**TRABAJO PRÁCTICO INTEGRADOR**

**PARADIGMA ORIENTADO A OBJETOS**

**“Gestor de Eventos”**

**Autor:**

Christian Sosa - LU1202494

**Profesor/as:**

Diego Sebastian Tolbaños

**Cuatrimestre: 01 – Año: 2025**



**INFORME**

Clase **Main**

**Responsabilidad:**  
Clase principal de la aplicación. Controla el flujo del programa, muestra un menú por consola y ejecuta las **acciones** seleccionadas por el usuario.

**Atributos:**  
*No posee atributos.*

**Métodos:**

**• public static void main(String[] args)**

Es el punto de entrada del programa. Crea una instancia de GestorEventos, carga los eventos desde archivo y ejecuta un bucle que muestra el menú principal hasta que el usuario elige salir.

**• private static void mostrarMenu()**

Muestra por consola las distintas opciones del menú que el usuario puede seleccionar para gestionar los eventos y asistentes.

**• private static int leerOpcion()**

Lee y convierte la opción ingresada por el usuario. Si el valor ingresado no es un número, devuelve -1 para indicar error.

**• private static void procesarOpcion(int opcion)**

Ejecuta la acción correspondiente a la opción seleccionada, llamando a los métodos internos según el caso.

**• private static void agregarEvento()**

Solicita los datos necesarios para crear un nuevo evento (título, fecha, ubicación y descripción), y lo agrega mediante gestor.

**• private static void buscarEvento()**

Pide un título por consola y busca el evento correspondiente. Si lo encuentra, muestra sus datos y asistentes.

**• private static void editarEvento()**

Permite modificar un evento existente. Si el usuario deja un campo vacío, ese atributo no se modifica. Usa el título como clave para encontrar el evento.

**• private static void eliminarEvento()**

Solicita un título de evento por consola y lo elimina si existe en la lista.

**• private static void agregarAsistenteAEvento()**

Pide el título del evento, el nombre y el email del asistente, y lo registra en el evento correspondiente.

**• private static void guardarEventosEnArchivo()**

Llama al método ArchivoUtils.guardarEventos() para guardar los eventos actuales en un archivo de texto.

**• private static void cargarEventosDesdeArchivo()**

Verifica si existe el archivo con eventos guardados y, si es así, los carga y los agrega al gestor actual.

**• private static Date parsearFecha(String fechaStr)**

Convierte una cadena en formato "dd/MM/yyyy HH:mm" a un objeto Date. Muestra un mensaje si el formato es incorrecto.

**• private static void mostrarCalendarioMensual()**

Dibuja un calendario textual del mes actual, marcando los días con eventos entre corchetes. También muestra un resumen de los eventos del mes.

**• private static List<Integer> obtenerDiasConEventos(int mes, int año)**

Devuelve una lista de los días (números) del mes indicado en los que hay al menos un evento registrado.

**• private static void mostrarEventosDelMes(int mes, int año)**

Muestra una lista detallada de los eventos programados para el mes y año especificado.

Clase **GestorEventos**

Se encarga de la lógica del sistema, gestiona la lista de eventos y los asistentes, y maneja la persistencia mediante archivos.

**Atributos:**

### • private List<Evento> listaEventos

Almacena todos los eventos registrados en una lista. Es la estructura principal del gestor.

**Métodos:**

**• public GestorEventos()**

Constructor que inicializa la lista de eventos como una nueva lista vacía.

**• public void agregarEvento(Evento evento)**

Agrega un nuevo evento a la lista si no está repetido y no es nulo.

**• public void editarEvento(String titulo)**

Busca un evento por su título y muestra un mensaje indicando que la función de edición fue llamada. (Método placeholder, se puede extender o eliminar si no se usa).

**• public void editarEvento(String titulo, Date nuevaFecha, String nuevaUbicacion, String nuevaDescripcion)**

Edita un evento ya existente, reemplazando solo los campos que se pasan como no nulos (fecha, ubicación, descripción).

**• public void listarEventosFuturos()**

Muestra todos los eventos cuya fecha es posterior a la actual (fecha futura).

**• public void listarEventosPasados()**

Muestra todos los eventos cuya fecha es anterior a la actual (eventos ya realizados).

**• public void listarTodosLosEventos()**

Muestra por consola todos los eventos registrados, sin importar la fecha.

**• public Evento buscarEventoPorTitulo(String titulo)**

Busca y devuelve un evento cuyo título coincida (ignorando mayúsculas/minúsculas). Devuelve null si no se encuentra.

**• public boolean eliminarEvento(String titulo)**

Elimina el evento con el título indicado si existe. Devuelve true si lo elimina, false si no lo encuentra.

**• public void agregarAsistenteAEvento(String tituloEvento, Asistente asistente)**

Agrega un asistente al evento indicado por su título. Si el evento no se encuentra, muestra un mensaje de error.

**• public List<Evento> getListaEventos()**

Devuelve una **copia** de la lista de eventos para evitar que se modifique la original desde fuera de la clase.

**• public int getNumeroDeEventos()**

Devuelve la cantidad total de eventos registrados en el gestor.

Clase **Eventos**

**Responsabilidad:**  
Representa un evento. Contiene la información básica del evento y la lista de asistentes inscriptos.

**Atributos:**

**• private String titulo**

Guarda el nombre del evento. Se usa como identificador único en el método equals().

**• private Date fecha**

Fecha y hora en que se realizará el evento.

**• private String ubicacion**

Dirección o lugar físico donde se lleva a cabo el evento.

**• private String descripcion**

Texto descriptivo con más información sobre el evento.

**• private ArrayList<Asistente> asistentes**

Lista de personas registradas para asistir al evento.

**Métodos:**

**public Evento(String titulo, Date fecha, String ubicacion, String descripcion)**

Constructor que crea un nuevo evento con los valores dados y una lista vacía de asistentes.

**• public String getTitulo()**

Devuelve el título del evento.

**• public Date getFecha()**

Devuelve la fecha y hora del evento.

**• public String getUbicacion()**

Devuelve la ubicación del evento.

**• public String getDescripcion()**

Devuelve la descripción del evento.

**• public ArrayList<Asistente> getAsistentes()**

Devuelve la lista de asistentes registrados al evento.

**• public void setTitulo(String titulo)**

Permite modificar el título del evento.

**• public void setFecha(Date fecha)**

Permite modificar la fecha del evento.

**• public void setUbicacion(String ubicacion)**

Permite cambiar la ubicación del evento.

**• public void setDescripcion(String descripcion)**

Permite cambiar la descripción del evento.

**• public void agregarAsistente(Asistente asistente)**

Agrega un asistente a la lista solo si no está repetido.

**• public void eliminarAsistente(Asistente asistente)**

Elimina un asistente específico de la lista.

**• public String toString()**

Devuelve una representación en texto del evento, incluyendo título, fecha, ubicación, descripción y cantidad de asistentes

**• public boolean equals(Object obj)**

Sobrescribe el método equals() para considerar que **dos eventos son iguales si tienen el mismo título**.

Clase **Asistente**

**Responsabilidad:**  
Modela a una persona que asistirá a un evento, guardando su nombre y correo electrónico.

**Atributos:**

**• private String nombre**

Guarda el nombre del asistente registrado para un evento.

**• private String email**

Almacena el correo electrónico del asistente, útil para contacto o identificación.

**Métodos:**

**public Asistente(String nombre, String email)**

Constructor que crea una instancia de asistente con su nombre y email.

**• public String getNombre()**

Devuelve el nombre del asistente.

**• public void setNombre(String nombre)**

Permite modificar el nombre del asistente.

**• public String getEmail()**

Devuelve el email del asistente.

**• public void setEmail(String email)**

Permite cambiar el email del asistente.

**• public String toString()**

Devuelve una cadena de texto con el nombre y el email del asistente, útil para mostrar en pantalla o guardar en archivo.

### • public boolean equals(Object obj)

Sobrescribe el método equals() para considerar que **dos asistentes son iguales si tienen el mismo email**, ya que se asume que el email es único por persona.

Clase **ArchivoUtils**

**Responsabilidad:**  
Clase utilitaria. Se encarga de guardar y recuperar datos desde archivos de texto para asegurar la persistencia de los eventos.

### Atributos: • private static final String ARCHIVO\_EVENTOS

Constante que define el nombre del archivo de texto donde se guardan los eventos (eventos.txt).

### • private static final SimpleDateFormat dateFormat

Formato utilizado para leer y escribir fechas en el archivo de eventos (dd/MM/yyyy HH:mm).

**Métodos:**

**• public static void guardarEventos(List<Evento> eventos)**

Guarda la lista de eventos en el archivo eventos.txt.  
Cada línea del archivo contiene:  
título | fecha | ubicación | descripción | asistentes  
Los asistentes se guardan en formato nombre:email separados por coma.

**• public static List<Evento> cargarEventos()**

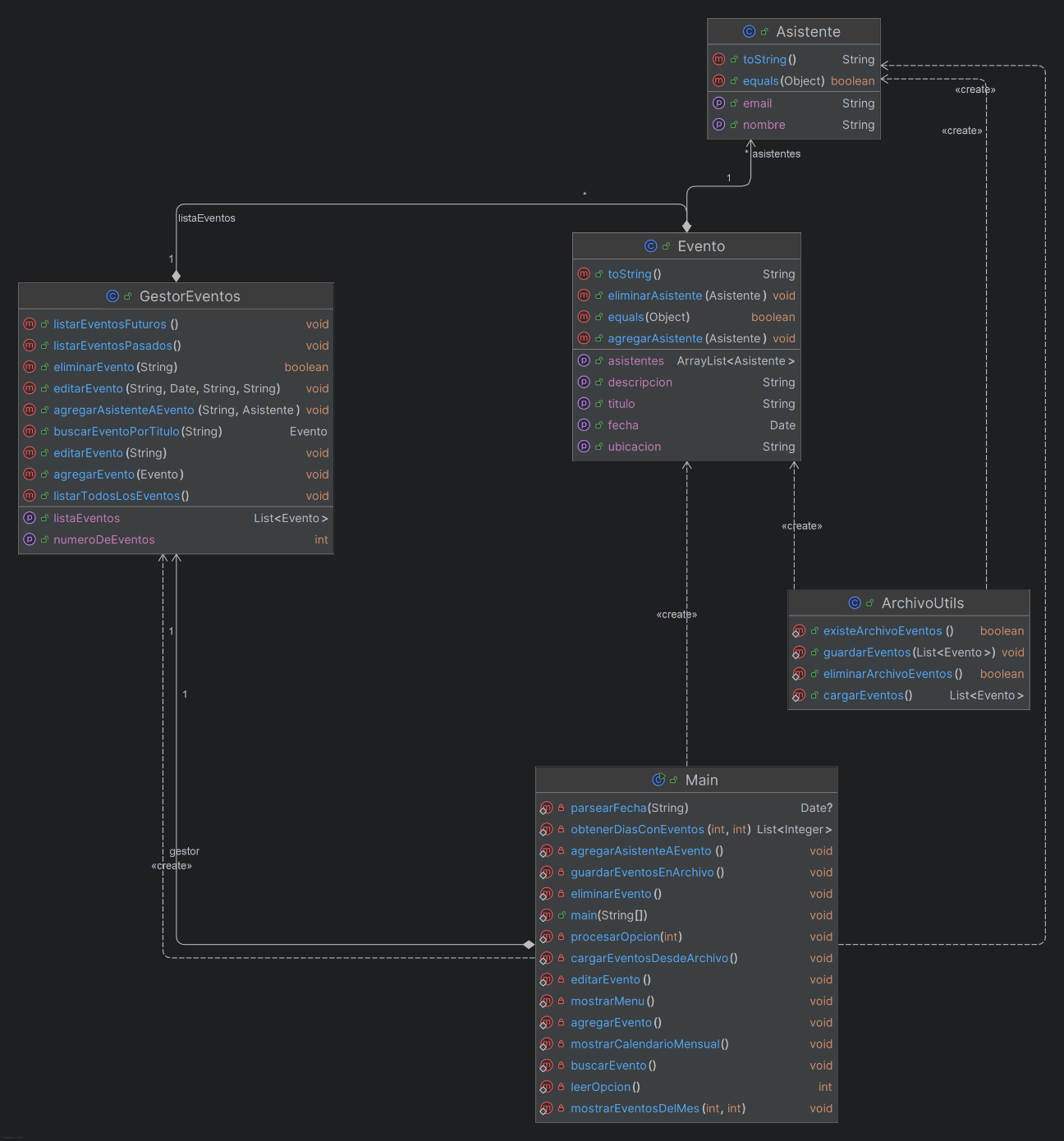
Lee el archivo eventos.txt, parsea los datos y reconstruye los objetos Evento y sus Asistentes.  
Devuelve una lista con todos los eventos encontrados en el archivo.

**• public static boolean existeArchivoEventos()**

Verifica si el archivo eventos.txt existe en el sistema de archivos.  
Devuelve true si existe, false si no.

**• public static boolean eliminarArchivoEventos()**

Intenta eliminar el archivo eventos.txt.  
Devuelve true si se eliminó correctamente, false si no existe o no se pudo borrar.

**DIAGRAMA**