Universidad de San Carlo de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Introducción a la programación y computación 1
Ing. Moisés Velásquez
Auxiliar. Pablo Oliva

PRACTICA No.1: MANUAL TECNICO

Guillermo Enrique Marroquin Morán

202103527

Guatemala 15 de ago. de 25

INDICE

Manual Técnico

Requerimientos mínimos del programa.

Para instalar apache NetBeans los requisitos mínimos son:

- 781 MB de Espacio Libre en el Disco Duro.
- 512 MB de RAM.
- Procesador Intel Pentium III a 800 MHz.
- Compatible con Windows, macOS y Linux.

(Uanl, 2022)

Requisitos del ordenador usado para la práctica:

Explicación de los módulos usados en la practica

Librerías usadas

```
//Autor Kique Marroquin y williams marroquin (mi viejo)
package practica1;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Random;
import java.util.Scanner;
public class Practica1 {
```

- La librería “java.time.LocalDateTime” permite trabajar con la hora y fecha del ordenador sin incluir la zona horaria, por ejemplo 15-08-2025, 19:55:30.
- La librería “java.time.format.DateTimeFormatter;” nos sire para dar un formato a una fecha y hora y ponerla como más me convenga (dd/mm/yyyy)/(mm/dd/yyyy) y (hh/mm/ss), para que al imprimir la hora y fecha den como resultado: 15/08/2025 19:58:25.
- La librería “random” la usamos para que nos genere números aleaorios (contando de 0 al número indicado) en la práctica se usó con el valor de 2, que solo genere 0 o 1 para la victoria de un personaje u otro.
- La librería “Scanner” permite leer datos, que escribamos con el teclado en la consola, sea los nombres, las armas, los niveles de poder y las habilidades.

Módulos usados.

Modulo personaje

```
    // el modulo de buscaer personajes, este se usa en las opciones 2,3,4,5,6 y 7, waw que monton
public static int buscarPer (String[][] personajes, int cantidad, String nombre) {
    for (int i = 0; i < cantidad; i++) {
        if$ (personajes[i][0] != null && personajes[i][0].equalsIgnoreCase(nombre)) {
            return i;
        }
    }
    return -1; // esto indica que si el personaje no se encuentra va a retornar un -1
}
```

El módulo de buscar personaje lo que hace en términos generales, en la matriz creada el “string [][]personajes” permite buscar por posiciones gracias al ciclo “for” que va desde el valor 0 hasta valor de personajes registrados, y si en dado caso al hacer llamado del módulo no se encuentra el nombre se retornara a -1 por lo que el personaje no ha de existir.

Módulo de Buscar personajes.

```
  //el modulo de mostrar personaje, se usa en las opciones 2,4,5 y posiblemente 7
public static void mostrarPer (String [][] personajes, int Per){
   System.out .println("Personaje");
   System.out .println("nombre:"+ personajes[Per][0]);
   System.out .println("arma:"+ personajes[Per][1]);
   System.out .println("nivel de poder:"+ personajes[Per][2]);
   System.out .println("habilidades:");
   for (int j =3; j<8; j++){
      System.out .println("nombre:"+ personajes[Per][j]);
   }
}
```

Este método, nos permite mostrar en pantalla los personajes y sus atributos, que ya este guardado en la matriz que esta como “una memoria”, esto representado como el (String [][] personajes, int Per), donde el string es la matriz y el “int per” indica la posición del personaje, imprime como una lista los atributos del personaje representado como una posición [0], [1], [2],[3], [j+3], en la línea de las habilidades el j+3 representa un bucle que desde la posición 3 ira la primera habilidad hasta la posición 7.

Partes del Código por explicar

```
public static void main (String[] args) {
   Scanner scanner = new Scanner(System.in );

   // Declaramos las variables
   int escribir;
 String  [][] personajes = new String [25][8]; // Matriz de personajes
 int cantidadPer = 0; //esto es el contador de personajes registrados

 String [] historialCo = new String [50]; //para guardar en matriz hasta 50 madrazos
 int cantPeleas = 0;
 String continuar;

 Random random = new Random();

 do{
     // Deplegar el  Menu principal
     System.out .println("Menu principal");
      System.out .println("1. Agregar Personaje");
      System.out .println("2. Modificar Personaje ");
      System.out .println("3. Eliminar Personaje");
      System.out .println("4. Ver Datos de un Personaje");
      System.out .println("5. Ver Listado de Personaje");
      System.out .println("6. Realizar pelea entre Personajes");
      System.out .println("7. Ver Historial de peleas");
      System.out .println("8. Ver datos de Estudiante");
      System.out .println("9. Salir");
      System.out .print("Elige una opción: ");
```

El método principal nos encontramos con la declaración una matriz de 25 columnas*8 filas para los personajes, y su debido contador llamado como cantidad de personajes, y otra matriz de 50 que es solo para los combates entre personajes, y su debido contador, vemos también una declaración de una variable random para los combates, y vemos el menú principal con sus debidas opciones hecho con un bucle do-while esto para optimizar, y no solo poner if´s.

```
escribir = scanner.nextInt();
//Ejecutar segun las opciones que se ponga en la consola, pudo haber sido if´s, pero es mas practico switch
switch (escribir) {
        case 1:
           do{

            if (cantidadPer >=25){
               System.out .println("Ya llegaste al maximo numero de personajes registrados, elimina alguno de ellos");
               break;
            }
            scanner.nextLine();
           System.out .println("Registro de un nuevo personaje");

           //verificamos si el nombre esta repetido o no
           String nombre;
           boolean Nrepetido = false;
           do{

             System.out .println("Ingrese el nombre del personje");
             nombre =scanner.nextLine();
             for(int i = 0; i<cantidadPer; i++){
                //el equals es para "ignorar" maysuculas y minusculas del nombre en cuestion
                if(personajes [i][0] != null && personajes[i][0].equalsIgnoreCase(nombre)){
                   Nrepetido = true;
                   System.out .println("Ese personaje ya existe, inventa otro nombre");
                    break;
                }
             }
           } while (Nrepetido);

           System.out .println("Ingresa el arma de tu preferencia");
           String arma = scanner.nextLine();

           int nivel_pod=0;
          System.out .println("Ingresa el nivel de poder (de 0 a 100)");
```

Aca miramos la variable “escribir” para escribir el número de opción para escoger, para cada opción unamos un Switch-case para optimizar, podemos ver una condición para cuando se hayan registrado más de 25 personajes, y para cuando haya un nombre repetido, esto lo hacemos con un valor booleano, que inicialmente es falso hasta que en la segunda condición se encuentre un nombre repetido (sin importar las mayúsculas o minúsculas), devuelva un valor verdadero para imprimir en pantalla que ese nombre ya está en uso, también podemos ver los apartados para agregar un nivel de poder y un arma para el personaje en cuestión.

```
            // Ingreso de las 5 habilidades
        String [] habilidades = new String [5];// esta linea me trajo problemas y muchos
        for (int i = 0; i <5; i ++){
            System.out .print("Habilidad #" + (i+1) + ": ");
           habilidades[i] = scanner.nextLine();
               }
    //Se van a guardar los personajes en una matriz de 25*8
    personajes[cantidadPer][0] =nombre;
    personajes [cantidadPer][1] = arma;
    personajes[cantidadPer][2] =String.valueOf (nivel_pod);
    for (int i =0; i<5; i++){
       personajes [cantidadPer][3+i] = habilidades[i];
    }
    cantidadPer++;

    // pregunta si quieres continuar
    System.out .print("¿Deseas ingresar otro personaje? (s/n): ");
    continuar = scanner.nextLine().toLowerCase();
          } while (continuar.equals("s"));
              break;
```

Para el ingreso de habilidades, se creó un arreglo de máximo 5, al hacer un bucle que re repita de 0 a 4 veces para ingresar el número de habilidad, y en las siguientes líneas podemos ver el guardado de los personajes, en la matriz de personajes, y el nivel de poder de ser un entero a ser solo texto, con las habilidades que va desde la posición 3 a la 7, al preguntar si se quiere continuar se hizo un bucle que re repite indefinidamente hasta elegir n, o superar los 25 personajes. Y su respectivo break, para que al terminar devuelva al menú principal

```
        case 2:
            System.out .println("Modificar atributos del Personaje" );
            scanner.nextLine();

            System.out .println("Ingresa el nombre del personaje a quien quieras modificar");
            String nombreMod = scanner.nextLine();
            int Mod = buscarPer (personajes, cantidadPer, nombreMod);//llamado del modulo de buscar personajes

            if (Mod ==-1){// eso es porque si al buscar el personaje, el modulo retornara un -1 y eso en una matriz no existe
               System.out .println("Ese personaje no existe, pon otro nombre");
break;
            }
            else { // se mostraran los datos iniciales, igual como en la opcion 4 y 5
               System.out .println("Estos son los datos actuales del personaje");
               mostrarPer (personajes, Mod); // llamado del modulo de mostrar personajes

               System.out .println("Ingresa un nuevo nombre");
               personajes[Mod][0] = scanner.nextLine();

               System.out .println("Elige un arma nueva");
               personajes[Mod][1] = scanner.nextLine();

              int nuevoNiv;
                 System.out .println("Ingresa un nuevo nivel de poder");
                 nuevoNiv = scanner.nextInt();
                 scanner.nextLine();

              for (int i = 0; i <5; i++ ){
                 System.out .println("Nueva habilidad #"+ (i+1) +":");
                 personajes[Mod][3+i]= scanner.nextLine();
              } break;
            }
```

Para la modificación de atributos, se usa el método de buscar personajes, primero para ver si el personaje existe en la matriz, y si en dado caso esto no pasa retorna -1 que indica que dicho personaje no existe, antes de modificar al personajes se hace llamado del método de mostrar personajes, que nos indica los datos actuales del personaje, luego llenamos los nuevos datos, igual que en la opción 1 las habilidades van de 0 a 4 veces para llenar nuevas habilidades, hasta llegar a la posición 7 de la matriz.

```
            case 3:
                System.out .println("Elimina un personaje");
                scanner.nextLine();
            System.out .println("Pon el nombre del personaje que quieras eliminar");
            String nombreEli = scanner.nextLine();
            int Eli = buscarPer (personajes,  cantidadPer, nombreEli);
            if (Eli == -1 ){ //esto es por si en la matriz no sale el nombre se vaya a la posicion -1
            System.out .println("Ese personaje no existe, pon otro nombre");
             }
             personajes [cantidadPer -1] = new String[8]; // se borra de la memoria la columna del personaje
             cantidadPer--;
            System.out .println("El personaje se ha eliminado");

            break;
                case 4:
                    System.out .println("Ver atributos del personaje :");
                    scanner.nextLine();

                    System.out .println("ingresa el nombre del personaje del que quieras ver sus atributos");
                    String nombreBus = scanner.nextLine();

                    int Buscar = buscarPer (personajes, cantidadPer, nombreBus);
                    if (Buscar ==-1){
                    System.out .println("No se encontro al personaje, deberias intentar con otro nombre");
                } else {
                        mostrarPer (personajes, Buscar);
                        }
                    break;
```

Para la opción 3, se hace llamado del método buscar personajes esto para eliminar un personaje de la matriz de personajes, pero si en dado caso nos retorna -1, nos indicara que el personaje no existe, al eliminar el personaje lo que hace es borrar la columna donde se encuentre el nombre de personaje que quisimos eliminar, para la opción 4, se hace un llamado del método de buscar personajes, para encontrar al personaje por su nombre, al dar con el nombre nos mostrará una lista con sus atributos, y si en dado caso no lo encuentra y nos retorna -1 el método de buscar personajes, se indicara que ese personaje no existe.

```
        //Hacemos un llamado del modulo de mostar personajes y del modulo de buscar
    case 5:
        System.out .println("Listado de de personajes");
       if(cantidadPer == 0){
         System.out .println("No hay ningun personaje registrado... de momento");
       } else{
         for (int i = 0; i <cantidadPer; i++){
            mostrarPer (personajes, i); // llamado del modulo de mostar personajes
            System.out .println();
            }
         }
        break;
```

En la opción 5 hacemos un llamado del método de mostrar personajes, si en dado caso la cantidad de personajes es 0 es porque no se ha registrado ningún personaje, y si hay personajes registrados, con un bucle “for”, imprimirá los atributos de los personajes que va 0 a la cantidad de personajes que hayamos registrado.

```
        case 6:
            System.out .println("¡Realizar combate entre personajes!");
            scanner.nextLine();

            System.out .println("Ingresa el nombre del primer peleador");
            String per1 = scanner.nextLine();
            int ID1 = buscarPer (personajes, cantidadPer, per1);

            System.out .println("Ingresa al segundo peleador");
            String per2 = scanner.nextLine();
            int ID2 = buscarPer (personajes, cantidadPer, per2);

            if(ID1==-1 || ID2==-1){
               System.out .println("Un personaje o los dos no existen, todavia");
            }else{
             // Usare la libreria random porque quiero y puedo
             int ganador = random.nextInt(2); //0 para un personaje 1 para el otro
             String resultado;
            if (ganador ==0){
               resultado = personajes [ID1][0] + "pierdes contra" + personajes[ID2][0];
            } else{
               resultado = personajes [ID2][0] + "pierdes contra" + personajes[ID1][0];
            }
            String fecha_hora = LocalDateTime.now ().format(DateTimeFormatter.ofPattern ("dd/MM/yyyy HH:mm:ss"));
            String Registrar = fecha_hora +"---"+ resultado;

            if (cantPeleas < historialCo.length){
               historialCo[cantPeleas] = Registrar;
               cantPeleas++;
            }
            System.out .println("Combate registrado:"+ Registrar);
             }
```

Para la opción 6 hacemos llamado del método de buscar personaje, para ver si los personajes que estamos solicitando existen, o no, si en dado caso nos retorna -1 el método nos indicara que uno o dos personajes no existen, se hace declaración de una nueva variable que es para declarar al ganador y el registro de combates, al ser

ingresados los personajes, se genera números aleatorios, (0 y 1) que si al ganador le toca 0 es el ganador y el resultado se guarda con la hora y fecha que se haya realizado, en el formato de (dd/mm/yyyy) y (hh/mm/ss) y el ganador del combate, y con una condicionante if que ira desde la cantidad de peleas al tamaño de la matriz de historial de combates (que es 50), se sume 1 a la cantidad de combates hechos, y que nos indique que el programa ya ha hecho registro del combate.

```
            case 7:
                System.out .println("Ver historial de combates");
                if(cantPeleas ==0){
                    System.out .println("No hay peleas registradas... aun");
                }else {
                    for(int i=0; i<cantPeleas; i++){
                        System.out .println(historialCo[i]);
                    }
                }
                break;
            case 8:
                System.out .println("Este programa fue hecho por Guillermo Enrique Marroquin Morán");
                System.out .println("202103527");
                System.out .println("Este programa tiene derechos de autor o no...");
                break;
            case 9:
                System.out .println("BYE BYEEEEE");
                break;
            default:
                System.out .println("!Esa opcion no existe!, ingresa otra opción valida.");
        }
    } while (escribir != 9); // Todo esto se repite indefinidamente hasta que pongan 9
      scanner.close();
}
}
```

Para la opción 7 se hace un llamado del registro de las peleas hechas, si el contador es 0 nos indicara que no se ha realizado nada, si hay peleas registradas nos mostrara con un bucle for que va de 0 hasta el número de combates realizados, estos se muestran en pantalla con el ganador, hora y fecha. Para la opción 8 solo imprime en pantalla mis datos de estudiante y una pequeña broma, que en los comentarios pongo algo que sea divertido de ver. En la última opción se muestra un mensaje de despedida, y si ponemos un numero que no está el programa nos indicara ello, y al poner 9 el bucle do while se cierra al cumplir la condición.