Taller 6 - Manejo de excepciones
ISIS-1226

David Valderrama Herrera
Martín Daniel Rincón Molina

Punto 1

Para el manejo del error se propuso incluir un nuevo atributo en la clase Librería, llamado *categorias_nuevas*, de tipo *ArrayList<Categoria>*. Si dentro del ciclo *while* del método *cargarCatalogo()* la categoría retornada por *buscarCategora(nombreCategoria)* es nula, significa que la categoría no existe. En este caso se crea una nueva categoría bajo el nombre de *nombreCategoria* y además de guardarla en la lista *categorias*, se le guarda también en *categorias_nuevas*; luego continua el flujo regular del método.

Luego de haberse cargado todos los libros, la interfaz pide a la libreria recien creada el valor de *categorias_nuevas*. Si *categorias_nuevas* no es un ArrayList vació, entonces se tuvieron que haber añadido categorías nuevas. En este caso, la interfaz muestra el nombre de las categorías recibidas en *categorias_nuevas* al usuario.

El código de la implementación se muestra a continuación:

```
1 if (laCategoria==null) {
2     Categoria cat_new=new Categoria(nombreCategoria,false);
3     ArrayList<Categoria> categorias_old = new ArrayList<Categoria>(
          Arrays.asList(categorias));
4     categorias_old.add(cat_new);
5     this.categorias_nuevas.add(cat_new);
6     categorias=categorias_old.toArray(categorias);
7     laCategoria=cat_new;
8 }
```

Punto 2

En la interfaz se crea el método *RenombrarCategoria()*, el cual se encargara en ultima instancia de manejar la excepción, informándole al usuario del problema.

Adicionalmente, este método cuida que el usuario no cambie el nombre de una categoría para poner el mismo. Ante cualquier excepción diferente a la esperada, el método imprime en consola el *TrackTrace* de la excepción:

```
1 public void RenombrarCategoria(){
2         String NombreViejo=JOptionPane.showInputDialog("Nombre de la
              categoria a cambiar:");
3         String NombreNuevo=JOptionPane.showInputDialog("Nombre nuevo");
4         if(NombreNuevo.equals(NombreViejo)) {
```

```
5               JOptionPane.showMessageDialog(this,"El nombre viejo y el
                    nuevo no pueden ser iguales","Cuidado",JOptionPane.
                    ERROR_MESSAGE);
6           }else {
7               try {
8                   libreria.RenombrarCategoria(NombreViejo, NombreNuevo);
9               }catch(ExceptionCategoriaRepetida ec) {
10                  JOptionPane.showMessageDialog(this,ec.getMessage(),"
                        Error",JOptionPane.ERROR_MESSAGE);
11              }catch(Exception e) {
12                  e.printStackTrace();
13              }
14          }
15  }
```

Desde *RenombrarCategoria()* en la interfaz se invoca al método de la Librería con el mismo nombre. Este método verifica que no exista previamente una categoría con el nombre nuevo y busca la librería a la que se le quiere cambiar el nombre. Si hay una categoría con el nombre nuevo, entonces arroja una *ExceptionCategoriaRepetida* al método invocador en la interfaz para que este último la trate.

Si no hay ningún problema, entonces se le cambia el nombre a la categoría:

```
1   public void RenombrarCategoria(String oldName, String newName) throws
        ExceptionCategoriaRepetida {
2
3           boolean existe = false;
4           Categoria cat=null;
5           int i=0;
6           while(!existe && i<this.categorias.length) {
7               if(categorias[i].darNombre().equals(newName)) {
8                   existe=true;
9               }
10              if(categorias[i].darNombre().equals(oldName)) {
11                  cat=categorias[i];
12              }
13              i++;
14          }
15          if(existe) {
```

```
16              throw new ExceptionCategoriaRepetida();
17          }
18          cat.CambiarNombre(newName);
19  }
```

Punto 3

En la interfaz se crea el método *eliminarLibros()*, el cual se encargará de gestionar la eliminación de los libros de acuerdo al actor. Este método se vale del elemento gráfico JOptionPane para interactuar con el usuario, lo cual incluye: pedirle la lista de autores e informarle del estado de la operación. en ultima instancia de manejar la excepción, informándole al usuario del problema.

Adicionalmente, este método se conecta con la función *eliminarLibros(String[] autores)* que tiene la lógica que finalmente permite la eliminación en memoria principal.

```
1  public void eliminarLibros() {
2
3          String grupoAutores = JOptionPane.showInputDialog("Grupo de
               autores: ");
4          String[] arrayStr = grupoAutores.split(",");
5
6          try {
7              int resp = libreria.eliminarLibros(arrayStr);
8              System.out.println(resp);
9              JOptionPane.showMessageDialog(this, "Operaci n exitosa, "
                   + String.valueOf(resp) + " libro(s) eliminado(s)", "
                   Operaci n exitosa", JOptionPane.INFORMATION_MESSAGE);
10         }
11
12         catch(ExcEliminarLibros ec) {
13             JOptionPane.showMessageDialog(this, ec.getMessage(), "Error
                   ", JOptionPane.ERROR_MESSAGE);
14         }
15     }
```

Una vez con todos los procedimientos gestionados desde la interfaz, el método *eliminarLibros(String[] autores)* presente en la clase Librería se encarga de disparar una excepción

específicamente definida para este requerimiento (ExcEliminarLibros). Para implementar esta función fue necesario designar un método en la clase Categoría para eliminar libros.

```
public int eliminarLibros(String[] autores) throws ExcEliminarLibros {

    int cant = 0;
    boolean trigger = false;

    ArrayList<String> noPosibles = new ArrayList<String>();
    for (String autor : autores) {
        ArrayList<Libro> librosAutor = buscarLibrosAutor(autor);
        if (librosAutor.isEmpty()) {
            noPosibles.add(autor);
            trigger = true;
        }
    }

    if (trigger == false) {
        for (int i = 0; i < autores.length; i++) {
            ArrayList<Libro> librosAutor = buscarLibrosAutor(autores[i
               ]);
            cant += librosAutor.size();
            for (Libro libro: librosAutor) {
                Categoria categoria = libro.darCategoria();
                catalogo.remove(libro);
                categoria.quitarLibro(libro);
            }
        }
    }

    else if (trigger = true) {
        throw new ExcEliminarLibros(noPosibles);
    }

    return cant;
}
```