```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Configuration;

using Nihulon2.Model;
using Nihulon2.Model.DbAccess;

namespace Nihulon2.RelationsList
{

    /* The class that assumes the layer between the view and DataBase.
    * It provides methods that get data from the dbConnector and
    * calls the view methods to fill its controls with the new data.
    * This controller works with the items related to the exam
    * like divisions, courses and rooms */
    public class RelationsList_Controller
    {
        private IRelationsListView _view; // An instance of the view for data visualization
        private DbConnector dbConn; // An instance of the class that provides connection to the DB
        private RelatedItem[] relatedItems; // Items that are in the data grid of RelationsList_View
        private string selectedType; // The type that is selected in the combo box of the view

        // Defines if disabled related items are going to be shown
        public bool ShowDisabled
        {
            get;
            set;
        }

        /* The constructor. It initializes instances of the dbConnector and
        * the view classes, and sets default values for flags and comboBoxes
        * view - The instance of a view that will work with this controller */
        public RelationsList_Controller(IRelationsListView view)
        {
            // Initializing connection to DB
            dbConn = DbConnector.Instance;

            ShowDisabled = false; // Don't show disabled related items by default

            // Binding with view
            _view = view;
            _view.setController(this);
            // set divisions as default related items and fill the data grid of the view
            selectedType = "חטיבות";
            _view.setRelatedItemsType(selectedType);
        }
        #region Interface

        /* Gets array of items from DB according to the type and
        * calls the method of the view that fills grid with the items
        * itemType - The type of items that will be loaded from the DB */
        public void loadDataByType(string itemType)
        {
            selectedType = itemType;
            /* Get an array of related items from DB
               itemType - The type of needed items (Divisions, Courses, Rooms)
               ShowDisabled - True or false. If we need the array including disabled items*/
            relatedItems = dbConn.GetRelatedItemsByType(itemType, ShowDisabled);

            _view.clearDataGrid();
            _view.fillDataGrid(relatedItems);
        }


        /* Adds new related item to the DB
        * newItemName - The name of the new item */
        public void addNewItem(string newItemName)
        {
            try
            {
                if(!string.IsNullOrEmpty(newItemName))
                    dbConn.insertRelatedItem(newItemName, selectedType); // Insert the new item
            }
            catch { };

            loadDataByType(selectedType); // reload the data grid after the new item was inserted
        }

        // Takes a name of item and changes its status (disabled or not) at the DB and reload the data grid
        public void changeStatusOfItem(string nameOfItem)
        {
            if(!string.IsNullOrEmpty(nameOfItem)) // if the instance was found
            {
                dbConn.changeStatusToRelatedItem(nameOfItem, selectedType);
                loadDataByType(selectedType); // reload the data grid after the item was changed
            }
        }

        // Get data from the DB and reload the view
        public void reload()
        {
            this.loadDataByType(selectedType);
        }
        #endregion
    }
}
```