|  |  |  |
| --- | --- | --- |
|  | Universidad Finis Terrae  Facultad de Ingeniería | Prof. Darío Rojas  2023–1 |

**Proyecto – Programación Avanzada**

|  |  |
| --- | --- |
| Resultados de Aprendizajes a Evaluar | * Resuelve problemas de regular complejidad implementado programas con paradigma Orientada a Objetos * Diseña y Documenta Software mediante Diagrama UML |
| Modalidad | Online y Grupal |
| Tiempo estimado de trabajo | 16 horas durante 5 semanas, 3-4 horas a la semana aproximadamente. |
| Fecha de Entrega | Jueves 29 junio 2023 a las 23:59hrs. – Presentación se entrega por Vídeo (ver detalles más abajo). |
| Forma de Entrega | Archivo comprimido con programa Python y recursos por plataforma eFinis: Programación Avanzada > Tareas > PROYECTO  El proyecto contempla una presentación grupal con interrogación individual |
| Instrucciones Generales | * El proyecto es grupal – 3 integrantes máximo, 2 integrantes mínimo * Puede utilizar cualquier material escrito o electrónico (no copy&paste) * Es responsabilidad del/la estudiante ser responsable de entregar el archivo correcto o cualquier acción que deba hacer para asegurar la entrega correcta de los entregables y respuestas solicitadas en las sección y formato en plataforma eFinis. Errores de entrega son calificados con un 1. * **DEBE ENTREGAR TODOS LOS ARCHIVOS EN UN SOLO ARCHIVO COMPRIMIDO .zip** * Por cada día de atraso el descuento es de 1 punto (cada 24 horas desde fecha y hora del cierre de la evaluación) * Tenga en cuenta los “CRITERIOS DE EVALUACIÓN”. * **La copia o la facilitación de respuestas entre compañeros es sancionada con un 1.** * **NOTA = Suma de puntaje + 1** |

|  |
| --- |
| **PROYECTO SIMULACIÓN POKEMON** |

Para este proyecto se solicita la implementación de una simulación del juego de cartas Pokemon. Donde dos jugadores se enfrentan utilizando cartas que se atacan entre ellas. Una explicación del juego original se puede encontrar en <https://www.youtube.com/watch?v=QO9Vx1M9smw>

**La funcionalidad básica y mínima requerida para el juego incluye:**

* Cada jugador tiene 20 cartas, las que incluyen pokemon y energías (no es necesario cartas entrenador). Es suficiente 8 pokemon en total y 12 cartas de energía. Al inicio se le reparten 4 cartas de su propio mazo a cada jugador. El resto se mantiene oculta (el jugador no las conoce)
* Las cartas son barajadas (ordenadas) aleatoriamente para cada jugador al comienzo del juego.
* Siempre comienza el jugador 1.
* Las acciones posibles del jugador son en orden: 1) saca una carta del mazo, 2) colocar un pokemon como parte del equipo, 3) colocar una energía, 4) atacar. (todas las acciones son opcionales)
* Un jugador solo puede tener hasta 3 pokemones listos para pelear y uno solo en la arena de batalla (pelando)
* Cada carta pokemon tiene nombre, descripción, puntos de vida y ataque (incluido el daño que hace el ataque).
* Al atacar, el pokemon que ataca suma la misma cantidad en indicada en su ataque como daño al oponente.
* Si un pokemon recibe igual o más daño que sus puntos de vida, entonces es descartado él y todas sus energías.
* El juego termina cuando un jugador ya no tiene más pokemon para pelear, gana el jugador con más pokemos sin descartar.

**Aspectos de Diseño:**

* Todos los datos de mazos, cartas, etc son leídos desde archivos o planillas.
* El juego puede ser gráfico o modo texto.
* Los Mazos y Cartas deben diseñarse y programarse con P.O.O
* Debe documentar mediante un diagrama de clases el software completo
* Puede realizar una implementación como simulación pura (no interviene el usuario y las acciones se generan al azar) o puede incluir interacción con el usuario y dos jugadores.
* Al finalizar el juego debe indicar cuál es el pokemon más ganador, el pokemon más perdedor y el ataque con más puntos realizado en toda la partida.
* Debe incluir, por lo menos, un contendor con lista enlazada en su código.

**Aspectos administrativos:** *Condiciones que deben cumplirse para poder acceder a la revisión del proyecto. No se revisarán programas que no cumplan las restricciones.*

* El programa debe ser un único archivo .py
* Debe incluir todos los archivos de datos, imágenes y sonidos utilizados.
* Debe incluir el diagrama UML de su software.
* Debe incluir un documento con instrucciones para iniciar el juego y reglas de su juego.
* Programas que no puedan ejecutarse (cualquier motivo) no serán revisados

**Aspectos de Presentación:**

La presentación debe cumplir la siguiente pauta:

|  |  |
| --- | --- |
| **Contenido** | **Ptje**. |
| En su entrega de código (archivo zip) debe incluir en el manual el link para el vídeo de la presentación en plataforma **Youtube** que debe ser de libre acceso con el link. | REQUISITO |
| El vídeo dura entre 10 y 15 minutos | REQUISITO |
| **3 minutos** para presentar el funcionamiento del videojuego, es decir, una o dos personas jugando. El video comienza con la presentación del grupo y el nombre del juego. Esta parte la puede hacer cualquier miembro del grupo. | 6 |
| **6 minutos o más** para presentar una explicación de las principales partes del código, aquí deben estar todos los participantes del grupo. Al comenzar, cada integrante debe indicar su nombre. | 8 |
| El programa debe finalizar con **1 minuto** indicando que fue los más importante aprendido en el desarrollo del proyecto (puede hacerlo cualquier integrante del grupo) | 1 |

|  |
| --- |
| **CRITERIOS DE EVALUACIÓN GENERAL** |

Siga la siguiente pauta de evaluación para cumplir con los requerimientos

|  |  |
| --- | --- |
| **Criterio** | **Ptje** |
| Utiliza, en forma correcta, todos los elementos de programación P.O.O: clases, constructores, instancias, etc. | 20 |
| El juego cumple totalmente con la funcionalidad requerida | 60 |
| El juego incluye instrucciones correctas y completas que permiten ejecutar y jugar el juego. | 5 |
| En presentación, demuestra dominio del programa **(ver detalles más arriba)** | 15 |