Trabajo Práctico N⁰ 2

0) Objetivos

1. Afianzar los conocimientos de programación orientada a objetos.
2. Aplicar las estructuras de datos y los patrones de diseño vistos en clase.

1) Pautas de Trabajo y Evaluación

1.1) Tamaño de los grupos

De dos a cuatro estudiantes.

1.2) Fechas importantes

Entrega	Comisión	
	Martes y Jueves	Miércoles y Viernes
Parcial	Martes 29/05/2018	Miércoles 30/05/2018
Final	Martes 19/06/2018	Viernes 22/06/2018

En la entrega parcial se deberá entregar el diseño general de la solución. En caso de modificaciones en los grupos por favor completar el formulario indicado abajo con los integrantes de cada grupo. Recuerden usar nombres nuevos para los nuevos grupos.
Formulario:
https://docs.google.com/forms/d/1v8z-X28GFKO8hx4Zbvt8gO-3O3tvOGyXNT9f9ESBbbA/

1.3) Cómo entregar

El TP se deberá entregar en formato impreso (sólo el informe) y digital (tanto el informe como el código), hasta las 23:59:59 del plazo solicitado.

Si usan control de versiones, bastará con que estén los commits en su proyecto, dentro del grupo de GitLab de la cátedra.

`https://gitlab.com/groups/untref-ayp2-estudiantes-2018-s1/tp2-<Nombre-del-Grupo>`

Si no, se deberá enviar a algo-untref-rw-doc@googlegroups.com un archivo .zip con la carpeta del proyecto. El archivo deberá tener la forma:

`tp2-grupo-<nombre_del_grupo>-turno-<mar_jue o mie_vie>.zip`

Respetando mayúsculas y minúsculas.

(Ejemplo: `tp2-grupo-nombre-turno-mar_jue.zip`)

1.4) Estructura

Ya se encuentra publicada una plantilla con la estructura solicitada.

De todos modos, se aclaran algunas cosas que el proyecto deberá contener:

- Informe en un archivo README.md ubicado en el directorio raíz, el cual incluirá:
  - Nombres de los integrantes del grupo
  - Decisiones de diseño tomadas
  - Descripción de cada archivo *.java comprendido en solución del problema
  - Conclusiones
- Diagrama de clases
- Casos de prueba
- Código fuente
- Archivos necesarios para importar el proyecto (ej.: archivos .project, .classpath, etc.)

1.5) Revisiones semanales

Cada grupo tiene un docente responsable de su seguimiento. Recomendamos que coordinen revisiones al menos, una vez por semana.

1.5.1) Metodología recomendada

Para agilizar el tiempo de revisión (lo ideal es llevarlo a 10'), se les solicita que preparen cada reunión.

Luego, por favor tengan preparadas las respuestas a la siguientes preguntas. (Para ir viendo su propio progreso, vayan agregando cada una de las respuestas en una tabla al README del proyecto.)

- ¿Qué hicieron desde la última revisión?
- ¿Qué problemas tuvieron? ¿Cómo los solucionaron?
- ¿Qué aprendieron?
- ¿Qué piensan hacer hasta la última revisión? (Para este punto pueden usar "issues" y "board" de GitLab.)

1.6) Evaluación

Si bien la idea es minimizar comentarios redundantes dentro del código, recuerden que esta es una instancia de evaluación: aclaren todo lo que consideren importante, aún lo que consideren trivial.

1.6.1) Pasos

I) Parte automática.

El proyecto debería ser capaz de:

1. Compilar

Ej: Que la tarea "`gradle`[1] `build`*" sea ejecutada con éxito*

2. Ejecutar y Pasar los tests (incluye el paso anterior)
   Ej: Que la tarea "`gradle test`*" sea ejecutada con éxito*
3. Ejecutarse
   Ej: Que la tarea "`gradle run`*" sea ejecutada con éxito*

II) Parte manual

Esta parte se realiza sólo si la anterior ha sido completada satisfactoriamente.

1. Evaluación Funcional y No-Funcional del Código Fuente.
2. Evaluación del Informe Final.

2) Ejercicio: Batallas Romanas

El objetivo del Trabajo Práctico es diseñar e implementar un juego de estrategia por turnos para 2 jugadores.

2.1) Dinámica General

El juego es de batallas entre ejércitos, para dos jugadores. Toda la mecánica del juego está organizada por turnos.

Cada ejército de cada jugador se compone de una o más legiones, o de soldados sueltos. A su vez, cada legión tiene un nombre y puede estar compuesta por una cantidad X de soldados sueltos de cada tipo *(ver tipos de soldados en sección 2.2.1)* u otras legiones.

Por ejemplo, la legión "Divide et impera" está compuesta por 1000 auxiliares, 2000 legionarios y 50 centuriones; mientras que el ejército "SPQR" está compuesto por las legiones "I" (que cuenta con 500 legionarios y 3 centuriones), "II" (que cuenta con 100 auxiliares), junto a 30 legionarios sueltos (sobrevivientes de una batalla de las guerras púnicas).

Almacenados en disco, existen dos archivos con legiones pre-construidas. Cada archivo está formateado con uno de los dos formatos soportados. *(Ver detalles acerca de cada uno de los formatos en la sección 2.2.1.1.)*

Existen varias fases en el juego:

1. Los jugadores tiran un dado para elegir quién comienza a armar su ejército en primer lugar.
2. Siguiendo el orden indicado por los dados (el mayor primero), los jugadores eligen, de una lista de legiones ya preconstruidas y soldados sueltos, la combinación que constituirá cada uno de sus ejércitos. Cada legión tiene un costo asociado (que es la suma de los costos de cada uno de sus soldados). Cada ejército tiene un costo asociado (que es la suma de los costos de cada una de sus legiones, más el de sus soldados sueltos). Cada jugador cuenta con 500.000 puntos para armar su ejército.

[1] Asumiendo que Gradle es la herramienta de administración de proyectos que usen. Otras herramientas tienen comandos similares.

Para que cada jugador pueda elegir cómo armar su ejército, el sistema debe desplegar una lista de las legiones indicando su nombre, y el total de soldados de cada tipo que posee cada legión. (En el caso de legiones compuestas por otras legiones, el sistema debe ser capaz de mostrar en forma transparente las cantidades totales de cada tipo de soldado). También debe mostrar el costo total de cada legión, el costo de comprar soldados individuales y la cantidad de dinero restante para realizar las compras.

El jugador puede elegir algunos de los ítems listados (legiones prearmadas, auxiliares, legionarios y centuriones) y la cantidad a comprar.
Si el dinero con el que cuenta el jugador es suficiente entonces se incorpora a su ejército. En caso contrario se debe rechazar la compra con un mensaje acorde.

No es preciso que el jugador gaste todo su dinero, sino que en cualquier momento puede terminar con la formación de su ejército.

3. Comienza la batalla, organizada por turnos. Por una cuestión de equilibrio, el jugador que ataca primero será quien empezó en segundo lugar a armar su ejército. En cada turno cada jugador ataca (una vez) al ejército rival. En el turno siguiente, el rival responde de la misma manera. Pierde el ejército que primero se queda sin soldados. *(Ver Asignación de Daño en la Sección 2.2.2.).*

   Si el jugador en acción (atacante) todavía cuenta con dinero, el sistema le debe dar la opción de comprar más soldados y/o legiones prearmadas.

2.2) Especificidades

2.2.1) Soldados

En una legión romana hay tres tipos de soldados: auxiliares, legionarios y centuriones. Todos los soldados comienzan con 100 puntos de vida. Si los puntos de energía de un soldado bajan a 0 (o menos), muere.

Característica	Auxiliares	Legionarios	Centuriones
Armas	Jabalinas	Espada corta, escudo	Espada corta, escudo
Daño	0.7	1.4	1
Habilidades especiales	Tiene una chance de acertar uno de cada dos ataques	Tiene una chance de acertar todos los ataques realizados.	Tiene una chance de esquivar uno de cada dos ataques

	realizados.		recibidos. Incrementa el ataque del ejército al que pertenece en un 10% (acumulativo con otros centuriones del mismo ejército).
Orden en ser atacados	1	2	3
Costo	50	100	200

2.2.1.1) Almacenamiento en disco de las legiones

El juego ya posee varios tipos de legiones prearmados, almacenados en disco. Por cuestiones operativas, existen dos tipos de archivos: el formato "FC" ("FormatoComa") y el FPC ("FormatoPuntoYComa"). Cada uno utiliza una variante de un formato de texto plano separado por comas (CSV)[2], y toman sus nombres del tipo de separador que utilizan.

El programa debe ser capaz de importar un archivo de legión sin importar en qué formato se encuentre (FC o FPC) de forma transparente al usuario. Además, debe ser capaz de convertir un archivo de un formato al otro.

Formato de cada línea del archivo

Cada línea describe una legión. El primer elemento de la línea es el nombre de la legión, a continuación le siguen nombres de otras legiones ya formadas, y por último hasta tres números enteros, donde el primero representa la cantidad de auxiliares, el segundo de legendarios y el tercero de centuriones.[3]

Ejemplo de legiones almacenadas en disco, en ambos formatos:

---

[2]Una opción para implementar el parseo de archivos en formato CSV es utilizando una o más de las clase propias de Java "BufferedReader" y/o "StringTokenizer". Otra opción es utilizar la clase CSVParser de la biblioteca externa ApacheCommons.

[3] Una opción para validar las líneas de los archivos en formato CSV es utilizar Expresiones Regulares (regexps, de "regular expressions"). Las regexps simplemente son una manera de definir patrones que uno debe cumplir.
De este modo, la siguiente expresión describe cualquier número celular de ámbito del AMBA (CABA y GBA), en formato internacional: *"Un signo más, 54 (código de país), un 9 (celular), un 11 (código de área) y dos secuencias de cuatro dígitos; todo separado por guiones opcionales."*
Como regexp, se expresa así: `\+54-?9-?11-?\d{4}-?\d{4}`
Ejemplos que corresponden a este patrón: `+54-9-11-1234-5678`, `+5491112345678`
Pueden probar regexps online (incluido el patrón dado arriba) en https://www.regexpal.com/
Tutorial oficial de regexps en Java: https://docs.oracle.com/javase/tutorial/essential/regex/index.html

Archivo “Legiones.FC”	Archivo “Legiones.FPC”
“CasiTodosAuxiliares”, 388, 7, 1 **“PurosLegionarios”, 0, 1000, 0** **“Guardia Pretoriana de César”, 0, 0, 88** **“Super Guardia”, “Guardia Pretoriana de César”, “CasiTodosAuxiliares”** **“Supreme Praetorian Guard”, “Super Guardia”, 10**	**“CasiTodosLegionarios”; 10; 200; 1** **“PurosAuxiliares”; 1000; 0; 0** **“Guardia Pretoriana de Adriano”; 0; 0; 222** **“Ultimate Praetorian Guard”; “Guardia Pretoriana de César”; 0; 0; 1000**

En formato FC

“CasiTodosAuxiliares”, 388, 7, 1
El nombre de la legión es “CasiTodosAuxiliares” y está compuesta por 388 auxiliares, 7 legionarios y 1 centurión

“PurosLegionarios”, 0, 1000
El nombre de la legión es “PurosLegionarios” y está compuesta solamente por 1000 legionarios

“Guardia Pretoriana de César”, 0, 0, 88
El nombre de la legión es “Guardia Pretoriana de César” y está compuesta por 88 centuriones.

“Super Guardia”, “Guardia Pretoriana de César”, “CasiTodosAuxiliares”
El nombre de la legión es “Super Guardia” y está compuesta por dos legiones, la “Guardia Pretoriana de César” y “CasiTodosAuxiliares”.

“Supreme Praetorian Guard”, “Super Guardia”, 10
El nombre de la legión es “Supreme Praetorian Guard”, y está compuesta por la legión “Super Guardia” y 10 auxiliares.

En formato FPC

“CasiTodosLegionarios”; 10; 200; 1
El nombre de la legión es “CasiTodosLegionarios”, y está compuesta por 10 auxiliares, 200 legionarios y 1 centurión.

“PurosAuxiliares”; 1000; 0; 0
El nombre de la legión es “PurosAuxiliares”, y está compuesta por 1000 auxiliares.

“Guardia Pretoriana de Adriano”; 0; 0; 222
El nombre de la legión es “Guardia Pretoriana de Adriano”, y está compuesta por 222 centuriones

“Ultimate Praetorian Guard”; “Guardia Pretoriana de César”; 0; 0; 1000
El nombre de la legión es “Ultimate Praetorian Guard”, y está compuesta por la legión “Guardia Pretoriana de César” y 1000 centuriones

2.2.1.2) Almacenamiento en memoria de los ejércitos

Los ejércitos se almacenan en memoria como un Composite.

2.2.2) Asignación de Daño

La asignación de daño está simplificada. Al momento de calcular el daño que un ejército hace a otro, se calcula la sumatoria del daño que realiza cada uno de los soldados que lo componen (estén en una legión o no). Se juntan los puntos de daño atacantes y se van “gastando” en las unidades del ejército defensor, siguiendo la siguiente secuencia: primero a los auxiliares, luego a los legionarios y finalmente a los centuriones.

2.2.3) Menú por consola

Se deberá proveer un menú por consola mínimo para poder ejecutar el juego. *(Lógicamente, sólo puede existir un menú a la vez.)*