[75.07 / 95.02]

Algoritmos y programación III

Trabajo práctico 2 (enunciado)

Segundo cuatrimestre del año 2020

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 aplicación consiste en un juego que permite aprender los conceptos básicos de programación, armando algoritmos utilizando bloques visuales.

Los algoritmos permitirán a un personaje moverse por la pantalla mientras dibuja con un lápiz, logrando realizar distintos diseños.

La aplicación se compondrá de 3 secciones:

- Tablero: el tablero comenzará siendo un espacio en blanco en el cual se colocará al personaje en su posición inicial
- Lista de bloques: la lista de bloques mostrará los bloques que el jugador tendrá disponibles para utilizar
- Algoritmo: el algoritmo comenzará en blanco, y el jugador podrá ir colocando bloques en esta sección, que luego se ejecutarán en forma secuencial

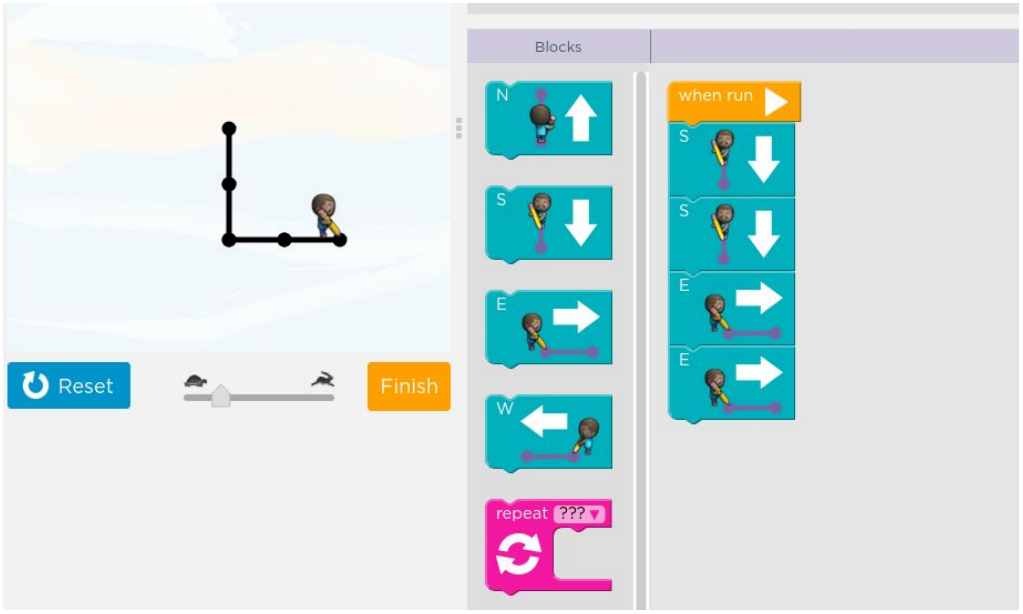
Una vez armado el algoritmo, el jugador podrá elegir “ejecutar” el mismo. En ese caso, cada bloque se irá procesando en forma secuencial, haciendo que el personaje realice la acción correspondiente.

El objetivo del juego es permitir al jugador experimentar con distintos algoritmos para ir aprendiendo los conceptos básicos de la programación.

En el siguiente link puede verse un ejemplo visual que muestra una aplicación similar en formato web:

https://studio.code.org/s/pre-express-2020/stage/8/puzzle/2

Y el siguiente es un screenshot de ese sitio:



3.1. Entidades

- ### Personaje:

El personaje se moverá por el tablero y realizará secuencialmente las acciones que indiquen los bloques del algoritmo.

- ### Tablero:

Es el tablero en el cual se mueve el personaje y en el cual se irán dibujando las figuras de acuerdo a las acciones que el personaje realiza.
- Sector bloques disponibles
Es el sector en el cual se mostrarán los bloques que el usuario puede utilizar.
-Sector algoritmo:
Es el sector en el cual el usuario puede ir colocando bloques en forma secuencial que luego serán ejecutados.
-Sector dibujo:
Es el sector en el cual se mueve el personaje, dibujando el mismo según corresponda.

- ### Bloques:

Existen distintos tipos de bloques:

- **Mover arriba**
  - Mueve al personaje una posición hacia arriba
- **Mover abajo**
  - Mueve al personaje una posición hacia abajo
- **Mover derecha**

    - Mueve al personaje una posición hacia la derecha
- **Mover izquierda**
    - Mueve al personaje una posición hacia la izquierda
- **Bajar lápiz**
    - Indica al personaje que coloque la punta del lápiz en el piso, de forma tal que cualquier movimiento que realice a continuación dibuje en el tablero
- **Subir lápiz**
    - Indica al personaje que coloque la punta del lápiz hacia arriba, de forma tal que cualquier movimiento que realice a continuación no dibuje en el tablero
- **Repetir 2 veces**
    - Este bloque permite colocar dentro de este otros bloques (cualquiera de los bloques disponibles). Al ejecutarse, ejecuta secuencialmente los bloques contenidos en el mismo dos veces seguidas
- **Repetir 3 veces**
    - Este bloque permite colocar dentro de este otros bloques (cualquiera de los bloques disponibles). Al ejecutarse, ejecuta secuencialmente los bloques contenidos en el mismo tres veces seguidas
- **Invertir comportamiento**
    - Este bloque permite colocar dentro de este otros bloques (cualquiera de los bloques disponibles). Al ejecutarse, ejecuta secuencialmente los bloques contenidos en el mismo una vez, pero cada bloque contenido se ejecutará en forma inversa (ejemplo, el bloque de mover a la derecha, moverá al personaje a la izquierda, el bloque de bajar lápiz subirá el lápiz, etc).
- **Algoritmo personalizado**
    - Este bloque no está disponible inicialmente y solamente se crea cuando el usuario hace click en el botón “Guardar algoritmo” que hará que el nuevo bloque pueda reproducir la misma ejecución de bloques que los que estaban dispuestos al momento de guardar el algoritmo. Es necesario que exista al menos un bloque al momento de hacer click en el botón de guardado. Al hacer click en el botón el usuario debe ingresar un nombre para el algoritmo que será adjudicado en el nuevo bloque para ser utilizado posteriormente.

3.2. Flujo del programa

- Fase inicial: El jugador coloca los bloques en la zona del algoritmo.
- Fase de ejecución: Cuando el jugador elige ejecutar el algoritmo, los bloques se procesan secuencialmente, haciendo que el personaje realice las acciones correspondientes.

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**.

5. Herramientas

1. **JDK (Java Development Kit)**: Versión 1.8 o superior.
2. **JavaFX**
3. **JUnit**: 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 utilizarse 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

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 preliminar.

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 del vencimiento 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

Se sobreentiende que cada entrega consta de las **pruebas + el código** que hace pasar dichas pruebas).

Entrega 0 (Semana del 7 de Diciembre)

- Planteo de modelo tentativo, diagrama de clases general y diagrama de secuencia para el caso de ejecución de un algoritmo con un solo bloque de mover a la derecha.

Entrega 1 (Semana del 14 de Diciembre)

- Pruebas:

- Creacion del personaje, por defecto con el lápiz arriba
- Levantar y bajar el lápiz del personaje (usando los bloques correspondientes)
- Mover al personaje en todas las direcciones (usando los bloques correspondientes)

Entrega 2 (Semana del 8 de Febrero)

- Pruebas:

- Creación del sector dibujo (en el modelo, sin interfaz gráfica)

- Mover al personaje con el pincel arriba y abajo, verificando que el sector dibujo quede dibujado según corresponda
- Creación de un algoritmo usando los bloques de repetición

Entrega 3 (Semana del 15 de Febrero)

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 1 de Marzo)

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

Tiempo total de desarrollo del trabajo práctico:

6 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 logre 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. Mencionar qué patrones de diseño fueron utilizados y por qué motivos.

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.