**PROYECTO FINAL**

Alejandro Parra

Nicolas Penagos

Marzo 2019.

Universidad Icesi.

Algoritmos y programación 2.

**Listado de Requerimientos:**

|  |  |
| --- | --- |
|  | R-1 Elegir Personaje |
| Resumen | Permite al usuario elegir entre dos personajes al inicio del juego. Este será el personaje con el que el usuario jugará. |
| Entradas |  |
| Resultado | El jugador controla al personaje escogido éxitosamente. |

|  |  |
| --- | --- |
|  | R-2 Mover personaje |
| Resumen | Permite al usuario mover al personaje en las 3 direcciones básicas (arriba, izquierda, derecha) por medio de las teclas. |
| Entradas | Evento donde presiona la tecla |
| Resultado | Se mueve el personaje acorde a la dirección que elige el usuario. |

|  |  |
| --- | --- |
|  | R-3 Destruir edificio |
| Resumen | Cuando el personaje golpea al edificio, este empieza a destruirse hasta derrumbarse. |
| Entradas | Coordenadas de ambos objetos que entran en colisión |
| Resultado | El edificio va desaparece en las coordenadas de colisión. |

|  |  |
| --- | --- |
|  | R-4 Enemigos |
| Resumen | El programa debe permitir la generación de enemigos. |
| Entradas |  |
| Resultado | Los enemigos son generados y entran en acción con la escena. |

|  |  |
| --- | --- |
|  | R-5 Llevar estadísticas de juego |
| Resumen | El programa debe permitir medir y llevar las estadísticas de la partida tales como: tiempo, número de golpes y puntaje. |
| Entradas |  |
| Resultado | Las estadísticas son tomadas correctamente y almacenadas. |

|  |  |
| --- | --- |
|  | R-6 Guardar las estadísticas del jugador |
| Resumen | El programa debe permitir guardar las estadísticas del jugador de manera segura para que estas no puedan ser modificadas. |
| Entradas |  |
| Resultado | El archivo es guardado de forma segura y correcta. |

|  |  |
| --- | --- |
|  | R-7 Mostrar estadísticas |
| Resumen | El programa debe mostrar las estadísticas de todos los jugadores que las guardaron de manera ordenada y efectiva. |
| Entradas |  |
| Resultado | Todos los registros son mostrados correctamente en pantalla. |

|  |  |
| --- | --- |
|  | R-8 Registrar nickname de jugador |
| Resumen | El programa debe permitir al usuario digitar el nickname que este desee. |
| Entradas | Nickname |
| Resultado | El nickname es definido, almacenado y mostrado correctamente en pantalla. |

|  |  |
| --- | --- |
|  | R-9 El Ordenar estadísticas |
| Resumen | El programa debe permitir ordenar las estadísticas por medio de diferentes criterios como: nickname, puntaje, golpes, tiempo y fecha. |
| Entradas | Nickname |
| Resultado | Los registros son ordenados de forma correcta. |

|  |  |
| --- | --- |
|  | R-10 Buscar jugador |
| Resumen | El programa debe permitir buscar un jugador por nickname o por posición en el ranking. |
| Entradas | Nickname, posición |
| Resultado | El jugador es encontrado y mostrado en pantalla. |

|  |  |
| --- | --- |
|  | R-11 Generar top 5 |
| Resumen | El programa debe generar un top 5 con los mejores jugadores de todos los registrados. |
| Entradas |  |
| Resultado | Los jugadores son seleccionados y mostrados en pantalla. |

|  |  |
| --- | --- |
|  | R-12 Calcular suma puntajes de top 5 |
| Resumen | El programa debe calcular la suma de los puntaje de los mejores jugadores del top 5 |
| Entradas |  |
| Resultado | Se calcula la suma de los puntajes y se muestra en pantalla. |

|  |  |
| --- | --- |
|  | R-13 Buscar primer jugador palíndromo |
| Resumen | El programa debe buscar si en la lista de jugadores se encuentra un jugador con nickname palíndromo, en caso de encontrarlo este debe ser mostrado en pantalla. |
| Entradas |  |
| Resultado | El nombre palíndromo es mostrado en pantalla. |

No funcionales:

|  |  |
| --- | --- |
| Interfaz JavaFx  Algoritmos de ordenamiento | Implementar el programa de computador con interfaz gráfica en JavaFx.  -Escritura y lectura de archivos planos.  -Escrituta y lectura con serialización.  -al menos 4 excepciones del api.  -3 excepciones personalizadas.  -algoritmos de ordenamiento (al menos dos criterios diferentes por cada método).  - implementaciones de búsqueda binaria (con diferente criterio).  -Dos listas enlazadas.  -4 herencias y 3 desacoplamiento a través de interfaces.  -Dos árboles binarios.  -4 métodos recursivos.  -dibujo con al menos 5 primitivas.  Al menos tres hilos adicionales al principal. |