**Universidad de las Fuerzas Armadas “ESPE”**



Departamento de Ciencias de la Computación

Carrera: Ingeniería en Tecnologías de la Información

Programación Orientada a Objetos

Actividad de aprendizaje Práctico Experimental n.° 2

Caso de Estudio 8: Sistema de gestión de eventos

Grupo 3

Integrantes:

Keyla Lucero Marcillo Flores

Issac Miguel Morales Alcoser

Barbara Camila Rodas Paredes

Rosa Silvana Quito Soto

Kevin Daniel Shagñay Arequipa

Fecha de entrega: 03/06/2024

Sangolquí – Ecuador

**Caso de Estudio 8: Sistema de Gestión de Eventos**

Enunciado:

Cliente: Necesitamos un sistema de gestión de eventos que permita registrar eventos, inscribir participantes y gestionar la información de los eventos y los participantes. Para los participantes, necesito solicitar la información de nombre y id\_participante. Para los eventos, yo podré ingresar el nombre del evento, la fecha y el organizador. Recuerda que id\_participante solo puede ser accedida en la inscripción y desde el evento, por lo que te recomiendo que sea privado. Además, los métodos que podrías utilizar son los siguientes:

registrarEvento()

inscribirParticipante()

cancelarEvento()

verParticipantes()

Si quieres aumentar o reemplazar los métodos, indica cuál sería el motivo.

**RESOLUCIÓN DEL PROBLEMA**

**DIAGRAMAS DE FLUJO**

 **Registrar evento**

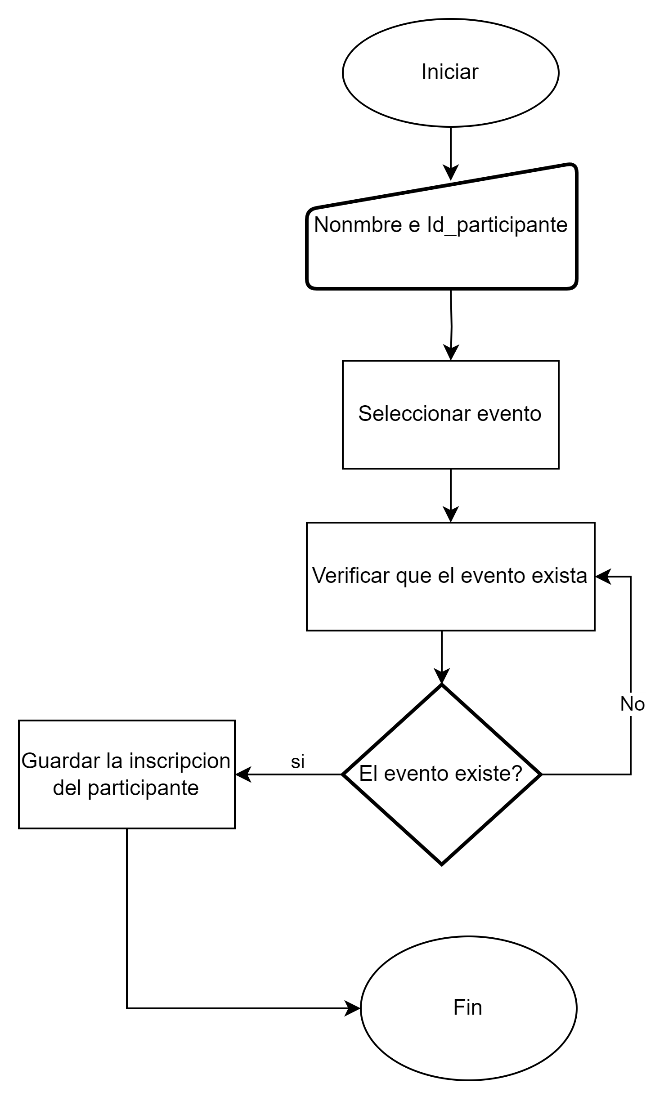
Este diagrama de flujo representa el proceso para gestionar la información de un evento. Los pasos que utilizamos son:

Inicio: Representado por un óvalo etiquetado como "Iniciar".

Ingresar datos del evento: Representado por una figura de datos de entrada, donde se ingresan los datos del evento.

Guardar la información del evento: Representado por un rectángulo, que indica que la información del evento se guarda en la base de datos.

Fin: Representado por un óvalo etiquetado como "Fin".

**Inscribir participante**

Este diagrama representa un flujo de proceso para la inscripción de un participante en un evento. Los pasos que utilizamos son:

Inicio: representado por un ovalo en la parte superior.

Nombre e ID del participante: Se solicita al usuario que ingrese su nombre e ID.

Seleccionar: El participante debe seleccionar el evento al que desea inscribirse.

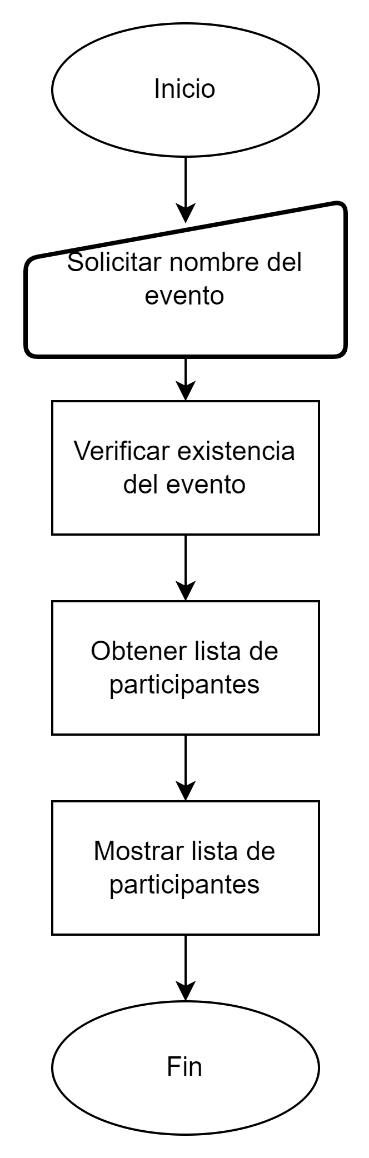
Verificar que el evento exista: Se verifica si el evento seleccionado existe.

Si el evento no existe, el flujo termina aquí (Fin).

Si el evento existe, el flujo continúa a guardar la inscripción del participante.

Fin: Representa el final del proceso, indicando que la inscripción se ha completado o que el proceso ha terminado porque el evento no existía.

**Ver participantes**



Este diagrama muestra el proceso para verificar y mostrar la lista de participantes de un evento, los pasos que utilizamos son:

Inicio: representado por un ovalo indicando "Inicio".

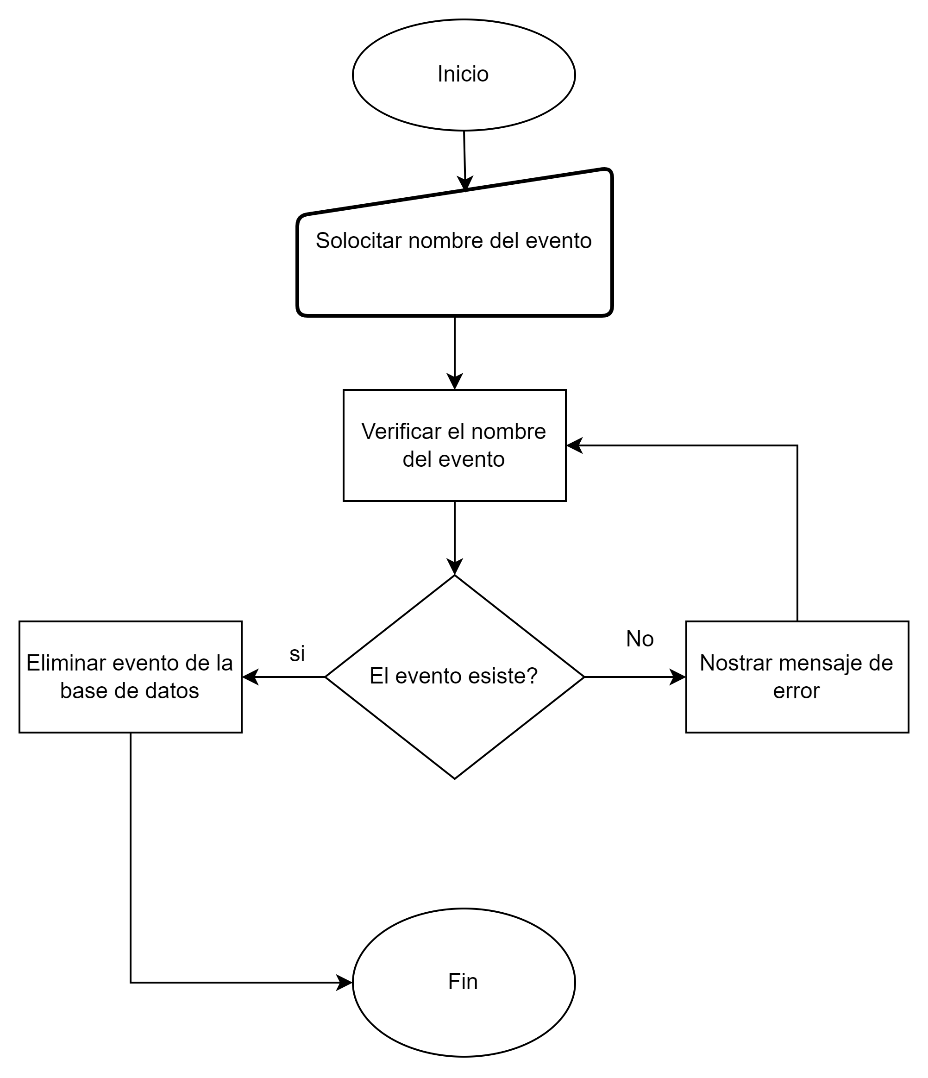
Solicitar nombre del evento: Se solicita el nombre del evento al usuario.

Verificar existencia del evento: Se verifica si el evento existe.

Obtener lista de participantes: Si el evento existe, se obtiene la lista de participantes.

Mostrar lista de participantes: La lista de participantes se muestra al usuario.

Fin: El flujo termina.

**Eliminar evento**

Este diagrama de flujo describe el proceso de verificación y eliminación de un evento en una base de datos. Los pasos que utilizamos son:

Inicio: Comienza el proceso.

Solicitar nombre del evento: Se pide al usuario que introduzca el nombre del evento.

Verificar el nombre del evento: Se comprueba si el nombre del evento existe en la base de datos.

¿El evento existe?: Evaluación de la existencia del evento.

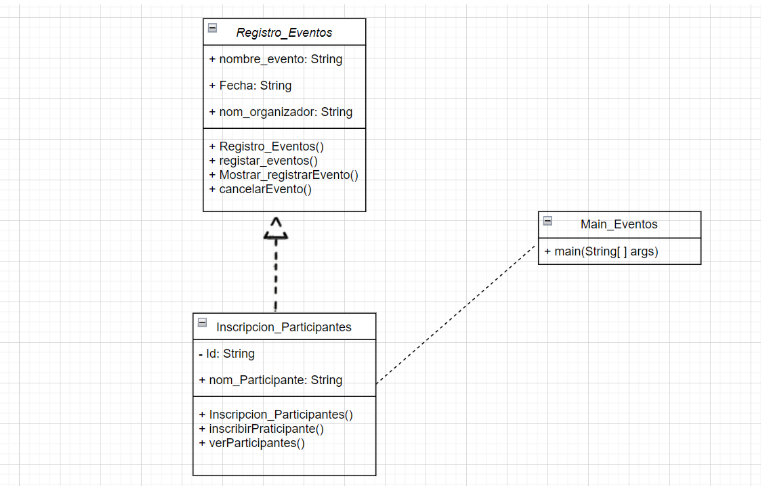
Sí: Si el evento existe, se procede a eliminarlo de la base de datos.

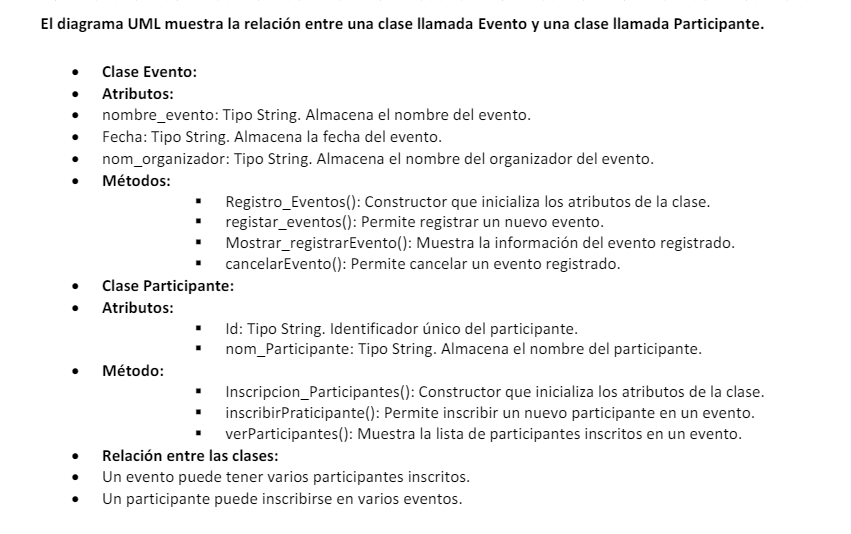
No: Si el evento no existe, se muestra un mensaje de error.

Si el evento existe, procede a eliminar evento de la base de datos: El evento es eliminado.

Fin: Finaliza el proceso.

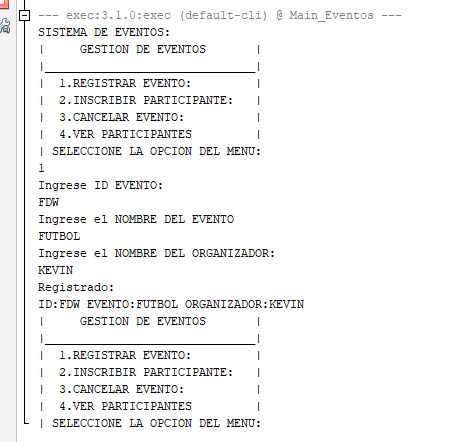
**DIAGRAMA DE CLASES**

****

****

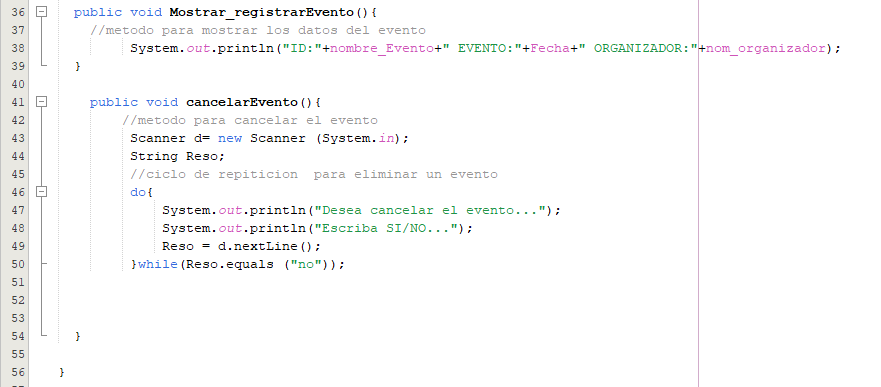
**CODIGO**

**PANTALLA**

****

**CLASE REGISTRO\_EVENTOS**

****

****

import java.util.Scanner;

/\*\*

\*

\* @author Kevin

\*/

public class Registro\_Eventos {

String nombre\_Evento,Fecha,nom\_organizador;

//metodo constructor que no devuelve ningun valor

public Registro\_Eventos(){

}

//metodo

public void registrarEvento(){

//metodo para el registro de un evento

Scanner dato= new Scanner (System.in);

System.out.println("Ingrese ID EVENTO:" );

nombre\_Evento=dato.nextLine();

System.out.println("Ingrese el NOMBRE DEL EVENTO" );

Fecha=dato.nextLine();

System.out.println("Ingrese el NOMBRE DEL ORGANIZADOR:" );

nom\_organizador =dato.nextLine();

System.out.println("Registrado:" );

}

public void Mostrar\_registrarEvento(){

//metodo para mostrar los datos del evento

System.out.println("ID:"+nombre\_Evento+" EVENTO:"+Fecha+" ORGANIZADOR:"+nom\_organizador);

}

public void cancelarEvento(){

//metodo para cancelar el evento

Scanner d= new Scanner (System.in);

String Reso;

//ciclo de repiticion para eliminar un evento

do{

System.out.println("Desea cancelar el evento...");

System.out.println("Escriba SI/NO...");

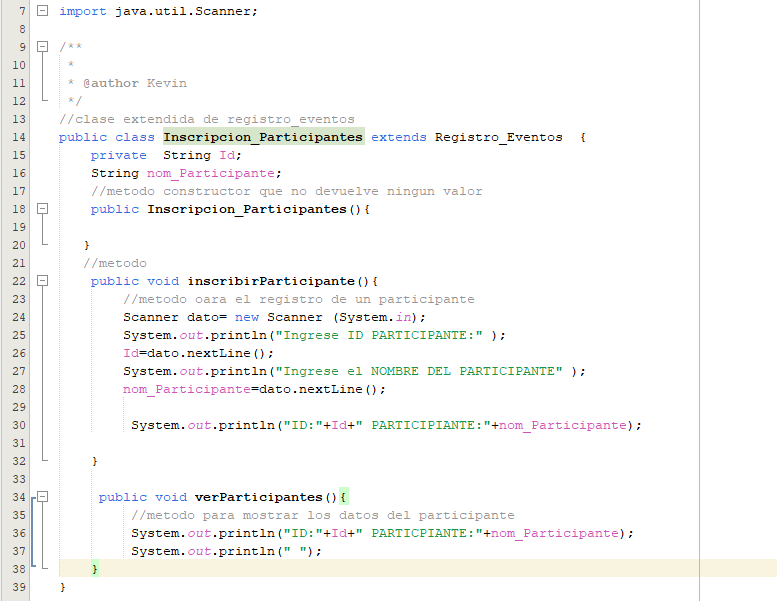
Reso = d.nextLine();

}while(Reso.equals ("no"));

}

}

**CLASE INSCRIPCION\_PARTICIPANTES**

****

import java.util.Scanner;

/\*\*

\*

\* @author Kevin

\*/

//clase extendida de registro\_eventos

public class Inscripcion\_Participantes extends Registro\_Eventos {

private String Id;

String nom\_Participante;

//metodo constructor que no devuelve ningun valor

public Inscripcion\_Participantes(){

}

//metodo

public void inscribirParticipante(){

//metodo oara el registro de un participante

Scanner dato= new Scanner (System.in);

System.out.println("Ingrese ID PARTICIPANTE:" );

Id=dato.nextLine();

System.out.println("Ingrese el NOMBRE DEL PARTICIPANTE" );

nom\_Participante=dato.nextLine();

System.out.println("ID:"+Id+" PARTICIPIANTE:"+nom\_Participante);

}

public void verParticipantes(){

//metodo para mostrar los datos del participante

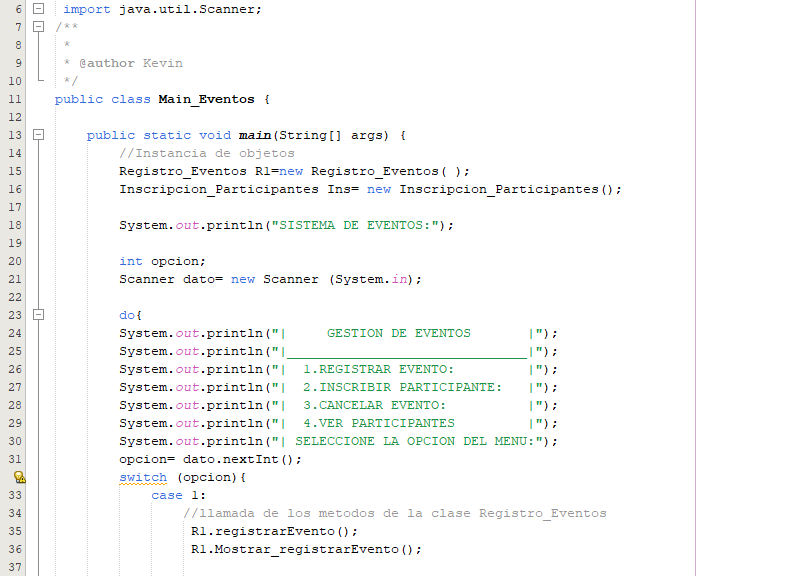
System.out.println("ID:"+Id+" PARTICPIANTE:"+nom\_Participante);

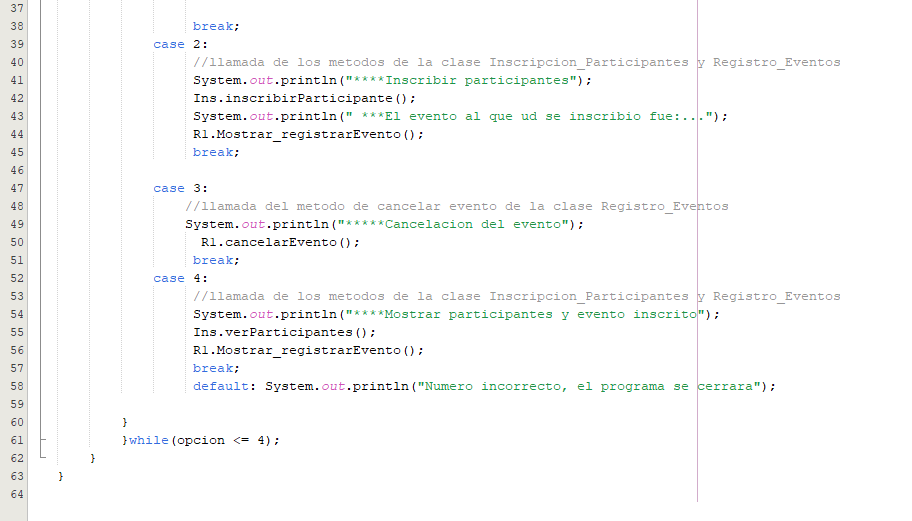
System.out.println(" ");

}

}

**CLASE MAIN**

****

****

import java.util.Scanner;

/\*\*

\*

\* @author Kevin

\*/

public class Main\_Eventos {

public static void main(String[] args) {

//Instancia de objetos

Registro\_Eventos R1=new Registro\_Eventos( );

Inscripcion\_Participantes Ins= new Inscripcion\_Participantes();

System.out.println("SISTEMA DE EVENTOS:");

int opcion;

Scanner dato= new Scanner (System.in);

do{

System.out.println("| GESTION DE EVENTOS |");

System.out.println("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|");

System.out.println("| 1.REGISTRAR EVENTO: |");

System.out.println("| 2.INSCRIBIR PARTICIPANTE: |");

System.out.println("| 3.CANCELAR EVENTO: |");

System.out.println("| 4.VER PARTICIPANTES |");

System.out.println("| SELECCIONE LA OPCION DEL MENU:");

opcion= dato.nextInt();

switch (opcion){

case 1:

//llamada de los metodos de la clase Registro\_Eventos

R1.registrarEvento();

R1.Mostrar\_registrarEvento();

break;

case 2:

//llamada de los metodos de la clase Inscripcion\_Participantes y Registro\_Eventos

System.out.println("\*\*\*\*Inscribir participantes");

Ins.inscribirParticipante();

System.out.println(" \*\*\*El evento al que ud se inscribio fue:...");

R1.Mostrar\_registrarEvento();

break;

case 3:

//llamada del metodo de cancelar evento de la clase Registro\_Eventos

System.out.println("\*\*\*\*\*Cancelacion del evento");

R1.cancelarEvento();

break;

case 4:

//llamada de los metodos de la clase Inscripcion\_Participantes y Registro\_Eventos

System.out.println("\*\*\*\*Mostrar participantes y evento inscrito");

Ins.verParticipantes();

R1.Mostrar\_registrarEvento();

break;

default: System.out.println("Numero incorrecto, el programa se cerrara");

}

}while(opcion <= 4);

}

}

**RESUMEN**

**CLASE REGISTRO DE EVENTOS**

**1.-** Lo primero que hacemos en la línea 1 es almacenar todas las clases dentro de un paquete para una correcta organización y gestión del código.

**2.-** En la línea 3 importamos la clase scanner de la biblioteca de utilidades (java.util). Utilizamos la clase scanner para leer datos ingresados por el usuario a través de un teclado.

**3.-** En la línea 5nombramos a nuestra clase pública, en este caso nuestra clase se llama **Registro\_Eventos.**

**4.-** En la línea 8 colocamos los atributos de la clase, nuestra clase posee tres atributos: el nombre del evento (nombre\_Evento), Fecha (Fecha) y el nombre del organizador (nom\_organizador). Podemos observar que los tres atributos son de tipo String es decir son utilizados para almacenar caracteres.

**5.-** En la línea 10 colocamos un constructor vacío el cual permite instanciar a la clase sin necesidad de parámetros.

**6.-** En la línea 15 hasta la 25 utilizamos el método public void registrarEvento para ingresar los detalles del evento.

* Dentro de este método, en la línea 17 creamos una instancia de la clase Scanner llamada dato la cual será utilizada para leer la entrada del usuario desde “System.in”.
* En la línea 18 se coloca un mensaje en consola pidiendo al usuario que ingrese el ID del evento.
* En la línea 19 se lee la línea completa que el usuario ingreso anteriormente y le asigna al atributo “nombre\_evento”.
* En la línea 20 se coloca un mensaje en consola pidiendo al usuario que ingrese el nombre del evento.
* En la línea 21 se lee la línea completa que el usuario ingreso anteriormente y le asigna al atributo “Fecha”.
* En la línea 22 se coloca un mensaje en consola pidiendo al usuario que ingrese el nombre del organizador.
* En la línea 23 se lee la línea completa que el usuario ingreso anteriormente y le asigna al atributo “nom\_organizador”.
* En la línea 24 se imprime un mensaje en la consola indicando que la información del evento ha sido registrada.

**7.-** Desde la línea 27 hasta la 30 se coloca el método “public void Mostrar\_registrarEvento”, este método va a imprimir los datos ingresados anteriormente (ID, nombre del evento, nombre del organizador) en la consola.

**8.-** Desde la línea 32 hasta la 41 utilizamos el método “cancelarEvento”

* Dentro de este método, en la línea 33 creamos una instancia de la clase Scanner llamada d la cual será utilizada para leer la entrada del usuario desde “System.in”.
* En la línea 34 se crea una variable de tipo String llamada “Reso”, la cual utilizaremos para almacenar la respuesta del usuario.
* Desde la línea 35 hasta la 39 se crea un bucle “do-while” el cual ejecuta el bloque de código dentro de “do” al menos una vez, y luego continúa ejecutándolo hasta que “Reso” sea igual a **no**.
  + En la línea 36 se imprime un mensaje en la consola preguntando al usuario si desea cancelar el evento.
  + En la línea 37 imprime dos opciones para el usuario escriba su respuesta **SI** o **NO.**
  + En la línea 38 se lee la línea completa que el usuario ingreso anteriormente y le asigna a la variable “Reso”.

**9.-** Finalizamos esta clase cerrando con llave.

**CLASE INSCRIPCION DE PARTICIPANTES**

**1.-** Lo primero que hacemos en la línea 1 es almacenar todas las clases dentro de un paquete para una correcta organización y gestión del código.

**2.-** En la línea 3 importamos la clase scanner de la biblioteca de utilidades (java.util). Utilizamos la clase scanner para leer datos ingresados por el usuario a través de un teclado.

**3.-** En la línea 5nombramos a nuestra clase pública, en este caso nuestra clase se llama **Inscripcion\_Participantes.** Esta clase hereda de la clase “Registro\_Eventos”, lo que implica que tiene acceso a los métodos y propiedades de “Registro\_Eventos”.

4.- En la línea 6 utilizamos un atributo privado String el cual almacena el ID del participante.

5.- En la línea 7 utilizamos un atributo por default String el cual almacena el nombre del participante.

6.- En la línea 9 colocamos un constructor vacío el cual inicializa la clase.

7.- Desde la línea 13 hasta la 23 colocamos el método “public void inscribirParticipante” para ingresar los datos del participante.

* Dentro de este método, en la línea 15 creamos una instancia de la clase Scanner llamada dato la cual será utilizada para leer la entrada del usuario desde “System.in”.
* En la línea 16 se coloca un mensaje en consola pidiendo al usuario que ingrese el ID del participante.
* En la línea 17 se lee la línea completa que el usuario ingreso anteriormente y le asigna al atributo “Id”.
* En la línea 18 se coloca un mensaje en consola pidiendo al usuario que ingrese el nombre del participante.
* En la línea 19 se lee la línea completa que el usuario ingreso anteriormente y le asigna al atributo “nom\_Participante”.
* En la línea 21 se imprimen los datos ingresados anteriormente (ID y nombre del participante)

8.- Desde la línea 25 hasta la 29 utilizamos el método “public void participantes” el cual nos va a mostrar los datos ingresados de los participantes.

* Se imprime en consola el ID y el nombre del participante almacenados en los atributos “Id” y “nom\_Participante”.
* Se imprime “ “ para una línea en blanco utilizada para espaciar la salida en consola.

9.- Para finalizar cerramos la clase con llave.

**CLASE MAIN EVENTOS**

**1.-** Lo primero que hacemos en la línea 1 es almacenar todas las clases dentro de un paquete para una correcta organización y gestión del código.

**2.-** En la línea 3 importamos la clase scanner de la biblioteca de utilidades (java.util). Utilizamos la clase scanner para leer datos ingresados por el usuario a través de un teclado.

**3.-** En la línea 4nombramos a nuestra clase pública, en este caso nuestra clase se llama **Main\_Eventos.**

**4.-** En la línea 6 utilizamos el método public static void main para poder correr el programa.

* Dentro de este método, en la línea 8 creamos una instancia de la clase Registro\_Eventos llamada R1.
* Dentro de este método, en la línea 9 creamos una instancia de la clase Inscripcion\_Participantes llamada Ins.
* En la línea 11 se imprime un mensaje indicando el sistema de eventos.
* En la línea 13 se declara una variable entera para almacenar la elección del menú de usuario.
* En la línea 14 se crea una instancia de un objeto llamado dato para leer la entrada del usuario.
* Desde la línea 16 hasta la 54 se crea un bucle Do-While, este bucle asegura que el menú se muestre y la entrada del usuario se procese repetidamente hasta que se seleccione una opción invalida (menor o igual a 4).
  + Desde la línea 17 hasta la 23 se imprimen las opciones del menú en la consola.
  + En la línea 24 se almacena la opción del usuario en la variable “opción”.
  + Desde la línea 25 hasta la 53 se utiliza una declaración “switch” para manejar la elección del usuario.
    - Desde la línea 26 hasta la 31 tenemos el primer caso el cual se encarga de mostrar un evento ingresado en los métodos de la clase Registro\_Eventos.
    - Desde la línea 32 hasta la 38 tenemos el caso 2 el cual en la línea 34 imprime un mensaje indicando el proceso para inscribir a un participante, en la línea 35 se llama al método inscribirParticipante dentro de la clase Inscripcion\_Participantes para poder inscribir los datos del participante, en la línea 36 se imprime un mensaje indicando el evento al cual fue inscrito el participante, en la línea 37 se llama al método Mostrar\_registrarEvento dentro de la clase Registro\_Eventos para mostrar los detalles del evento, finalmente se corta este caso.
    - Desde la línea 40 hasta la 44 se encuentra el caso 3 el cual imprime un mensaje indicando el proceso de cancelación del evento y llama al método cancelarEvento dentro de la clase Registro\_Eventos para cancelar un evento.
    - Desde la línea 44 hasta la 50 se encuentra el caso 4 el cual imprime un mensaje indicando el proceso para mostrar los participantes y el evento inscrito, se llama al método verParticipnates de la clase Inscripcion\_Participantes para poder mostrar a los participantes, posteriormente se llama al método Mostrar\_resgistrarEvento de la clase Registro\_Eventos para mostrar los detalles del evento.
    - Finalmente si se ingresa un número invalido, se imprimirá un mensaje notificándonos que el número que ingresamos no es el correcto y que el programa se cerrara.
* Para terminar cerramos la clase con llaves.