```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;


using Nihulon2.Model;
using System.Collections;
using DGVPrinterHelper;
using System.IO;

namespace Nihulon2.SupervisorsAdministration
{
    /*
     * The class of view that displays the data of exams
     * ISupervisorsAdministrationView - interface that provides methods to the controller
     * for displaying the data
     */
    public partial class SupervisorAdministration_View : UserControl, ISupervisorsAdministrationView
    {
        // Instance of the controller that manages this view
        private SupervisorsAdministration_Controller _controller;

        public SupervisorAdministration_View()
        {
            InitializeComponent();

            radioButtonShowAll.Checked = true; // set "show all exams" as a default filter
        }

        #region Form Events

        // Adding a new exam
        private void btnAddExam_Click(object sender, EventArgs e)
        {
            // If all required fields at the form are filled
            if(cboCourse_NewExam.SelectedIndex != -1 &&
                cboDivision_NewExam.SelectedIndex != -1 && cboRoom_NewExam.SelectedIndex != -1)
            {
                // Create a new exam and fill its fields
                Exam newExam = new Exam();
                newExam.course = cboCourse_NewExam.Text;
                newExam.Date = DatePicker.Text;
                newExam.DisciplineName = txtDiscipline_NewExam.Text;
                newExam.division = cboDivision_NewExam.Text;
                newExam.EndingTime = ToTimePicker.Text;
                newExam.GroupName = txtGroup_NewExam.Text;
```

```
                newExam.hasExtraTime = cbExtraTime_NewExam.Checked;
                newExam.room = cboRoom_NewExam.Text;
                newExam.StartTime = FromTimePicker.Text;
                newExam.SupervisorName = txtSupervisor_NewExam.Text;
                newExam.dateOfCreation = DateTime.Today.ToString
                  ("dd.MM.yyyy");

                // Add the new exam to the list of exams and insert into the
                  DB
                _controller.addExam(newExam);

                clearAddingForm();
            }
        }

        // Clear the adding form
        private void btnClearAddingForm_Click(object sender, EventArgs e)
        {
            clearAddingForm();
            // Disable the button of clearing the form
            btnClearAddingForm.Enabled = false;
        }

        // Create a copy of selected exam
        private void btn_CopyRow_Click(object sender, EventArgs e)
        {
            // get selected exam as an instance of Exam
            Exam selectedExam = dgvExamTable.CurrentRow.DataBoundItem as Exam;

            if(selectedExam != null)
                _controller.addExam(selectedExam); // Add exam
        }

        // Delete an exam
        private void btnDeleteExam_Click(object sender, EventArgs e)
        {
            int examId;
            Exam exam;
            // Go through all selected exams and delete them
            foreach (DataGridViewRow row in dgvExamTable.SelectedRows)
            {
                exam = row.DataBoundItem as Exam;
                if(exam != null)
                {
                    examId = exam.Id; // get exam id from selected exam
                    _controller.deleteExam(examId); // delete
                }
            }
            // Get the updated data from the DB
            _controller.fillTable();
        }

        // Open the form of changing an exam by click
        // on the button "change exam"
        private void btnChangeExam_Click(object sender, EventArgs e)
        {
```

```
            this.callFormOfExamChanging();
        }
        // Open the form of changing an exam by double click on the exam at the table
        private void dgvExamTable_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
        {
            this.callFormOfExamChanging();
        }

        // When one of the checkboxes, "canceled" or "has extra time" changes its state,
        // change the exam state and save into the DB
        private void dgvExamTable_CellValueChanged(object sender, DataGridViewCellEventArgs e)
        {
            Exam changedExam = dgvExamTable.CurrentRow.DataBoundItem as Exam;
            if(changedExam != null)
                _controller.changeExam(changedExam);
        }

        // When a row in the table selected, fill textbox with the comments of the selected exam
        // If there is multi-selected rows, disable the buttons of changing and copying
        private void dgvExamTable_SelectionChanged(object sender, EventArgs e)
        {
            // If there is selected row at the table, fill the text box with comments
            if (dgvExamTable.CurrentRow != null)
            {
                Exam exam = dgvExamTable.CurrentRow.DataBoundItem as Exam;
                if (exam != null)
                    txt_Comments.Text = exam.Comments;
            }


            // If there are more than one selected row, disable the buttons
            // of changing and copying exams
            if(dgvExamTable.SelectedRows.Count != 1)
            {
                btnChangeExam.Enabled = false;
                btn_CopyRow.Enabled = false;
            }
            else
            {
                btnChangeExam.Enabled = true;
                btn_CopyRow.Enabled = true;
            }
        }

        // Save changes after editing the comments
        private void txt_Comments_Leave(object sender, EventArgs e)
        {
            // get selected exam
            Exam changedExam = (dgvExamTable.CurrentRow.DataBoundItem as
```

```
                Exam);

            // If casted successful
            if(changedExam != null)
            {
                changedExam.Comments = txt_Comments.Text; // set new comments
                _controller.changeExam(changedExam); // change into the DB
            }
        }

        /*
         * When one of the radiobuttons at the time filters was checked,
         * enable the needed option and disable others
         */
        // Show all
        private void radioButtonShowAll_CheckedChanged(object sender, EventArgs e)
        {
            dateTimePickerOneDay_Filter.Enabled = false;
            dateTimePickerFrom_Filter.Enabled = false;
            dateTimePickerTo_Filter.Enabled = false;
        }
        // Period
        private void radioButtonPeriod_CheckedChanged(object sender, EventArgs e)
        {
            dateTimePickerOneDay_Filter.Enabled = false;
            dateTimePickerFrom_Filter.Enabled = true;
            dateTimePickerTo_Filter.Enabled = true;
        }
        // One day
        private void radioButtonOneDay_CheckedChanged(object sender, EventArgs e)
        {
            dateTimePickerOneDay_Filter.Enabled = true;
            dateTimePickerFrom_Filter.Enabled = false;
            dateTimePickerTo_Filter.Enabled = false;
        }

        // When the button "show exams" has been pressed,
        // set all values of the filters at the controller and ask to get the needed data
        private void btnShowExams_Click(object sender, EventArgs e)
        {
            /* Set the filters of period of time */

            // show all or show new
            if (radioButtonShowAll.Checked == true || radioButtonShowNew.Checked == true)
            {
                _controller.dateFromFilter = "";
                _controller.dateToFilter = "";
            }
            // show exams for one day
            else if (radioButtonOneDay.Checked == true)
            {
```

```
                _controller.dateFromFilter = dateTimePickerOneDay_Filter.Text;
                _controller.dateToFilter = dateTimePickerOneDay_Filter.Text;
            }
            // show exams for a period
            else if (radioButtonPeriod.Checked == true)
            {
                _controller.dateFromFilter = dateTimePickerFrom_Filter.Text;
                _controller.dateToFilter = dateTimePickerTo_Filter.Text;
            }

            // Set other filters from the form to the controller
            _controller.divisionFilter = cboDivisionFilter.Text;
            _controller.courseFilter = cboCourseFilter.Text;
            _controller.roomFilter = cboRoomFilter.Text;
            _controller.showDisabledFilter = cbShowDisabledExams.Checked;
            _controller.showNewFilter = radioButtonShowNew.Checked;
            _controller.showOverlaps = false;

            // Ask the controller to fill the table according to the filters
            _controller.fillTable();
        }

        // If one of the dates at the filter form was changed,
        // check if the "From" date is earlier than the "To" date.
        // If not, set the "To" date equal to the "From" date
        private void dateTimePickerFilter_ValueChanged(object sender, EventArgs e)
        {
            this.checkAndSetFilterDates(dateTimePickerTo_Filter, dateTimePickerFrom_Filter);
        }

        // The method that is called when something has been changed at the form
        // of adding a new exam. It makes the button "clear the form" being enabled
        private void creationFormChanged(object sender, EventArgs e)
        {
                btnClearAddingForm.Enabled = true;
        }

        // Printing the Exam table
        // The method uses DGVPrinter module that serves
        // the printing of data Grid Views
        private void btn_Print_Click(object sender, EventArgs e)
        {
            DGVPrinter printer = new DGVPrinter();

            // Settings of the printing
            printer.HeaderCellAlignment = StringAlignment.Near;
            printer.PageSettings.Landscape = true;
            printer.PageSettings.Color = false;
            printer.PageSettings.Margins.Left = 30;
            printer.RowHeight = DGVPrinter.RowHeightSetting.CellHeight;

            // Print
```

```
            printer.PrintDataGridView(dgvExamTable);
        }

        // Create a form for taking the name of Excel file
        private void btn_ExportToExcel_Click(object sender, EventArgs e)
        {
            frmGetNameOfExcelFile getNameForm = new frmGetNameOfExcelFile();
            // Event called when user entered the name of the file
            getNameForm.onFileNameConfirm += excelNameEntered;

            getNameForm.StartPosition = FormStartPosition.CenterParent;
            getNameForm.ShowDialog();
        }
        // Loads exams from excel
        private void btn_ImportFromExcel_Click(object sender, EventArgs e)
        {
            // Create a folder for Excel files if not exists
            string location = System.Windows.Forms.Application.StartupPath;
            location += @"\Excel files";
            if (!Directory.Exists(location))
            {
                Directory.CreateDirectory(location);
            }

            // Open the file dialog from the folder with Excel files
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.InitialDirectory = location;
            openFileDialog.Filter = "Excel files (*.xlsx)|*.xlsx|All files (*.*)|*.*";

            if (openFileDialog.ShowDialog() == DialogResult.OK && openFileDialog.CheckFileExists)
            {
                _controller.loadFromExcel(openFileDialog.FileName);
            }
        }

        // Checks each exam if it has time overlaps and paint its row red
        private void dgvExamTable_DataBindingComplete(object sender, DataGridViewBindingCompleteEventArgs e)
        {
            foreach (DataGridViewRow dgvr in dgvExamTable.Rows)
            {
                if (Convert.ToBoolean(dgvr.Cells["hasOverlap"].Value))
                    dgvr.DefaultCellStyle.BackColor = Color.IndianRed;
            }
        }

        // Show only the exams that have time overlaps
        private void btnOverlaps_Click(object sender, EventArgs e)
        {
            _controller.showOverlaps = true;

            _controller.fillTable();
        }

```

```
        // Runs searching of exams with time overlaps
        private void btnCheckOverlaps_Click(object sender, EventArgs e)
        {
            _controller.checkOverlaps();
        }

        #endregion

        #region Interface
        // binds the view with its controller
        public void setController(SupervisorsAdministration_Controller controller)
        {
            _controller = controller;
        }

        // Binds the list of exams at the controller with the table at the view
        public void bindExamsWithTable(ref SortableBindingList<Exam> examsList)
        {
            dgvExamTable.DataSource = examsList;   // bind
            this.formatTable(); // set column sizes
        }

        // Fill the comboboxes of Divisions at the form of filters and at the form of adding
        public void fillDivisionsCbo(string[] divisionsNames)
        {
            cboDivisionFilter.Items.Add("הכול");
            // Fill the comboboxes with names
            for (int i = 0; i < divisionsNames.Length; i++)
            {
                cboDivisionFilter.Items.Add(divisionsNames[i]);
                cboDivision_NewExam.Items.Add(divisionsNames[i]);
            }
            // Set "show all exams" as selected
            cboDivisionFilter.SelectedIndex = 0;
        }
        // Fill the comboboxes of Courses at filters form and at the adding form
        public void fillCoursesCbo(string[] coursesNames)
        {
            cboCourseFilter.Items.Add("הכול");
            // Fill the comboboxes with names
            for (int i = 0; i < coursesNames.Length; i++)
            {
                cboCourseFilter.Items.Add(coursesNames[i]);
                cboCourse_NewExam.Items.Add(coursesNames[i]);
            }
            // Set "show all exams" as selected
            cboCourseFilter.SelectedIndex = 0;
        }
        // Fill the comboboxes of Rooms at filters form and at the adding form
        public void fillRoomsCbo(string[] roomsNames)
        {
```

```
            cboRoomFilter.Items.Add("הכול");
            // Fill the comboboxes with names
            for (int i = 0; i < roomsNames.Length; i++)
            {
                cboRoomFilter.Items.Add(roomsNames[i]);
                cboRoom_NewExam.Items.Add(roomsNames[i]);
            }
            // Set "show all exams" as selected
            cboRoomFilter.SelectedIndex = 0;
        }

        // Clear all combo boxes
        public void clearComboBoxes()
        {
            cboCourseFilter.Items.Clear();
            cboDivisionFilter.Items.Clear();
            cboRoomFilter.Items.Clear();
            cboCourse_NewExam.Items.Clear();
            cboDivision_NewExam.Items.Clear();
            cboRoom_NewExam.Items.Clear();
        }

        // Get updated data from the DB and reload the view
        public void reloadView()
        {
            _controller.reload();
        }

        // Set visible or hide the button and the label that show the overlaps at the table
        public void showOverlapsWarning(bool hasOverlaps)
        {
            if (hasOverlaps == true && btnOverlaps.Visible == false)
            {
                btnOverlaps.Visible = true;
                labelOverlaps.Visible = true;
            }
            else if (hasOverlaps == false)
            {
                btnOverlaps.Visible = false;
                labelOverlaps.Visible = false;
            }
        }

        #endregion

        #region Private methods

        // Clear txt fields and checkbox at the form for adding exams
        private void clearAddingForm()
        {
            txtDiscipline_NewExam.Clear();
            txtGroup_NewExam.Clear();
            txtSupervisor_NewExam.Clear();
            cboRoom_NewExam.SelectedIndex = -1;
            cboCourse_NewExam.SelectedIndex = -1;
```

```
            cboDivision_NewExam.SelectedIndex = -1;

            cbExtraTime_NewExam.Checked = false;
        }

        // Set column sizes and hide columns that user doesn't need
        // like Id, date of creation etc.
        private void formatTable()
        {
            // hide the columns that user doesn't need
            dgvExamTable.Columns["comments"].Visible = false;
            dgvExamTable.Columns["id"].Visible = false;
            dgvExamTable.Columns["dateOfCreation"].Visible = false;
            dgvExamTable.Columns["hasOverlap"].Visible = false;

            // set fixed size for not textual columns
            dgvExamTable.Columns["Date"].Width = 80;
            dgvExamTable.Columns["isCanceled"].Width = 70;
            dgvExamTable.Columns["hasExtraTime"].Width = 70;
            dgvExamTable.Columns["startTime"].Width = 70;
            dgvExamTable.Columns["endingTime"].Width = 70;
            dgvExamTable.Columns["GroupName"].Width = 70;
            dgvExamTable.Columns["room"].Width = 100;
            dgvExamTable.Columns["division"].Width = 100;

            // fill the entire space with textual columns
            dgvExamTable.Columns["SupervisorName"].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
            //dgvExamTable.Columns["division"].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
            dgvExamTable.Columns["course"].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
            //dgvExamTable.Columns["GroupName"].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
            dgvExamTable.Columns["DisciplineName"].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
            //dgvExamTable.Columns["room"].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

            // forbid editing for all columns excepting "canceled" and "has extra time"
            dgvExamTable.Columns["Date"].ReadOnly = true;
            dgvExamTable.Columns["startTime"].ReadOnly = true;
            dgvExamTable.Columns["endingTime"].ReadOnly = true;
            dgvExamTable.Columns["SupervisorName"].ReadOnly = true;
            dgvExamTable.Columns["division"].ReadOnly = true;
            dgvExamTable.Columns["course"].ReadOnly = true;
            dgvExamTable.Columns["GroupName"].ReadOnly = true;
            dgvExamTable.Columns["DisciplineName"].ReadOnly = true;
            dgvExamTable.Columns["room"].ReadOnly = true;
        }

        // Checks if the date "From" is earlier than the "To" date
        // If not, sets the "To" date equal to the "From" date
        private void checkAndSetFilterDates(DateTimePicker dateTimePickerTo, DateTimePicker dateTimePickerFrom)
```

```
        {
            if (dateTimePickerTo.Value < dateTimePickerFrom.Value)
            {
                dateTimePickerTo.Value = dateTimePickerFrom.Value;
            }
        }

        // Event handler that is called when the user enter
        // the name of the excel file for saving the table of exams
        private void excelNameEntered(string excelName)
        {
            _controller.createExcel(excelName);
        }


        // Called when one of the time pickers at the form of creating
        // new exams has been changed.
        // Checks if the "from" date is earlier than "to" date
        private void TimePicker_ValueChanged(object sender, EventArgs e)
        {
            checkAndSetFilterDates(ToTimePicker, FromTimePicker);
        }

        // Get changed exam and save the changes into the DB
        private void saveChangedExam(Exam changedExam)
        {
            _controller.changeExam(changedExam);
        }

        // Creates the form of changing exams,
        // fills its controls and set the exam
        // that is needed to be changed
        private void callFormOfExamChanging()
        {
            // get selected exam as an instance of Exam
            Exam selectedExam = (Exam)dgvExamTable.CurrentRow.DataBoundItem;

            // Show form for changing exams
            frmChangeExam changeForm = new frmChangeExam();
            // Subscribe on exam changes
            changeForm.onSaveChanges += saveChangedExam;
            // Fill the comboBoxes of the changeForme with related items
            changeForm.fillComboBoxes(cboDivision_NewExam.Items,
              cboCourse_NewExam.Items, cboRoom_NewExam.Items);
            // Set the exam that needed to be changed
            changeForm.setExam(selectedExam);

            // Show the form at the center
            changeForm.StartPosition = FormStartPosition.CenterParent;
            changeForm.ShowDialog();
        }

        #endregion
    }
}
```