Documentacion Practica UND2

08/10/2023
2ºDAM

RAUL RAMIREZ BOTELLA

ÍNDICE

Clase ArchivoRegistros

La clase “ArchivoRegistros” constituye un componente fundamental de nuestra aplicación de gestión de registros. Su función principal es la administración de un archivo que almacena información relativa a los registros de datos. Esta clase posibilita la creación, modificación, recuperación y gestión de los registros contenidos en el archivo.

Constructor

“ArchivoRegistros(String ruta)”

El constructor de la clase “ArchivoRegistros” acepta una ruta de archivo como argumento. En caso de que el archivo no exista en la ubicación especificada, se generará automáticamente con registros de ejemplo.

```
ArchivoRegistros archivoRegistros = new ArchivoRegistros("ruta_del_archivo");
```

Métodos

“crearArchivoPorDefecto()”

Este procedimiento genera automáticamente un archivo predeterminado con registros de ejemplo en caso de que el archivo no se encuentre en la ubicación especificada.

“agregarRegistro(Registro registro)”

Este procedimiento habilita la inclusión de un nuevo registro en el archivo. Se realiza una verificación para determinar si la clave (código) suministrada ya se encuentra presente en el archivo, y en caso de existir duplicados, se solicita una clave alternativa.

```
Registro nuevoRegistro = new Registro("0004", "Luisa", "Perez");
archivoRegistros.agregarRegistro(nuevoRegistro);
```

“verificarClaveExistente(String clave)”

Verifique la existencia de una clave (código) en el archivo de registros. Devuelva true si la clave ya existe, de lo contrario, devuelva false.

“recuperarRegistro(String codigoBuscar)”

Recupere un registro del archivo según su código. Devuelva un objeto Registro si se encuentra, o null si no se encuentra ningún registro con el código proporcionado.

```
Registro registroRecuperado = archivoRegistros.recuperarRegistro("0002");
```

“modificarRegistro(String codigoModificar, Registro nuevoRegistro)”

Esta funcionalidad permite la modificación de un registro preexistente en el archivo, sustituyendo los datos del registro correspondiente con la información actualizada proporcionada.

```
Registro nuevoRegistroModificado = new Registro("0002", "Alvaro", "Gomez");
archivoRegistros.modificarRegistro("0002", nuevoRegistroModificado);
```

“mostrarCodigosRegistros()”

Muestra los códigos de registros disponibles en el archivo.

```
archivoRegistros.mostrarCodigosRegistros();
```

“mostrarRegistrosInsertados()”

Muestra todos los registros insertados en el archivo, excluyendo los registros marcados como borrados.

```
archivoRegistros.mostrarRegistrosInsertados();
```

“marcarRegistroComoBorrado(String codigoBorrar)”

Marca un registro como borrado en el archivo. Los registros marcados como borrados no se mostrarán en la lista de registros insertados.

```
archivoRegistros.marcarRegistroComoBorrado("0003");
```

“mostrarRegistrosMarcadosComoBorrados()”

Muestra los registros marcados como borrados en el archivo.

```
archivoRegistros.mostrarRegistrosMarcadosComoBorrados();
```

“compactarArchivo()”

Compacta el archivo eliminando registros marcados como borrados. El archivo original se renombra como "registro.bak" y se crea un nuevo archivo "registro.txt" sin los registros marcados como borrados.

```
archivoRegistros.compactarArchivo();
```

Clase GestorRegistros

La clase “GestorRegistros” tiene la responsabilidad de administrar los registros de datos en un archivo, posibilitando la realización de operaciones tales como la inclusión, modificación, recuperación y eliminación de registros.

Constructor

“GestorRegistros(String ruta)”

El constructor de la clase GestorRegistros recibe la ruta del archivo en la que se almacenan los registros. Si el archivo no existe en la ubicación especificada, se crea automáticamente con registros de ejemplo.

```
GestorRegistros gestor = new GestorRegistros("ruta_del_archivo");
```

Métodos

“agregarRegistro(Registro registro)”

Agrega un nuevo registro al archivo. Verifica si la clave (código) proporcionada ya existe en el archivo y solicita una clave diferente en caso de duplicados.

```
Registro nuevoRegistro = new Registro("0004", "Luisa", "Perez");
gestor.agregarRegistro(nuevoRegistro);
```

“verificarClaveExistente(String clave)”

Verifica si una clave (código) ya existe en el archivo de registros. Retorna true si la clave ya existe, de lo contrario, retorna false.

“crearArchivoPorDefecto()”

Crea un archivo por defecto con registros de ejemplo si el archivo no existe en la ubicación especificada.

“recuperarRegistro(String codigoBuscar)”

Recupera un registro del archivo basado en su código. Retorna un objeto Registro si se encuentra, o null si no se encuentra ningún registro con el código proporcionado.

```
Registro registroRecuperado = gestor.recuperarRegistro("0002");
```

“modificarRegistro(String codigoModificar, Registro nuevoRegistro)”

Permite modificar un registro existente en el archivo. Reemplaza los datos del registro correspondiente con los nuevos datos proporcionados.

```
Registro nuevoRegistroModificado = new Registro("0002", "Alvaro", "Gomez");
gestor. modificarRegistro("0002", nuevoRegistroModificado);
```

“mostrarCodigosRegistros()”

Muestra los códigos de registros disponibles en el archivo.

```
gestor.mostrarCodigosRegistros();
```

“mostrarRegistrosInsertados()”

Muestra todos los registros insertados en el archivo, excluyendo los registros marcados como borrados.

```
gestor.mostrarRegistrosInsertados();
```

“marcarRegistroComoBorrado(String codigoBorrar)”

Marca un registro como borrado en el archivo. Los registros marcados como borrados no se mostrarán en la lista de registros insertados.

```
gestor.marcarRegistroComoBorrado("0003");
```

“mostrarRegistrosMarcadosComoBorrados()”

Muestra los registros marcados como borrados en el archivo.

```
gestor.mostrarRegistrosMarcadosComoBorrados();
```

“compactarArchivo()”

Compacta el archivo eliminando registros marcados como borrados. El archivo original se renombra como "registro.bak" y se crea un nuevo archivo "registro.txt" sin los registros marcados como borrados.

```
gestor.compactarArchivo();
```

“actualizarIndicesDespuesDeCompactar()”

Actualiza los índices después de compactar el archivo. Esto es importante para mantener la integridad de los registros y sus posiciones.

Clase ArchivoRegistrosFijos

La clase ArchivoRegistrosFijos se emplea para la gestión de registros de datos en un archivo de registros fijos. Facilita la realización de operaciones tales como la adición, modificación, recuperación, eliminación, compactación y ordenamiento de registros.

Constructor

“ArchivoRegistrosFijos(String ruta)”

El constructor de la clase ArchivoRegistrosFijos recibe la ruta del archivo en la que se almacenan los registros. Si el archivo no existe en la ubicación especificada, se crea automáticamente con registros de ejemplo.

```
ArchivoRegistrosFijos archivoRegistros = new ArchivoRegistrosFijos("ruta_del_archivo");
```

Métodos

“crearArchivoPorDefecto()”

Crea un archivo por defecto con registros de ejemplo si el archivo no existe en la ubicación especificada.

“agregarRegistro(Registro registro)”

Agrega un nuevo registro al archivo. Verifica si la clave (código) proporcionada ya existe en el archivo y solicita una clave diferente en caso de duplicados.

```
Registro nuevoRegistro = new Registro("0004", "Luisa", "Perez");
archivoRegistros.agregarRegistro(nuevoRegistro);
```

“verificarClaveExistente(String clave)”

Verifica si una clave (código) ya existe en el archivo de registros. Retorna true si la clave ya existe, de lo contrario, retorna false.

“recuperarRegistro(String codigoBuscar)”

Recupera un registro del archivo basado en su código. Retorna un objeto Registro si se encuentra, o null si no se encuentra ningún registro con el código proporcionado.

```
Registro registroRecuperado = archivoRegistros.recuperarRegistro("0002");
```

“modificarRegistro(String codigoModificar, Registro nuevoRegistro)”

Permite modificar un registro existente en el archivo. Reemplaza los datos del registro correspondiente con los nuevos datos proporcionados.

```
Registro nuevoRegistroModificado = new Registro("0002", "Alvaro", "Gomez");
archivoRegistros.modificarRegistro("0002", nuevoRegistroModificado);
```

“mostrarCodigosRegistros()”

Muestra los códigos de registros disponibles en el archivo.

```
archivoRegistros.mostrarCodigosRegistros();
```

“mostrarRegistrosInsertados()”

Muestra todos los registros insertados en el archivo, excluyendo los registros marcados como borrados.

```
gestor.mostrarRegistrosInsertados();
```

“marcarRegistroComoBorrado(String codigoBorrar)”

Marca un registro como borrado en el archivo. Los registros marcados como borrados no se mostrarán en la lista de registros insertados.

```
gestor.marcarRegistroComoBorrado("0003");
```

“mostrarRegistrosMarcadosComoBorrados()”

Muestra los registros marcados como borrados en el archivo.

```
gestor.mostrarRegistrosMarcadosComoBorrados();
```

“compactarArchivo()”

Compacta el archivo eliminando registros marcados como borrados. El archivo original se renombra como "registro.bak" y se crea un nuevo archivo "registro.txt" sin los registros marcados como borrados.

```
gestor.compactarArchivo();
```

“insertarRegistroOrdenado(Registro registro)”

Inserta un registro de manera ordenada por código. Los registros se ordenan alfabéticamente por su código.

```
Registro nuevoRegistroOrdenado = new Registro("0000", "Maria", "Lopez");
archivoRegistros.insertarRegistroOrdenado(nuevoRegistroOrdenado);
```

“handleError(String message, Exception e)”

Método privado para manejar errores y mostrar un mensaje de error en la consola junto con la traza de la excepción.

```
archivoRegistros.handleError("Error al realizar una operación.", e);
```

Clase Registro

La clase Registro representa un registro de datos que contiene un código, nombre y apellido. Es utilizada para modelar y manipular información de registros en un archivo.

Atributos

- codigo (String): El código asociado al registro.
- nombre (String): El nombre asociado al registro.
- apellido (String): El apellido asociado al registro.

Constructor

“Registro(String codigo, String nombre, String apellido)”

El constructor de la clase Registro recibe un código, nombre y apellido para crear un nuevo objeto de registro.

```
Registro nuevoRegistro = new Registro("0001", "Juan", "Pérez");
```

Métodos

“getCodigo()”

Devuelve el código del registro.

```
String codigo = registro.getCodigo();
```

“setCodigo(String codigo)”

Establece el código del registro.

```
registro.setCodigo("0002");
```

“getNombre()”

Devuelve el nombre del registro.

```
String nombre = registro.getNombre();
```

“getApellido()”

Devuelve el apellido del registro.

```
String apellido = registro.getApellido();
```

“parse(String linea)”

Método estático que permite crear un objeto Registro a partir de una línea de texto en un formato específico. La línea de texto debe contener código, nombre y apellido, separados por espacios. Retorna un objeto Registro si el formato es válido, de lo contrario, retorna null.

```
String linea = "0001    Juan          Pérez                    ";
Registro registro = Registro.parse(linea);
```

“toString()”

Devuelve una representación en formato de cadena del registro que incluye el código, nombre y apellido.

```
String representacion = registro.toString();
```

Ejemplo de Uso

```
Registro nuevoRegistro = new Registro("0001", "Juan", "Pérez");
String codigo = nuevoRegistro.getCodigo();
String nombre = nuevoRegistro.getNombre();
String apellido = nuevoRegistro.getApellido();

String linea = "0001    Juan          Pérez                    ";
Registro registro = Registro.parse(linea);

String representacion = registro.toString();
```

Clase Indice

La clase Indice se utiliza para representar un índice que mapea una clave a una posición en un archivo de registros.

Atributos

- clave (String): La clave que se mapea al registro.
- posicion (long): La posición en el archivo donde se encuentra el registro correspondiente a la clave.

Constructores

"Indice()"

Constructor por defecto que no recibe argumentos. No realiza ninguna acción especial.

"Indice(String clave, long posicion)"

Constructor que recibe una clave y una posición para crear un nuevo objeto Indice.

```
Indice indice = new Indice("0001", 12345);
```

Métodos

"getClave()"

Devuelve la clave del índice.

```
String clave = indice.getClave();
```

“setClave(String clave)”

Establece la clave del índice.

```
indice.setClave("0002");
```

“setPosicion(long posicion)”

Establece la posición del índice.

```
indice.setPosicion(54321);
```

Ejemplo de Uso

```
Indice indice = new Indice("0001", 12345);
String clave = indice.getClave();
long posicion = indice.getPosicion();

indice.setClave("0002");
indice.setPosicion(54321);
```