoritmos y programación III

Trabajo práctico 2: GPS Challenge

(trabajo grupal)

Alumno:

Nombre	Padrón	Mail

Tutor:

Nota Final:

Índice

1. Objetivo

Desarrollar una aplicación de manera grupal aplicando todos los conceptos vistos en el curso, utilizando un lenguaje de tipado estático (Java) con un diseño del modelo orientado a objetos y trabajando con las técnicas de TDD e Integración Continua.

2. Consigna general

Desarrollar la aplicación completa, incluyendo el **modelo** de clases e interfaz gráfica. La aplicación deberá ser acompañada por pruebas unitarias e integrales y documentación de diseño.

3. Especificación de la aplicación a desarrollar

La empresa Algo Ritmos SA dedicada al desarrollo de video juegos a decidido contratar a un grupo de programadores para implementar el juego GPS Challenge

Game play

GPS es un juego de estrategia por turnos. El escenario es una ciudad y el objetivo, guiar un vehículo a la meta en la menor cantidad de movimientos posibles.

El juego se jugará por turnos, y en cada turno el usuario decide hacia cual de las 4 esquinas posibles avanzará.

Vehículos

El jugador podrá optar por tres diferentes tipos de vehículos.

- moto
- auto
- 4x4

Obstáculos

Al atravesar una cuadra el jugador se podrá encontrar con alguno de los siguientes obstáculos:

- Pozos: Le suma 3 movimientos de penalización a autos y motos. Para una 4x4 penaliza en 2 movimientos luego de atravesar 3 pozos.
- Piquete: Autos y 4x4 deben pegar la vuelta, no pueden pasar. Las motos pueden pasar con una penalización de 2 movimientos.
- Control Policial: Para todos los vehículos la penalización es de 3 movimientos, sin embargo la probabilidad de que el vehículo quede demorado por el control y sea penalizado es de 0,3 para las 4x4, 0,5 para los autos y 0,8 para las motos ya que nunca llevan el casco puesto.

Sorpresas

También se podrán encontrar diferentes tipos de sorpresas:

- Sorpresa Favorable: Resta el 20% de los movimientos hechos.
- Sorpresa Desfavorable: Suma el 25% de los movimientos hechos.
- Sorpresa Cambio de Vehículo: Cambia el vehículo del jugador. Si es una moto, la convierte en auto. Si es un auto lo convierte en 4x4. Si es una 4x4 la convierte en moto.

Las sorpresas figuran en el mapa como un regalo y no se sabrá que es hasta que el vehículo la accione.

Escenario

Para hacerlo más interesante y jugable, el jugador no podrá ver más que dos manzanas a la redonda de la posición de su vehículo y la bandera a cuadros que marca la meta. El resto del mapa permanecerá en sombras.

El tamaño del escenario no será fijo, y tendrá un punto de partida y una meta.

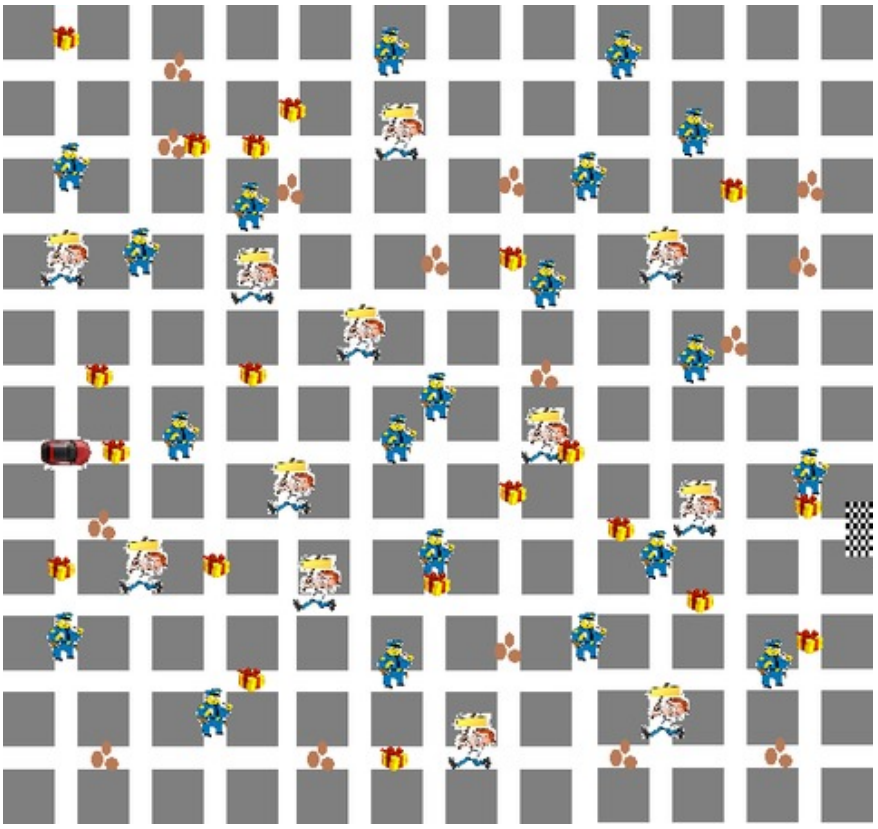

Figura 1: Ejemplo de escenario

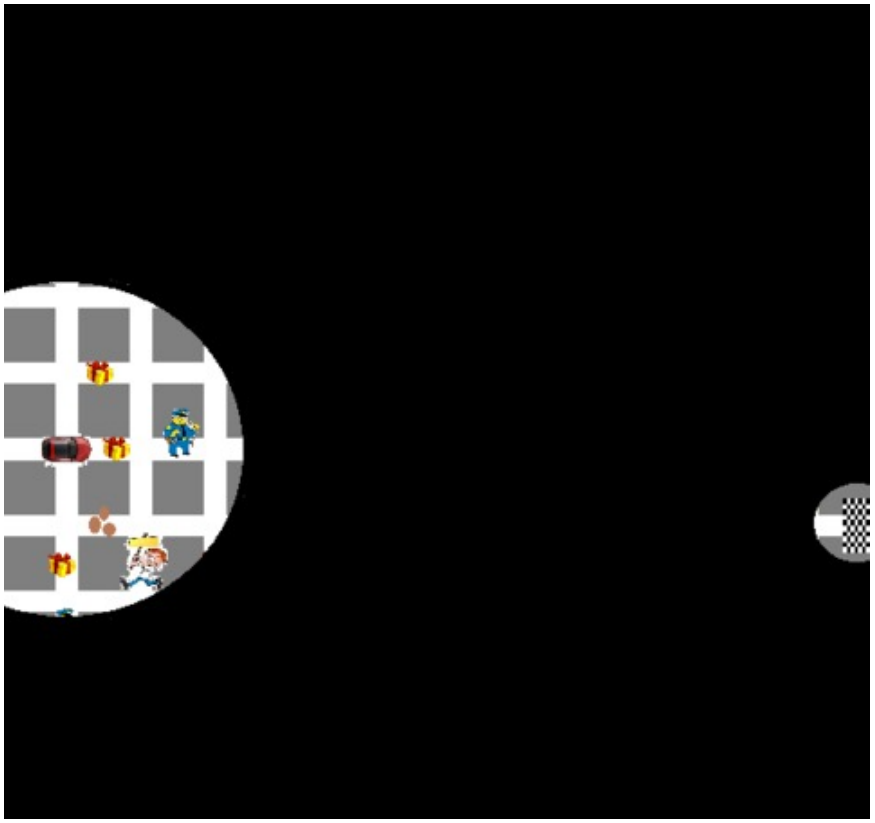

Figura 2: Ejemplo de escenario como lo visualiza el jugador

Puntajes altos

Se debe almacenar un ranking donde figuren los mejores puntajes asociados a un nickname que indique el usuario.

4. Interfaz gráfica

La interacción entre el usuario y la aplicación deberá ser mediante una interfaz gráfica intuitiva. Consistirá en una aplicación de escritorio utilizando **JavaFX** y se pondrá mucho énfasis y se evaluará como parte de la consigna su **usabilidad**. (*en el anexo I se explicarán algunas buenas prácticas para armar la interfaz gráfica de usuario o GUI*)

5. Herramientas

1. **JDK (Java Development Kit)**: Versión 1.8 o superior.
2. **JavaFX**
3. **JUnit 5**: Framework de pruebas unitarias para Java.
4. **IDE (Entorno de desarrollo integrado)**: Su uso es opcional y cada integrante del grupo puede utilizar uno distinto o incluso el editor de texto que más le guste. Lo importante es que el repositorio de las entregas no contenga ningún archivo de ningún IDE y que la construcción y ejecución de la aplicación sea totalmente independiente del entorno de desarrollo. Algunos de los IDEs más populares son:
    a. Eclipse
    b. IntelliJ
    c. Netbeans
5. **Herramienta de construcción:** Se deberán incluir todos los archivos XML necesarios para la compilación y construcción automatizada de la aplicación. El informe deberá contener instrucciones acerca de los comandos necesarios (preferentemente también en el archivo README.md del repositorio). Puede usarse Maven o Apache Ant con Ivy.
6. **Repositorio remoto**: Todas las entregas deberán ser subidas a un repositorio único en GitHub para todo el grupo en donde quedarán registrados los aportes de cada miembro. El repositorio puede ser público o privado. En caso de ser privado debe agregarse al docente corrector como colaborador del repositorio.
7. **Git**: Herramienta de control de versiones
8. **Herramienta de integración continua**: Deberá estar configurada de manera tal que cada *commit* dispare la compilación, construcción y ejecución de las pruebas unitarias automáticamente. Algunas de las más populares son:
    a. Travis-CI
    b. Jenkins
    c. Circle-CI
    d. GitHub Actions (recomendado)

Se recomienda basarse en la estructura del proyecto base armado por la cátedra.

6. Entregables

Para cada entrega se deberá subir lo siguiente al repositorio:

1. **Código fuente de la aplicación completa**, incluyendo también: código de la prueba, archivos de recursos.
2. **Script para compilación y ejecución** (Ant o Maven).
3. **Informe**, acorde a lo especificado en este documento (en las primeras entregas se podrá incluir

solamente un enlace a Overleaf o a Google Docs en donde confeccionen el informe e incluir el archivo PDF solamente en la entrega final).

No se deberá incluir ningún archivo compilado (formato .class) ni tampoco aquellos propios de algún IDE (por ejemplo .idea). Tampoco se deberá incluir archivos de diagramas UML propios de alguna herramienta. **Todos los diagramas deben ser exportados como imágenes de manera tal que sea transparente la herramienta que hayan utilizado para crearlos.**

7. Formas de entrega

Habrá **4 entregas formales** que tendrán una calificación de **APROBADO o NO APROBADO** en el momento de la entrega. Además, se contará con una entrega 0 preliminares.

Aquel grupo que acumule 2 no aprobados, quedará automáticamente desaprobado con la consiguiente pérdida de regularidad en la materia de **todos** los integrantes del grupo. *En cada entrega se deberá incluir el informe actualizado*.

8. Evaluación

El día de cada entrega, cada ayudante convocará a los integrantes de su grupo, solicitará el informe correspondiente e iniciará la corrección mediante una entrevista grupal. **Es imprescindible la presencia de todos los integrantes del grupo el día de cada corrección.**

Se evaluará el trabajo grupal y a cada integrante en forma individual. El objetivo de esto es comprender la dinámica de trabajo del equipo y los roles que ha desempeñado cada integrante del grupo. Para que el alumno apruebe el trabajo práctico debe estar aprobado en los dos aspectos: grupal e individual (se revisarán los commits de cada integrante en el repositorio).

Dentro de los ítems a chequear el ayudante evaluará aspectos formales (como ser la forma de presentación del informe), aspectos funcionales: que se resuelva el problema planteado y aspectos operativos: que el TP funcione integrado.

9. Entregables para cada fecha de entrega

Cada entrega consta de las **pruebas + el código** que hace pasar dichas pruebas.

Entrega 0 (Semana del 30 de Mayo)

- Planteo de modelo tentativo, diagrama de clases general y diagrama de secuencia para el caso de uso: *Una moto atraviesa la grilla sin obstáculos y la cantidad de movimientos es X*.
- Repositorio de código creado según se explica aquí.
- Servidor de integración continua configurado.
- Al menos un commit realizado por cada integrante, actualizando el `README.md`.

Entrega 1 (Semana del 06 de Junio)

Pruebas (sin interfaz gráfica)

`Caso de uso 1`

- Una moto atraviesa la ciudad y se encuentra con un Pozo. Es penalizada en tres movimientos.

`Caso de uso 2`

- Un auto atraviesa la ciudad y se encuentra con un Pozo. Es penalizada en tres movimientos.

`Caso de uso 3`

- Una 4x4 atraviesa la ciudad y se encuentra con un Pozo. No es penalizada.

`Caso de uso 4`

- A cargo del equipo.

`Caso de uso 5`

- A cargo del equipo.

Entrega 2 (Semana del 13 de Junio)

Pruebas (sin interfaz gráfica)

`Caso de uso 1`

- Un vehículo atraviesa la ciudad y encuentra una sorpresa favorable.

`Caso de uso 2`

- Un vehículo atraviesa la ciudad y encuentra una sorpresa desfavorable.

`Caso de uso 3`

- Un vehículo atraviesa la ciudad y encuentra una sorpresa cambio de vehículo.

`Caso de uso 4`

- A cargo del equipo.

`Caso de uso 5`

- A cargo del equipo.

Entrega 3 (Semana del 20 de Junio)

1. Modelo del juego terminado
2. Interfaz gráfica inicial básica: comienzo del juego y visualización del tablero e interfaz de usuario básica.

Entrega 4 - Final: (Semana del 27 de Junio)

1. Trabajo Práctico completo funcionando, con interfaz gráfica final, sonidos e informe completo.

Tiempo total de desarrollo del trabajo práctico:

5 semanas

10. Informe

El informe deberá estar subdividido en las siguientes secciones:

Supuestos

Documentar todos los supuestos hechos sobre el enunciado. Asegurarse de validar con los docentes.

Diagramas de clases

Varios diagramas de clases, mostrando la relación estática entre las clases. Pueden agregar todo el texto necesario para aclarar y explicar su diseño de manera tal que el modelo logre comunicarse de manera efectiva.

Diagramas de secuencia

Varios diagramas de secuencia, mostrando la relación dinámica entre distintos objetos planteando una gran cantidad de escenarios que contemplen las secuencias más interesantes del modelo.

Diagrama de paquetes

Incluir un diagrama de paquetes UML para mostrar el acoplamiento de su trabajo.

Diagramas de estado

Incluir diagramas de estados, mostrando tanto los estados como las distintas transiciones para varias entidades del modelo.

Detalles de implementación

Deben detallar/explicar qué estrategias utilizaron para resolver todos los puntos más conflictivos del trabajo práctico. Justificar el uso de herencia vs. delegación, mencionar que principio de diseño aplicaron en qué caso y mencionar qué patrones de diseño fueron utilizados y por qué motivos.

IMPORTANTE

No describir el concepto de herencia, delegación, principio de diseño o patrón de diseño. Solo justificar su utilización.

Excepciones

Explicar las excepciones creadas, con qué fin fueron creadas y cómo y dónde se las atrapa explicando qué acciones se toman al respecto una vez capturadas.

Anexo 0: ¿Cómo empezar?

Requisitos, análisis

¿Cómo se empieza? La respuesta es **lápiz y papel**. No código!.

1. Entiendan el dominio del problema. Definir y utilizar un lenguaje común que todo el equipo entiende y comparte. Ej.: Si hablamos de “X entidad”, todos entienden que es algo ... Si los conceptos son ambiguos **nunca podrán** crear un modelo congruente.
2. **Compartan** entre el grupo, pidan opiniones y refinen la idea en papel **antes de sentarse a programar.** Pueden escribir en ayudante-virtual si tienen dudas

Anexo I: Buenas prácticas en la interfaz gráfica

El objetivo de esta sección es recopilar algunas recomendaciones en la creación de interfaces gráficas de usuario o GUI. La idea no es exigir un diseño refinado y prolijo, sino que pueda ser usado por el grupo de docentes de Algo3 sin impedimentos “lo mejor posible”.

Prototipo

Venimos escribiendo código, integrales y UML todo el cuatrimestre. Me están pidiendo una interfaz gráfica. ¿Cómo se empieza? La respuesta es **lápiz y papel**. No código!.

1. Armen un **dibujo** o **prototipo en papel** (pueden usar google docs o cualquier herramienta también) de todas las “pantallas”. No tiene que ser perfecto.
2. **Compartan** entre el grupo, pidan opiniones y refinen la idea en papel **antes de sentarse a programar.** Pueden escribir en ayudante-virtual si tienen dudas

JavaFX

Entender cómo funciona JavaFX es clave para implementar la GUI correctamente. **No subestimen el tiempo que lleva implementar y modificar la UI o interfaz de usuario.** Lean todo lo que ofrece y las buenas prácticas de la tecnología. Hay plugins específicos para el IDE, pero por más que usen herramientas WYSIWYG, siempre conviene entender la API para mejorar el código autogenerado que casi nunca es óptimo.

Recomendaciones visuales

Tamaño de elementos

- Se recomienda un tamaño de tipografía de al menos a **10 puntos al mayor contraste** negro contra blanco para asegurar la legibilidad, o bien 12 puntos**.**
- Para los botones o elementos interactivos, el **tamaño mínimo** del área debería ser de **18x18**.

- Extra: Los elementos más importantes deberían ser más grandes y estar en posiciones más accesibles (poner ejemplos)

Contraste

- Aseguren que el texto y los elementos tengan **buen contraste** y se puedan leer bien
- Se sugiere un contraste cercano a 3 entre textos y fondos para texto grande e imágenes.
- Herramientas para verificar contraste: https://colourcontrast.cc/ https://contrast-grid.eightshapes.com

Uso del color

- La recomendación es **no utilizar más de 3 colores** para la UI y los elementos
- En el enunciado se ejemplifica con una paleta accesible, pero pueden usar cualquiera para las fichas
- Accesibilidad: Si usan para las fichas **rojo** y **verde**, o **verde** y **azul**, aseguren que las personas con daltonismo puedan distinguirlas usando letras o símbolos sobre las mismas.
- Dudas eligiendo paletas? https://color.adobe.com

Tipografía

- No se recomienda usar más de 2 tipografías para toda la aplicación
- Asegurarse de que esas tipografías se exporten correctamente en en TP
- Evitar tipografías “artísticas” para texto, menú y botones, ya que dificultan la lectura

Recomendaciones de interacción

Manejo de errores

- **No escalar excepciones a la GUI**. Es un No absoluto. Enviar a la consola.
- Si muestran errores al usuario, el mensaje de error debe estar escrito **sin jerga técnica** y **permitir al usuario continuar** y entender lo que está pasando
- Siempre optar por validar y prevenir errores, a dejar que el usuario ejecute la acción y falle.

Confirmaciones

- Antes de cerrar o ejecutar cualquier operación terminal, una buena práctica es pedir confirmación al usuario (para el alcance del tp no sería necesario)

Visibilidad del estado y otros

- Mostrar el estado en el cual está el juego. Siempre debería estar accesible
- Permitir al usuario terminar o cerrar en cualquier momento