UBA
Universidad de Buenos Aires

[75.07 / 95.02]

Algoritmos y programación III

Trabajo práctico 2 (enunciado)

Segundo cuatrimestre del año 2018

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 Contínua.

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 por turnos basado en el clásico juego Age of Empires II. Para ello vamos a especificar las unidades disponibles, edificios y reglas del juego.

3.1. Unidades

Nombre:	Aldeano
Vida:	50
costo:	25 oro
Se crea en edificio:	Plaza central
Habilidades:	• Construye edificios (plaza central y cuarteles) • Repara edificios (todos) • recolectar oro

Nombre:	Espadachín
Vida:	100
costo:	50 oro
Se crea en edificio:	Cuartel

Habilidades:	• Ataque cuerpo a cuerpo (distancia = 1) a:
	  ○ otras unidades (daño = 25)
	  ○ edificios (daño = 15)

Nombre:	Arquero
Vida:	75
costo:	75 oro

Se crea en edificio:	Cuartel
Habilidades:	● Ataque a distancia <= 3 a: ○ otras unidades (daño = 15) ○ edificios (daño = 10)

Nombre:	Arma de asedio
Vida:	150
costo:	200 oro
Se crea en edificio:	Castillo
Habilidades:	● Ataque a distancia <= 5 ○ edificios (daño = 75) Para poder atacar el arma de asedio debe primero montarse. Eso consume un turno. Al turno siguiente estará disponible para atacar todos los edificios que estén en su rango. Mientras el arma de asedio se encuentra desmontada se puede mover (pero no atacar). Cuando está montada puede atacar, pero no moverse (hay que desarmarla, consume 1 turno, para que se pueda volver a mover)

3.2. Edificios

Nombre:	Plaza central
Vida:	450
Costo:	100 oro
Crea unidades:	Aldeano
Turnos en construirse:	3 turnos
Tamaño:	4 casilleros
Velocidad de reparación:	Recupera +25 de vida por cada turno que el aldeano esté reparando este edificio.

Nombre:	Cuartel
Vida:	250
Costo:	50 oro
Crea unidades:	● Espadachin ● Arquero
Turnos en construirse:	3 turnos

Tamaño:	4 casilleros
Velocidad de reparación:	Recupera +50 de vida por cada turno que el aldeano esté reparando este edificio.

Nombre:	Castillo
Vida:	1000
Costo:	No se puede construir
Crea unidades:	• arma de asedio
Turnos en construirse:	No se puede construir
Tamaño:	8 casilleros
Velocidad de reparación:	Recupera +15 de vida por cada turno que el aldeano esté reparando este edificio.
Habilidades	El castillo ataca a **todas** las unidades que se encuentren en su rango de ataque. La distancia de ataque del castillo es <= 3 con daño = 20 tanto a unidades como edificios enemigos.

3.3. Jugabilidad

★ Es un juego por turnos. Hay dos jugadores.
★ El jugador que inicia es decidido al azar.
★ Cada jugador comienza con una plaza central y un castillo. No es posible construir un castillo. Sólo existirá el que es asignado al empezar la partida. Sí es posible construir plazas centrales y cuarteles sin límite.
★ Cada jugador comienza la partida con 3 aldeanos y 100 de oro
★ **Gana la partida el jugador que destruye el único castillo enemigo.**
★ Los jugadores inician en extremos opuestos del mapa.
★ Cada edificio puede ser construido / reparado por 1 solo aldeano a la vez. (no se pueden poner 3 aldeanos a construir / reparar un edificio)
★ Cada aldeano que un jugador tenga produce 20 unidades de oro por turno (por el solo hecho de "existir", no importa si está herido)
★ Cuando un aldeano está construyendo o reparando un edificio **no genera las 20 unidades de oro** en cada turno que esté reparando / construyendo.
★ Cuando un aldeano ya no puede reparar más un edificio porque el edificio alcanzó el 100% de vida, al turno siguiente el aldeano deja de reparar y **vuelve a producir las 20 unidades de oro.**
★ El límite de población es 50. Cada unidad del jugador ocupa 1 lugar de población. Los edificios no ocupan lugar de población.
★ Cada unidad ocupa 1 sola posición en el mapa
★ Hay una sola superficie sobre la cual moverse que es la tierra
★ Todas las unidades se mueven a igual velocidad, es decir se mueven 1 casillero por turno en la dirección elegida.
★ Cada unidad puede atacar a **UN** solo objetivo por turno. (por más que en su rango de ataque

haya más de 1 enemigo, ya sea este un edificio u otra unidad)

- ★ Las unidades no atacan solas, sino que se les debe indicar a qué quieren atacar.
- ★ El castillo en cambio, en cada turno, chequea todos los enemigos en su rango de ataque y los ataca a todos.
- ★ El tamaño del mapa será rectangular a definir por cada grupo.

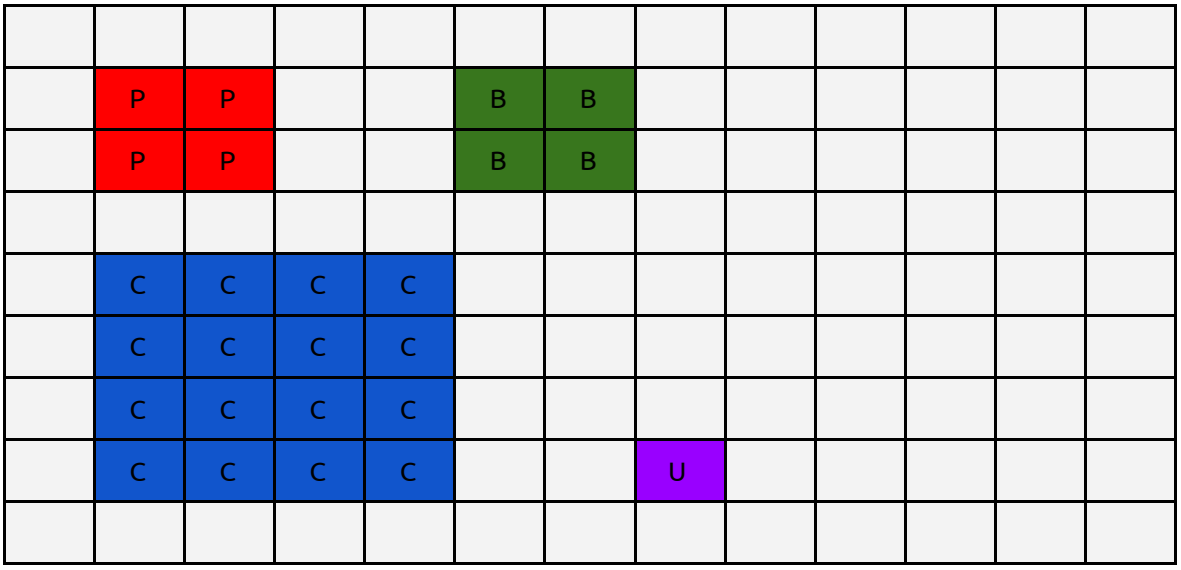
3.4. Distancia

La distancia de ataque se mide en casilleros. Por ejemplo, un **arma de asedio** que posee distancia de ataque menor o igual a 5 podrá atacar a cualquier edificio que se encuentre en cualquiera de los casilleros marcados con una O en el siguiente diagrama pero no a aquellos que estén en los casilleros marcados con una X:.

X	X	X	X	X	X	X	X	X	X	X	X	X
X	O	O	O	O	O	O	O	O	O	O	O	X
X	O	O	O	O	O	O	O	O	O	O	O	X
X	O	O	O	O	O	O	O	O	O	O	O	X
X	O	O	O	O	O	O	O	O	O	O	O	X
X	O	O	O	O	O	O	O	O	O	O	O	X
X	O	O	O	O	O	**ASD**	O	O	O	O	O	X
X	O	O	O	O	O	O	O	O	O	O	O	X
X	O	O	O	O	O	O	O	O	O	O	O	X
X	O	O	O	O	O	O	O	O	O	O	O	X
X	O	O	O	O	O	O	O	O	O	O	O	X
X	O	O	O	O	O	O	O	O	O	O	O	X
X	X	X	X	X	X	X	X	X	X	X	X	X

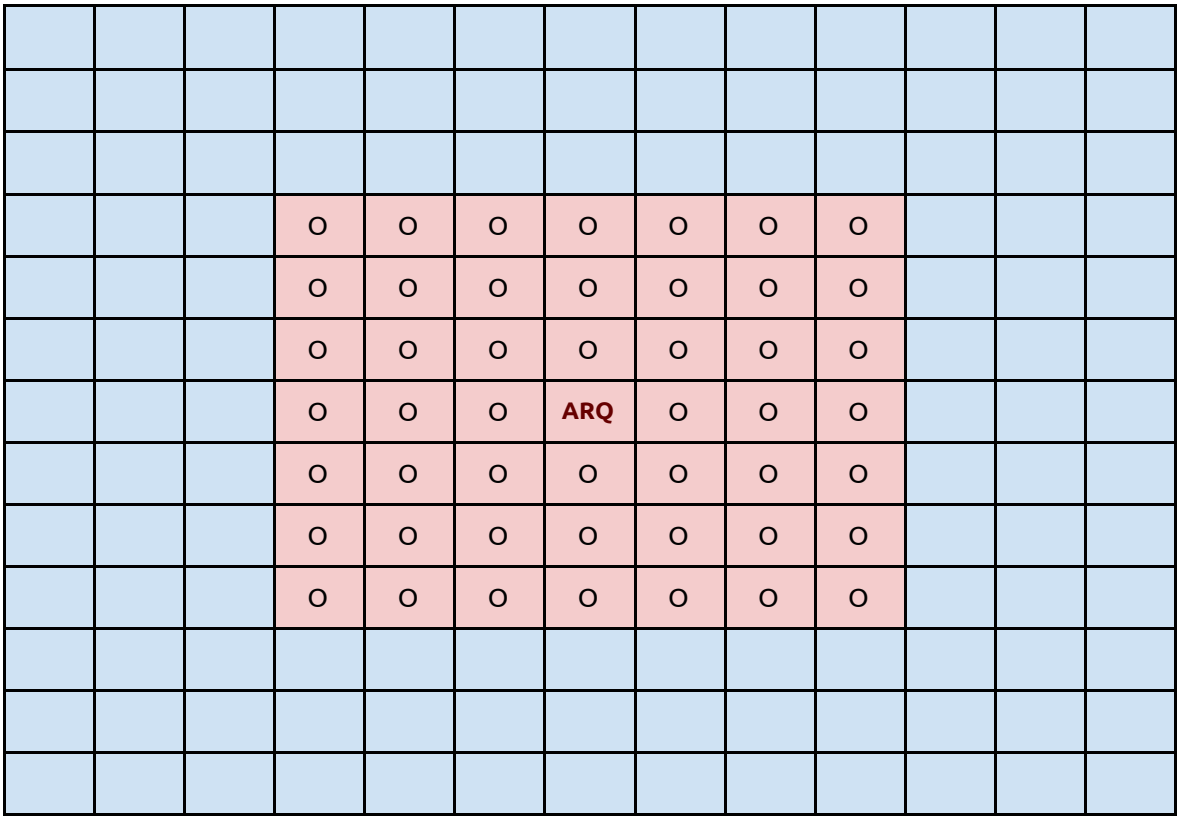
3.5. Tamaños de edificios y unidades

- [P] Plaza central
- [B] Cuartel
- [C] Castillo
- [U] Cualquier unidad (aldeano, arma de asedio, arquero, espadachín)



3.6. Distancia de ataque y superposición con edificios:

Contemplemos el caso de un **arquero** que tiene un alcance de ataque en los casilleros marcados con una O en el siguiente diagrama:



Ahora pongamos edificios y unidades en su rango de ataque:

										B2	B2	
		P	P							B2	B2	
		P	P	O	O	O	O	O	O			
			O	O	O	O	O	O	O			
			O	O	O	O	O	O	B1	B1		
			O	O	O	ARQ	O	O	B1	B1		
			O	O	O	O	O	O	O			
			O	O	O	O	O	O	O			
C	C	C	C	O	O	O	O	O	O			
C	C	C	C									
C	C	C	C								U	
C	C	C	C									

Notamos que el arquero podrá atacar a:

- **[P] Plaza central**
- **[B1] Cuartel 1**
- **[C] Castillo**

Pero NO podrá atacar a:

- **[U] Unidad enemiga**
- **[B2] Cuartel 2**

4. Interfaz gráfica

La interacción entre los usuarios 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 1.7 o superior que incluya JavaFX.
2. **JUnit**: Framework de pruebas unitarias para Java.

3. **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
4. **Apache Ant**: Se deberá incluir el archivo `build.xml` y todos los archivos adicionales necesarios (por ejemplo, `ivy.xml`) 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).
5. **Repositorio**: Todas las entregas deberán ser subidas a un repositorio único para todo el grupo en donde quedarán registrados los aportes de cada miembro. Se podrá elegir entre los siguientes:
    a. GitHub
    b. GitLab
    c. BitBucket
6. **Git**: Herramienta de control de versiones
7. **Herramienta de integración contínua**: 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. Gitlab CI

6. Entregables

Para cada entrega se deberá subir lo siguiente al repositorio:

1. Código fuente de la aplicación completa, incluyendo también: código de la pruebas, archivos de recursos
2. Script para compilación y ejecución (Ant)
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.

Aquél 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 (preferentemente impreso).*

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. Casos de prueba para cada entrega

Entrega 0 (8 de noviembre)

- Repositorio con la estructura básica del proyecto con los scripts de Ant y la herramienta de integración contínua configurada. Se podrá utilizar el proyecto base provisto por la cátedra como referencia.
- Solamente un miembro del grupo (cualquiera) deberá subir el enlace del repositorio mediante el botón “Agregar entrega” del campus virtual.

Entrega 1 (Semana del 12 de noviembre)

- Pruebas del Mapa
    - Tamaño
    - Colocar unidades y edificios
- Pruebas de unidades
    - Pruebas de movimiento y dirección (1 casillero por turno en las 8 posibles direcciones, siempre y cuando no intenten ir más allá del mapa)
        - aldeano
        - arquero
        - espadachín
        - arma de asedio
    - Pruebas de construcción
        - aldeano
            - verificar construcción de cuartel y plaza central
            - verificar que se haga en los turnos propios al jugador
            - verificar que no suma oro
    - Pruebas de reparación
        - aldeano
            - verificar reparación
            - verificar que se finalizada la reparación, sume oro
- Pruebas de edificios
    - Cuartel crea:
        - espadachín
        - arquero

  - Plaza central crea aldeano
  - castillo crea arma de asedio

Entrega 2 (Semana del 19 de noviembre)

- Distribución de los jugadores en el mapa
  - Pruebas de inicio del juego, posición, edificios, aldeanos y oro necesarios.
- Reglas de población
  - Crear unidades => sube la población
  - Matar unidades => baja población
  - Matar aldeanos => baja población y baja producción de oro
  - Verificar tope poblacional
- **Pruebas de distancia y ataques (tanto para las unidades como para el castillo)**

Entrega 3 (Semana del 26 de noviembre)

- Interfaz gráfica inicial básica: solicitud de jugador 1, jugador 2, comienzo de partida, disposición de jugadores en el tablero, botonera de acciones.
- Reglas de finalización de juego (modelo)

Entrega 4 - Final: (6 de diciembre)

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

Tiempo total de desarrollo del trabajo práctico: cinco 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.