Collège LaSalle
Montréal

Course Number: 420-P34-AS
Course Title: Advanced Object Programming
Session: Autumn 2016

# FINAL PROJECT REPORT

## TITLE: HI-TECH MANAGEMENT SYSTEM

**DUE DATE: 01/12/2016**

**SUBMITTED**

**TO**

**TEACHER: QUANG HOANG CAO**

**BY**

**CLAUDIA LISBOA - 1531342**

# I. PROJECT DESCRIPTION

1. A short description of the project
   With this project the fictitious company, Hit-Tech Distribution Inc., is able to supply computer science books and softwares to all the Colleges and Universities in Quebec.

2. Users and Operations

| User | Operations |
|---|---|
| MIS Manager<br>(Henry Brown) | • Save/update/delete employee information<br>• Search/list employee information<br>• Save/update/delete user information<br>• Search/List user information |
| Sales Manager<br>(Thomas Moore) | •Save/Update/Delete client information<br>• Search/List client information |
| Inventory Controller<br>(Peter Wang) | • Save/update/delete product information<br>• Search/List product information |
| Order Clerks<br>- Mary Brown<br>- Jennifer Bouchard | • Save/Update/Cancel clients' orders<br>• Search/List clients' orders |

3. Technologies used to develop the application
   It was developed in C# in Visual Studio 2015 according to the object oriented programming principles.

# II. PROJECT DEVELOPMENT PROCESS

1. **Analysis**: To access the system, each user is required to enter his/her valid username and password. The user can change the password when necessary.

2. **Design**:
   GUI: Book Form, Change Password Form, Client Form Employee Form, Login Form, Main Form, Order Form, Product Form, Select Product Form, Software Form, User Form.

   Application Domain (Business) : Author, AuthorCollection, Book, BooksCollection, CategoriesColletion, Category, Client, ClientsCollection, Employee, EmploeesCollection, GenericList, Order, OrderLine, Paument, Permission, PermissionsCollection, Publisher, Software, SoftwaresCollection, Supplier, SuppliersCollection, User, UsersCollection.

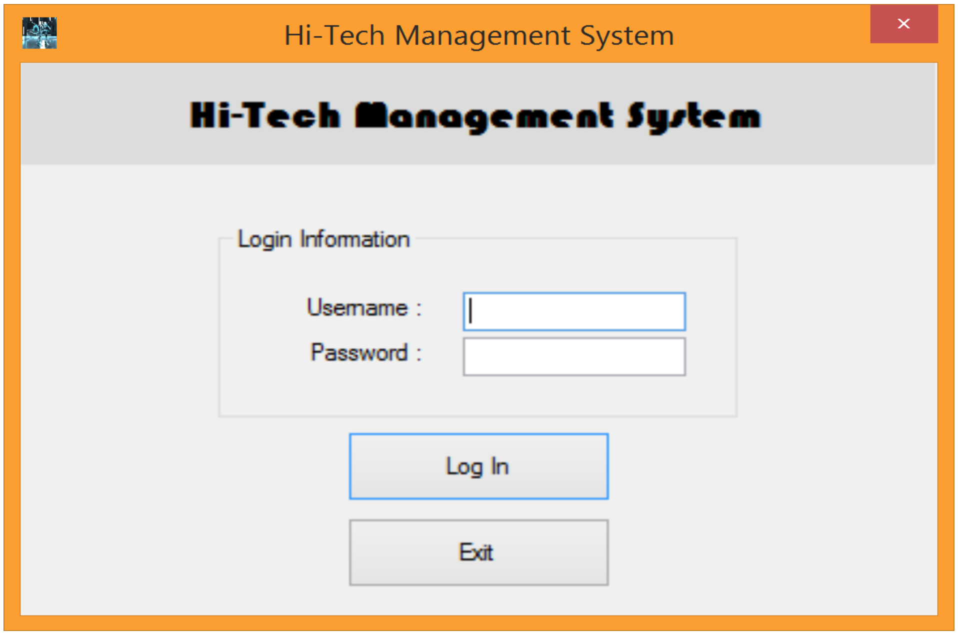
   Data Access Classes : AuthorDataAccess, BookDataAccess, CategoryBookDataAccess, GategorySoftwareDataAccess, ClientDataAccess, EmployeeDataAccess, GenericDataAccess, HiTechDBDataAccess, OrderDataAccess, OrderLineDataAccess, PaymentsDataAccess, PermissionDataAccess, ProductDataAccess, PublisherDataAccess, SoftwareDataAccess, SupplierDataAccess, UserDataAccess.
3. **Implementation**: Source Code Listings
4. **Testing**: Test results of the application in well-defined table format
5. **Deploying the Application**: Hardware and software requirements for the application : 16GB RAM, minimum 6 Gb of free disk and Windows 8 or 10

# III. CONCLUSION

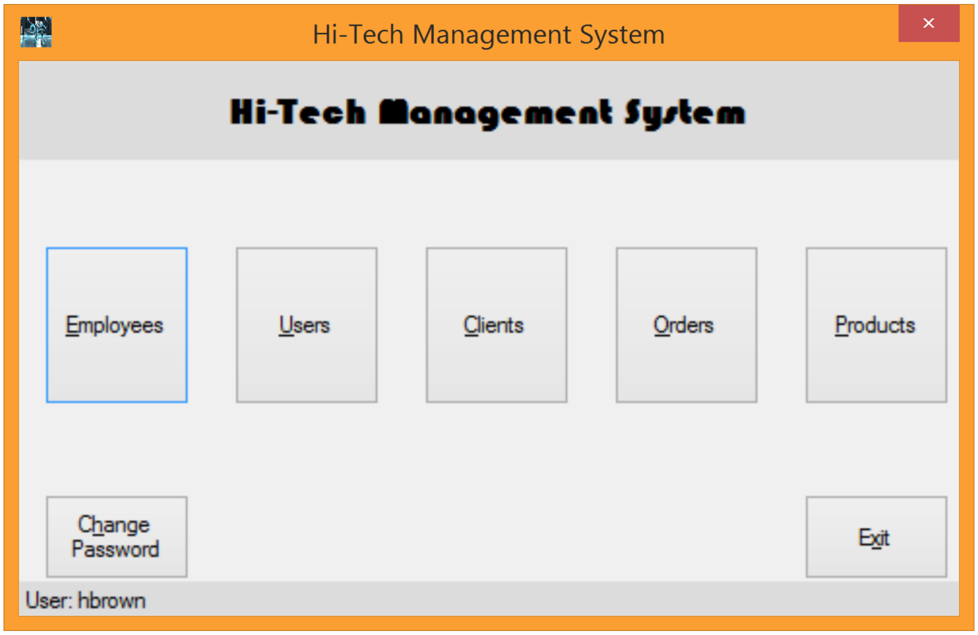
This project allowed me to put in practice and test the various software development techniques learned during the session.

The operation consists of a set of tasks that requires validations and system requirements from users. The purpose of this application is to provide convenience for the Company. It requires support strictly by developers. However, using classes, we can see how could be ease to implement and also reuse.

**When the user starts the system the login window is presented, so he/she can provide the credentials to have access to the system.**

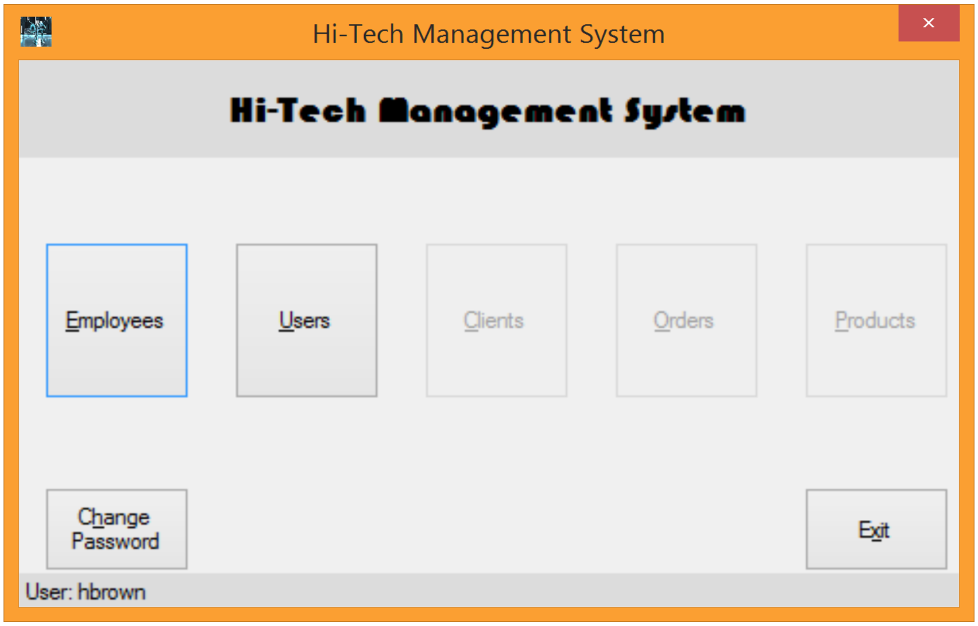


**This is the main application window with all modules available.**



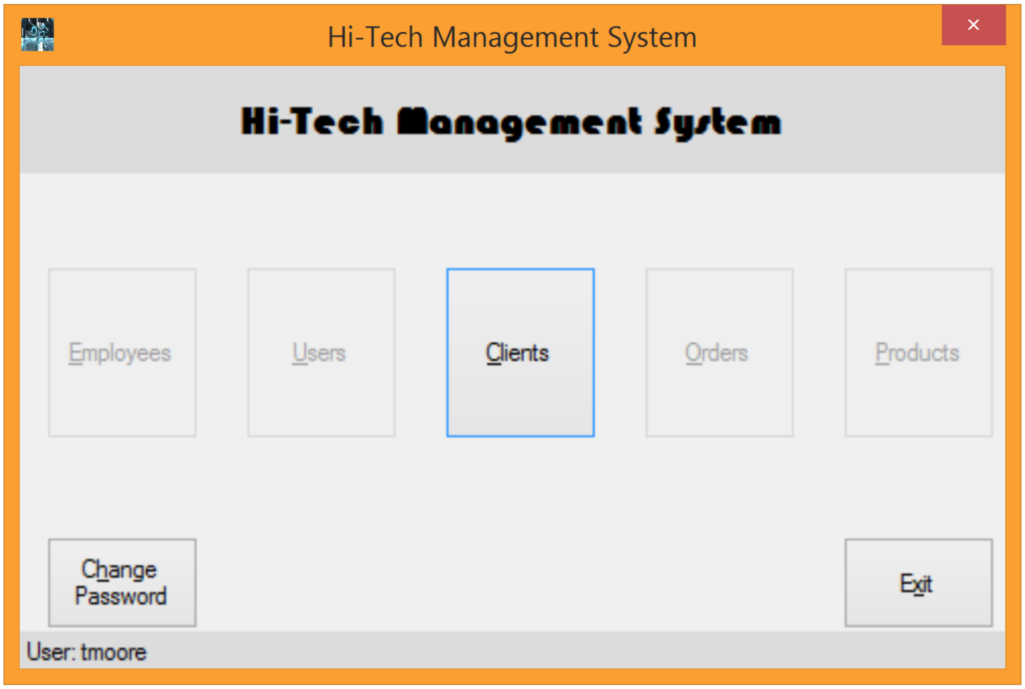
User: hbrown

**This is the main window showing the modules available for the MIS Manager.**



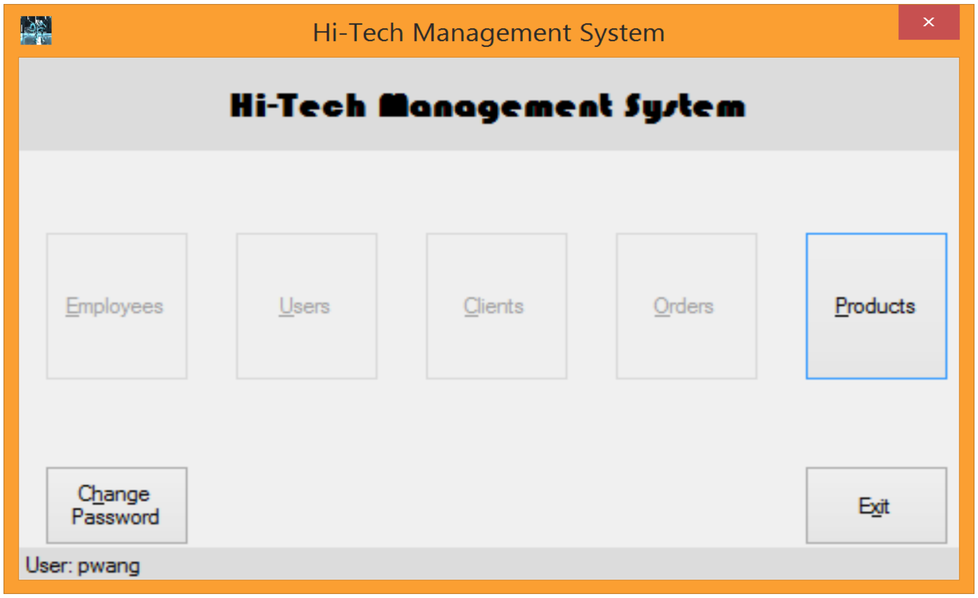
User: hbrown

**This is the main window showing the module available for the Sales Manager.**



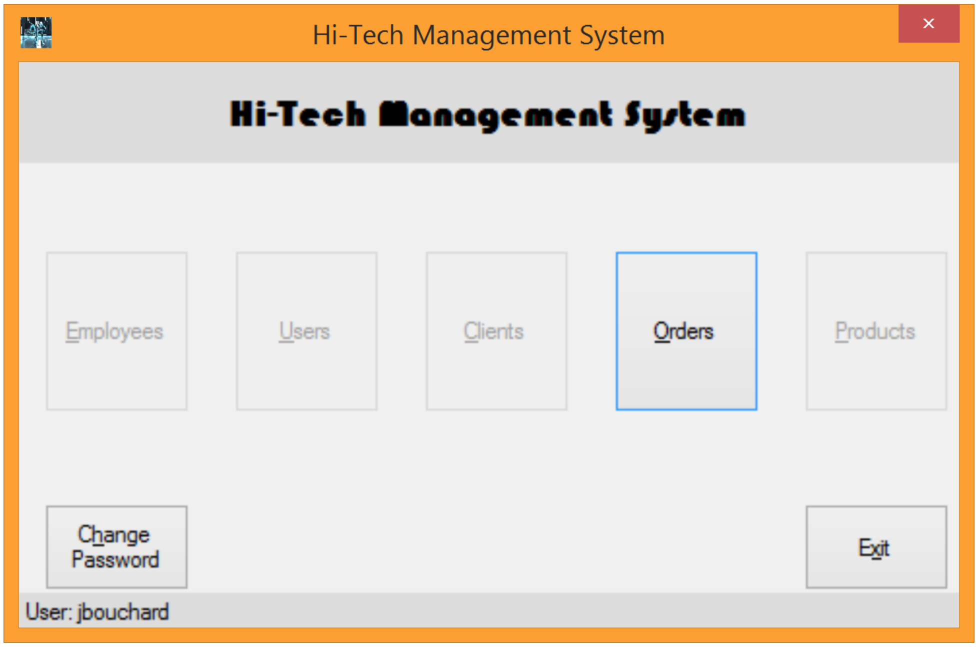
User: tmoore

**This is the main window showing the module available for the Inventory Controller.**

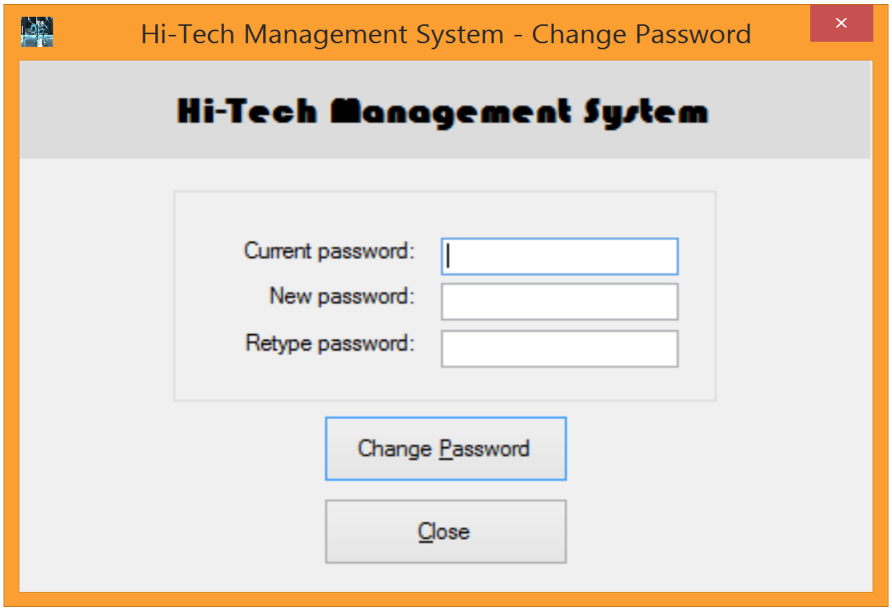


User: pwang

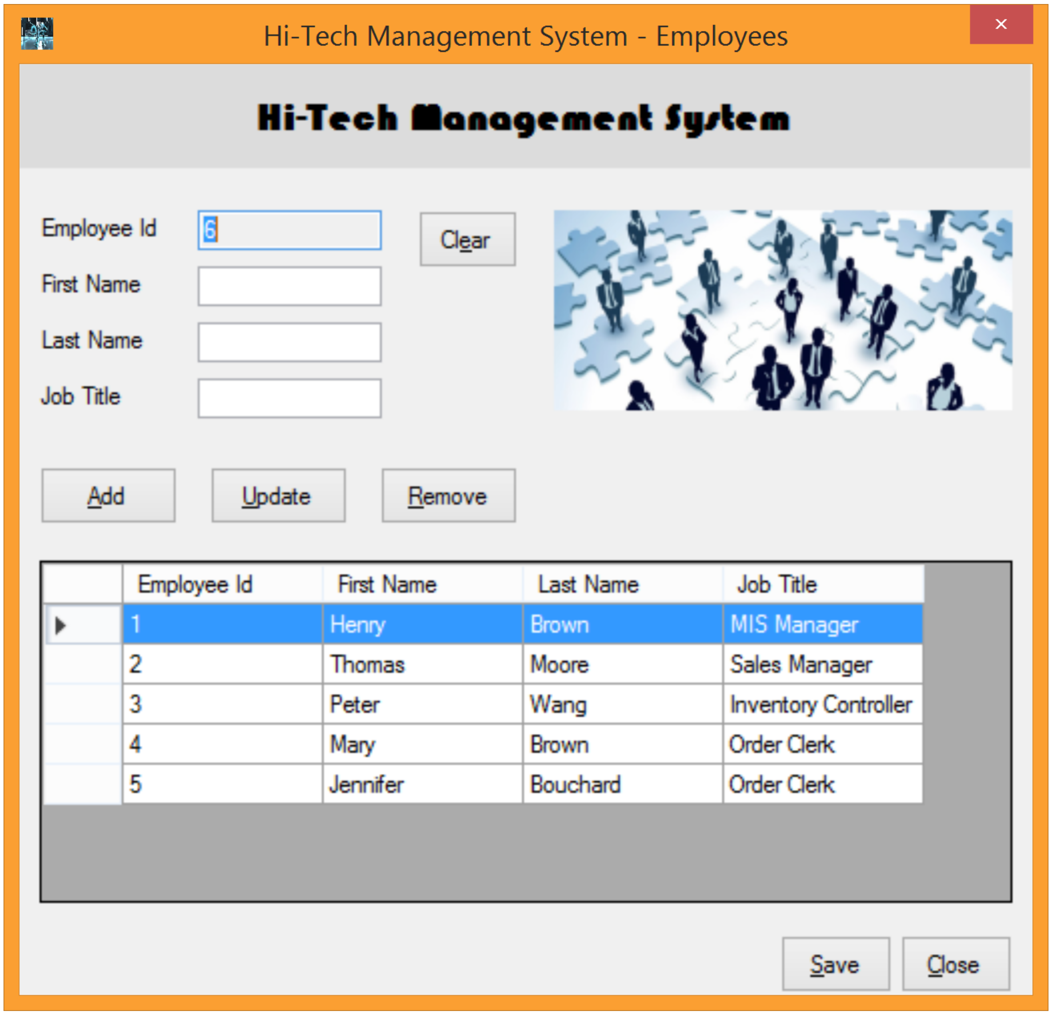
**This is the main window showing the module available for Order Clerks.**



**When necessary the user can change the password.**


Hi-Tech Management System - Change Password
×
Hi-Tech Management System
Current password:
New password:
Retype password:
Change Password
Close

# The Employees window


Hi-Tech Management System - Employees
Hi-Tech Management System
Clear
Add
Update
Remove
Save
Close

Employee Id: 6

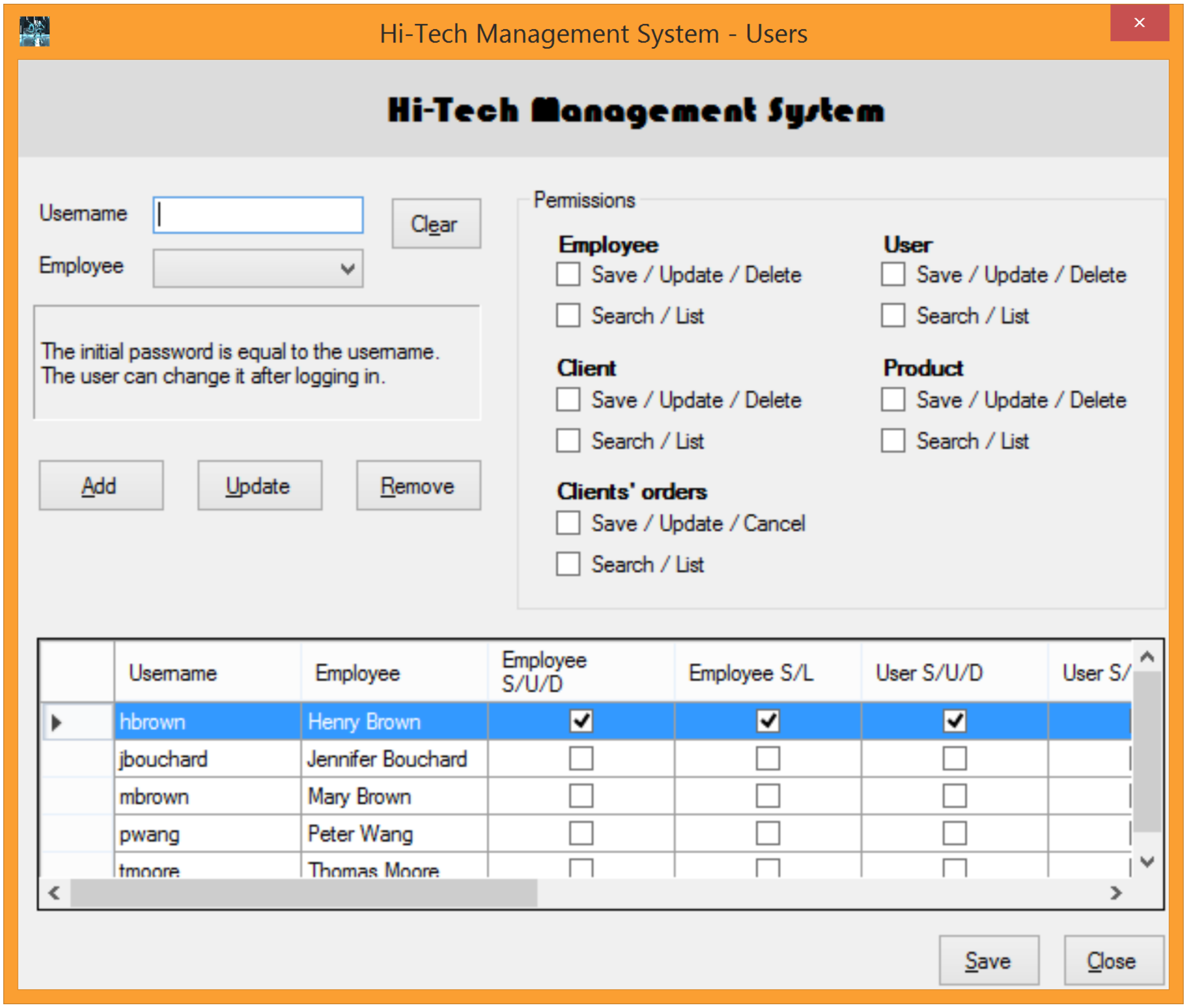
First Name:

Last Name:

Job Title:

| | Employee Id | First Name | Last Name | Job Title |
|---|---|---|---|---|
| ▶ | 1 | Henry | Brown | MIS Manager |
| | 2 | Thomas | Moore | Sales Manager |
| | 3 | Peter | Wang | Inventory Controller |
| | 4 | Mary | Brown | Order Clerk |
| | 5 | Jennifer | Bouchard | Order Clerk |

# The Users window



The initial password is equal to the username.
The user can change it after logging in.

Permissions

**Employee**

☐ Save / Update / Delete

☐ Search / List

**User**

☐ Save / Update / Delete

☐ Search / List

**Client**

☐ Save / Update / Delete

☐ Search / List

**Product**

☐ Save / Update / Delete

☐ Search / List

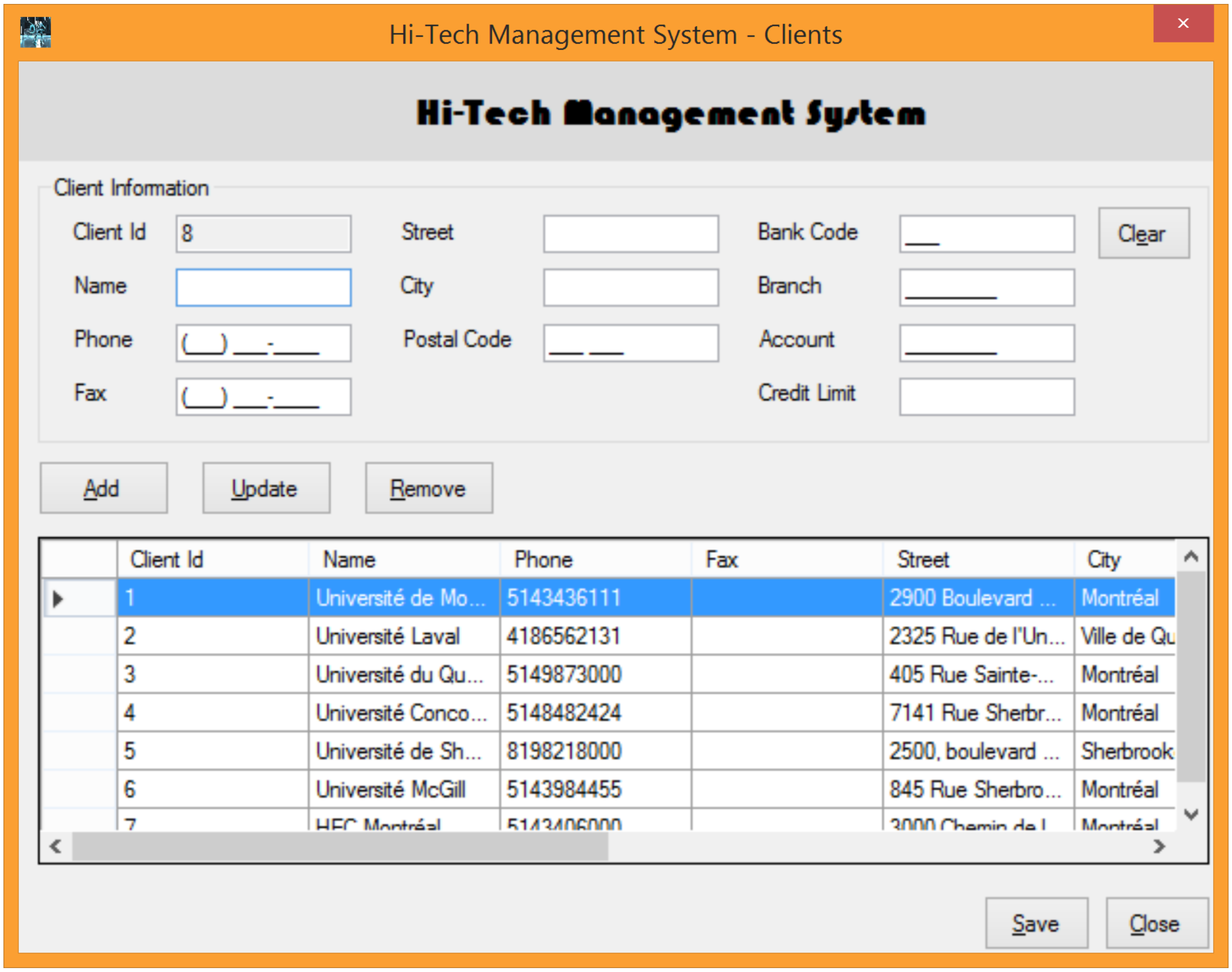
**Clients' orders**

☐ Save / Update / Cancel

☐ Search / List

| | Username | Employee | Employee S/U/D | Employee S/L | User S/U/D | User S/ |
|---|---|---|---|---|---|---|
| ▶ | hbrown | Henry Brown | ☑ | ☑ | ☑ | |
| | jbouchard | Jennifer Bouchard | ☐ | ☐ | ☐ | |
| | mbrown | Mary Brown | ☐ | ☐ | ☐ | |
| | pwang | Peter Wang | ☐ | ☐ | ☐ | |
| | tmoore | Thomas Moore | ☐ | ☐ | ☐ | |

# The Clients window


Hi-Tech Management System - Clients
Hi-Tech Management System
Client Information
Clear
Add
Update
Remove
Save
Close

| | | | | | |
|---|---|---|---|---|---|
| Client Id | 8 | Street | | Bank Code | ___ |
| Name | | City | | Branch | ________ |
| Phone | (___) ___-____ | Postal Code | ___ ___ | Account | ________ |
| Fax | (___) ___-____ | | | Credit Limit | |

| Client Id | Name | Phone | Fax | Street | City |
|---|---|---|---|---|---|
| 1 | Université de Mo... | 5143436111 | | 2900 Boulevard ... | Montréal |
| 2 | Université Laval | 4186562131 | | 2325 Rue de l'Un... | Ville de Qu |
| 3 | Université du Qu... | 5149873000 | | 405 Rue Sainte-... | Montréal |
| 4 | Université Conco... | 5148482424 | | 7141 Rue Sherbr... | Montréal |
| 5 | Université de Sh... | 8198218000 | | 2500, boulevard ... | Sherbrook |
| 6 | Université McGill | 5143984455 | | 845 Rue Sherbro... | Montréal |
| 7 | HEC Montréal | 5143406000 | | 3000 Chemin de l | Montréal |

# **Implementation**: Source Code Listings

## AuthorForm

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

namespace Solution_HitechSystem.GUI
{
    public partial class AuthorForm : Form
    {
        private ProductForm productForm;

        private Author author = new Author();
        private string authorId = default(int).ToString();
        private string firstName = string.Empty;
        private string lastName = string.Empty;


        public AuthorForm()
        {
            InitializeComponent();
        }

        public AuthorForm(ProductForm productForm)
        {
            InitializeComponent();
            this.productForm = productForm;
        }

        public AuthorForm(BookForm bookForm)
        {
            // This form can be opened via BookForm so that the user can choose
            // an author for the book she/he is adding/updating.
            //
            // If the author does not exist the user can add it in this window
            // and then select it by clicking btnSelectAuthor. This button is
            // visible only when this form is opened from BookForm.
            //
            // If the user does not want to select an author she/he can click
            // on btnClose. This way no author will be sent back to BookForm.

            InitializeComponent();
            this.btnSelectAuthor.Visible = true;
        }

        public Author SelectedAuthor()
        {
            GetSelection();
            return this.author;
        }

        private void AuthorForm_Load(object sender, EventArgs e)
        {
            ClearFields();
            LoadAuthors();

            // These settings will make this form behave like
            // a dialog window that returns an answer when a button is clicked.
            // This way the calling form will know whether the user
            // selected an author or cancelled the operation.
            btnSelectAuthor.DialogResult = DialogResult.OK;
            btnClose.DialogResult = DialogResult.Cancel;
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            if (!ValidateFields())
            {
                return;
            }

            this.author = new Author(this.authorId, this.firstName, this.lastName);
            if (AuthorsCollection.AddAuthor(this.author))
            {
                this.grdAuthors.Rows.Add(this.authorId, this.firstName, this.lastName);
                ClearFields();
            }
            else
            {
                MessageBox.Show("This Author Id already exists.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                return;
            }
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            if (!ValidateFields())
            {
                return;
            }

            // Updating the properties of the Author this.author.
            this.author.FirstName = this.firstName;
            this.author.LastName = this.lastName;

            // At this point this.author refers to an object inside AuthorsCollection.
            // So the property updates above have also updated the Author object inside
            // that collection (refer to grdAuthors_CellMouseDoubleClick).

            // Grabbing the selected row.
            int selectedRow =
this.grdAuthors.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            DataGridViewRow row = this.grdAuthors.Rows[selectedRow];

            // Updating the row data.
            row.Cells["colAuthorId"].Value = this.author.AuthorId;
            row.Cells["colFirstName"].Value = this.author.FirstName;
            row.Cells["colLastName"].Value = this.author.LastName;
            ClearFields();
        }

        private void btnRemove_Click(object sender, EventArgs e)
        {
            GetSelection();

            // TODO: check if there is a book associated with this author.
            //if (UsersCollection.GetUserByEmployee(this.employee.EmployeeId) != null)
            //{
            //    MessageBox.Show("You cannot remove this employee because there is a user 
assigned to her/him.", this.Text,
            //        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            //    return;
            //}

            // Grabbing the selected row and removing it.
            int selectedRow = 
this.grdAuthors.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            if (selectedRow >= 0)
            {
                // Removing the selected row.
                this.grdAuthors.Rows.RemoveAt(selectedRow);
                // Removing the author from AuthorsCollection.
                AuthorsCollection.RemoveAuthor(this.author);
            }

            ClearFields();
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            ClearFields();
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            HiTechDB.SaveData();
        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void grdAuthors_CellMouseDoubleClick(object sender, 
DataGridViewCellMouseEventArgs e)
        {
            // Ignoring double-clicks on the fixed cells.
            if (e.RowIndex >= 0)
            {
                GetSelection();
            }
        }

        private void ClearFields()
        {
            this.txtAuthorId.Text = AuthorsCollection.NextAuthorId();
            this.txtFirstName.Clear();
            this.txtLastName.Clear();
            if (this.Visible)
            {
                this.txtFirstName.Focus();
            }
        }

        private void GetSelection()
        {
            // Retrieves author information from the selected row in grdAuthors.
            // The information is loaded into the non-UI fields (declared at the
            // beginning of this class) and is shown in the UI fields so the
            // logged in user can update/remove it.

            int selectedRow = 
this.grdAuthors.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            if (selectedRow >= 0)
            {
                DataGridViewRow row = this.grdAuthors.Rows[selectedRow];

                // Retrieving author information from the selected row.
                this.authorId = row.Cells["colAuthorId"].Value.ToString();
                this.firstName = row.Cells["colFirstName"].Value.ToString();
                this.lastName = row.Cells["colLastName"].Value.ToString();

                // Retrieving the selected author from AuthorsCollection
                // and storing it in the Author this.author.
                this.author = AuthorsCollection.GetAuthor(this.authorId);

                // Showing the author information in the UI fields.
                this.txtAuthorId.Text = this.authorId;
                this.txtFirstName.Text = this.firstName;
                this.txtLastName.Text = this.lastName;
            }
        }

        private void LoadAuthors()
        {
            // TODO: solve the authorId sorting issue.

            GenericList<string, Author> authorList = AuthorsCollection.GetCollection();

            this.grdAuthors.Rows.Clear();

            for (int i = 0; i < authorList.GetCount(); i++)
            {
                Author author = authorList[i];

                // Creating a new row first as it will include the columns created at design-
time.
                int rowId = this.grdAuthors.Rows.Add();

                // Grabbing the new row.
                DataGridViewRow row = this.grdAuthors.Rows[rowId];

                // Adding the data.
                row.Cells["colAuthorId"].Value = author.AuthorId;
                row.Cells["colFirstName"].Value = author.FirstName;
                row.Cells["colLastName"].Value = author.LastName;
            }
        }
        private bool ValidateFields()
        {
            // Loading the provided author information into the
            // non-UI fields (declared at the beginning of this class).
            this.authorId = this.txtAuthorId.Text.Trim();
            this.firstName = this.txtFirstName.Text.Trim();
            this.lastName = this.txtLastName.Text.Trim();

            if (string.IsNullOrEmpty(this.firstName))
            {
                MessageBox.Show("Please enter the author's first name.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtFirstName.Focus();
                return false;
            }

            if (string.IsNullOrEmpty(this.lastName))
            {
                MessageBox.Show("Please enter the author's last name.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtLastName.Focus();
                return false;
            }

            return true;
        }

        private void btnSelectAuthor_Click(object sender, EventArgs e)
        {

        }
    }
}
```

---

## BookForm

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

namespace Solution_HitechSystem.GUI
{
    public partial class BookForm : Form
    {
        private ProductForm productForm;
        private Book book = new Book();
        private string bookId = default(int).ToString();
        private string title = string.Empty;
        private string isbn = string.Empty;
        private string yearPublished = string.Empty;
        private string authorId = default(int).ToString();
        private string categoryId = default(int).ToString();
        private int quantityOnHand = default(int);
        private double unitPrice = default(double);

        public BookForm()
        {
            InitializeComponent();
        }

        public BookForm(ProductForm productForm)
        {
            InitializeComponent();
            this.productForm = productForm;
        }

        private void BookForm_Load(object sender, EventArgs e)
        {
            ClearFields();
            LoadBooks();
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {

        }

        private void btnRemove_Click(object sender, EventArgs e)
        {
            GetSelection();

            //TODO: check if the book appears in any client order.

            // Grabbing the selected row and removing it.
            int selectedRow =
this.grdBooks.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            if (selectedRow >= 0)
            {
                // Removing the selected row.
                this.grdBooks.Rows.RemoveAt(selectedRow);
                // Removing the book from BooksCollection.
                BooksCollection.RemoveBook(this.book);
            }

            ClearFields();
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            ClearFields();
        }
        private void btnSave_Click(object sender, EventArgs e)
        {
            HiTechDB.SaveData();
        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void btnAuthor_Click(object sender, EventArgs e)
        {
            // This button shows btnAuthor so that the user can choose an author for the book.

            AuthorForm frm = new AuthorForm(this);
            if (frm.ShowDialog() == DialogResult.OK)
            {
                Author author = frm.SelectedAuthor();
                this.txtAuthorId.Text = author.AuthorId;
                this.txtAuthor.Text = author.FullName;
            }
        }

        private void btnCategory_Click(object sender, EventArgs e)
        {
            //CategoryForm frm = new CategoryForm(this);
            //frm.ShowDialog();
        }

        private void grdBooks_CellMouseDoubleClick(object sender,
DataGridViewCellMouseEventArgs e)
        {
            // Ignoring double-clicks on the fixed cells.
            if (e.RowIndex >= 0)
            {
                GetSelection();
            }
        }

        private void ClearFields()
        {
            // Clearing the textboxes.
            foreach (Control ctrl in grpBookInformation.Controls)
            {
                if (ctrl is TextBox)
                {
                    (ctrl as TextBox).Clear();
                }
            }

            this.txtBookId.Text = BooksCollection.NextBookId();

            if (this.Visible)
            {
                this.txtTitle.Focus();
            }
        }

        private void GetSelection()
        {
            // Retrieves book information from the selected row in grdBooks.
            // The information is loaded into the non-UI fields (declared at the
            // beginning of this class) and is shown in the UI fields so the
            // logged in user can update/remove it.

            int selectedRow =
this.grdBooks.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            if (selectedRow >= 0)
            {
                DataGridViewRow row = this.grdBooks.Rows[selectedRow];

                // Retrieving book information from the selected row.
                this.bookId = row.Cells["colBookId"].Value.ToString();
                this.title = row.Cells["colTitle"].Value.ToString();
                this.isbn = row.Cells["colISBN"].Value.ToString();
                this.yearPublished = row.Cells["colYearPublished"].Value.ToString();
                this.authorId = row.Cells["colAuthorId"].Value.ToString();
                string authorFullName = row.Cells["colAuthorFullName"].Value.ToString();
                this.categoryId = row.Cells["colCategoryId"].Value.ToString();
                string categoryDescription =
row.Cells["colCategoryDescription"].Value.ToString();
                this.quantityOnHand = 0;
                bool parseOKQuantityOnHand =
int.TryParse(row.Cells["colQuantityOnHand"].Value.ToString(), out this.quantityOnHand);
                this.unitPrice = 0;
                bool parseOKUnitPrice =
double.TryParse(row.Cells["colUnitPrice"].Value.ToString(), out this.unitPrice);

                // Retrieving the selected book from BooksCollection
                // and storing it in the Book this.book.
                this.book = BooksCollection.GetBook(this.bookId);

                // Showing the book information in the UI fields.
                this.txtBookId.Text = this.bookId;
                this.txtTitle.Text = this.title;
                this.txtISBN.Text = this.isbn;
                this.txtYearPublished.Text = this.yearPublished;
                this.txtAuthorId.Text = this.authorId;
                this.txtAuthor.Text = authorFullName;
                this.txtCategoryId.Text = this.categoryId;
                this.txtCategory.Text = categoryDescription;
                this.txtQuantityOnHand.Text = this.quantityOnHand.ToString();
                this.txtUnitPrice.Text = this.unitPrice.ToString();
            }
        }

        private void LoadBooks()
        {
            GenericList<string, Book> bookList = BooksCollection.GetCollection();

            this.grdBooks.Rows.Clear();

            for (int i = 0; i < bookList.GetCount(); i++)
            {
                Book book = bookList[i];
                Author author = AuthorsCollection.GetAuthor(book.AuthorId);
                Category category = CategoriesCollection.GetCategory(book.CategoryId);

                // Creating a new row first as it will include the columns created at design-
time.
                int rowId = this.grdBooks.Rows.Add();
                // Grabbing the new row.
                DataGridViewRow row = this.grdBooks.Rows[rowId];

                // Adding the data.
                row.Cells["colBookId"].Value = book.ProductId;
                row.Cells["colTitle"].Value = book.ProductName;
                row.Cells["colISBN"].Value = book.ISBN;
                row.Cells["colYearPublished"].Value = book.YearPublished;
                row.Cells["colAuthorId"].Value = book.AuthorId;
                row.Cells["colAuthorFullName"].Value = author.FullName;
                row.Cells["colCategoryId"].Value = book.CategoryId;
                row.Cells["colCategoryDescription"].Value = category.Description;
                row.Cells["colQuantityOnHand"].Value = book.QuantityOnHand;
                row.Cells["colUnitPrice"].Value = book.UnitPrice;
            }
        }

        private bool ValidateFields()
        {
            // Loading the provided book information into the
            // non-UI fields (declared at the beginning of this class).
            this.bookId = this.txtBookId.Text.Trim();
            this.title = this.txtTitle.Text.Trim();
            this.isbn = this.txtISBN.Text.Trim();
            this.yearPublished = this.txtYearPublished.Text.Trim();
            this.authorId = this.txtAuthorId.Text.Trim();
            this.categoryId = this.txtCategoryId.Text.Trim();
            this.quantityOnHand = 0;
            bool parseOKQuantityOnHand = int.TryParse(this.txtQuantityOnHand.Text.Trim(), out
this.quantityOnHand);
            this.unitPrice = 0;
            bool parseOKUnitPrice = double.TryParse(this.txtUnitPrice.Text.Trim(), out
this.unitPrice);

            if (string.IsNullOrEmpty(this.title))
            {
                MessageBox.Show("Please enter the book's title.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtTitle.Focus();
                return false;
            }

            //TODO: validate ISBN format.
            if (string.IsNullOrEmpty(this.isbn))
            {
                MessageBox.Show("Please enter the book's ISBN.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtISBN.Focus();
                return false;
            }

            if (string.IsNullOrEmpty(this.yearPublished))
            {
                MessageBox.Show("Please enter the year the book was published.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtYearPublished.Focus();
                return false;
            }
            else
            {
                // Converting this.yearPublished to int to check whether it is within an
acceptable range.
                int year = 0;
                bool parseOKYearPublished = int.TryParse(this.yearPublished, out year);

                if (!parseOKYearPublished || (year < 1940) || (year > DateTime.Now.Year))
                {
                    MessageBox.Show("The year you entered seems incorrect. Please try again.",
this.Text,
                        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    this.txtYearPublished.Focus();
                    return false;
                }
            }

            if (string.IsNullOrEmpty(this.authorId))
            {
                MessageBox.Show("Please enter the book's author.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtAuthor.Focus();
                return false;
            }

            if (string.IsNullOrEmpty(this.categoryId))
            {
                MessageBox.Show("Please select a category for this book.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtCategory.Focus();
                return false;
            }

            if (!parseOKQuantityOnHand)
            {
                // Just checking whether the value entered is a valid number or not.
                // Not checking if it is greater than zero because it might be out of stock.

                MessageBox.Show("Please enter a valid quantity on hand. Leave this field blank
if the book is out of stock.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtQuantityOnHand.Focus();
                return false;
            }

            if (!parseOKUnitPrice || (this.unitPrice == 0))
            {
                MessageBox.Show("Please enter a valid unit price for this book.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtUnitPrice.Focus();
                return false;
            }

            return true;
        }
    }
}
```

## CategoryForm

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

namespace Solution_HitechSystem.GUI
{
    public partial class CategoryForm : Form
    {
        private ProductForm productForm;

        private Category category = new Category();
        private string categoryId = default(int).ToString();
        private string description = string.Empty;

        public CategoryForm()
        {
            InitializeComponent();
        }

        public CategoryForm(ProductForm productForm)
        {
            InitializeComponent();
            this.productForm = productForm;
        }

        private void CategoryForm_Load(object sender, EventArgs e)
        {
            ClearFields();
            LoadCategories();
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            if (!ValidateFields())
            {
                return;
            }

            this.category = new Category(this.categoryId, this.description);
            if (CategoriesCollection.AddCategory(this.category))
            {
                this.grdCategories.Rows.Add(this.categoryId, this.description);
                ClearFields();
            }
            else
            {
                MessageBox.Show("This Category Id already exists.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                return;
            }
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            if (!ValidateFields())
            {
                return;
            }

            // Updating the properties of the Category this.category.
            this.category.Description = this.description;

            // At this point this.category refers to an object inside CategoriesCollection.
            // So the property updates above have also updated the Category object inside
            // that collection (refer to grdCategories_CellMouseDoubleClick).

            // Grabbing the selected row.
            int selectedRow = 
this.grdCategories.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            DataGridViewRow row = this.grdCategories.Rows[selectedRow];

            // Updating the row data.
            row.Cells["colCategoryId"].Value = this.category.CategoryId;
            row.Cells["colDescription"].Value = this.category.Description;

            ClearFields();
        }

        private void btnRemove_Click(object sender, EventArgs e)
        {
            GetSelection();

            // TODO: check if there is a product (book/software) associated with this 
category.
            //if (UsersCollection.GetUserByEmployee(this.employee.EmployeeId) != null)
            //{
            //    MessageBox.Show("You cannot remove this employee because there is a user 
assigned to her/him.", this.Text,
            //        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            //    return;
            //}

            // Grabbing the selected row and removing it.
            int selectedRow = 
this.grdCategories.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            if (selectedRow >= 0)
            {
                // Removing the selected row.
                this.grdCategories.Rows.RemoveAt(selectedRow);
                // Removing the category from CategoriesCollection.
                CategoriesCollection.RemoveCategory(this.category);
            }

            ClearFields();
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            ClearFields();
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            HiTechDB.SaveData();
        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void grdCategories_CellMouseDoubleClick(object sender,
DataGridViewCellMouseEventArgs e)
        {
            // Ignoring double-clicks on the fixed cells.
            if (e.RowIndex >= 0)
            {
                GetSelection();
            }
        }

        private void ClearFields()
        {
            this.txtCategoryId.Text = CategoriesCollection.NextCategoryId();
            this.txtDescription.Clear();

            if (this.Visible)
            {
                this.txtDescription.Focus();
            }
        }

        private void GetSelection()
        {
            // Retrieves category information from the selected row in grdCategories.
            // The information is loaded into the non-UI fields (declared at the
            // beginning of this class) and is shown in the UI fields so the
            // logged in user can update/remove it.

            int selectedRow =
this.grdCategories.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            if (selectedRow >= 0)
            {
                DataGridViewRow row = this.grdCategories.Rows[selectedRow];

                // Retrieving category information from the selected row.
                this.categoryId = row.Cells["colCategoryId"].Value.ToString();
                this.description = row.Cells["colDescription"].Value.ToString();

                // Retrieving the selected category from CategoriesCollection
                // and storing it in the Category this.category.
                this.category = CategoriesCollection.GetCategory(this.categoryId);

                // Showing the category information in the UI fields.
                this.txtCategoryId.Text = this.categoryId;
                this.txtDescription.Text = this.description;
            }
        }

        private void LoadCategories()
        {
            // TODO: solve the categoryId sorting issue.

            GenericList<string, Category> categoryList = CategoriesCollection.GetCollection();

            this.grdCategories.Rows.Clear();

            for (int i = 0; i < categoryList.GetCount(); i++)
            {
                Category category = categoryList[i];

                // Creating a new row first as it will include the columns created at design-
time.
                int rowId = this.grdCategories.Rows.Add();

                // Grabbing the new row.
                DataGridViewRow row = this.grdCategories.Rows[rowId];

                // Adding the data.
                row.Cells["colCategoryId"].Value = category.CategoryId;
                row.Cells["colDescription"].Value = category.Description;
            }
        }

        private bool ValidateFields()
        {
            // Loading the provided category information into the
            // non-UI fields (declared at the beginning of this class).
            this.categoryId = this.txtCategoryId.Text.Trim();
            this.description = this.txtDescription.Text.Trim();

            if (string.IsNullOrEmpty(this.description))
            {
                MessageBox.Show("Please enter the category's description.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtDescription.Focus();
                return false;
            }

            return true;
        }
    }
}
```

## ChangePasswordForm

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

namespace Solution_HitechSystem.GUI
{
    public partial class ChangePasswordForm : Form
    {
        private MainForm mainForm;

        public ChangePasswordForm()
        {
            InitializeComponent();
        }

        public ChangePasswordForm(MainForm mainForm)
        {
            InitializeComponent();
            this.mainForm = mainForm;
        }

        private void btnChangePassword_Click(object sender, EventArgs e)
        {
            if (!ValidateFields())
            {
                return;
            }

            this.mainForm.UserLoggedIn.Password = this.txtNewPassword.Text;

            this.Close();
        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private bool ValidateFields()
        {
            string currentPassword = this.txtCurrentPassword.Text;
            string newPassword = this.txtNewPassword.Text;
            string retypedPassword = this.txtRetypePassword.Text;

            if (string.IsNullOrEmpty(currentPassword))
            {
                MessageBox.Show("Please enter your current password.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtCurrentPassword.Focus();
                return false;
            }
            // Checking whether the password entered in
            // this.txtCurrentPassword is indeed this user's password.
            if (!this.mainForm.UserLoggedIn.CheckPassword(currentPassword))
            {
                MessageBox.Show("This is not your current password. Please try again.",
this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtCurrentPassword.Focus();
                return false;
            }

            if (string.IsNullOrEmpty(newPassword))
            {
                MessageBox.Show("Please enter your new password.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtNewPassword.Focus();
                return false;
            }

            // The new password must be different from the current one.
            if (this.mainForm.UserLoggedIn.CheckPassword(newPassword))
            {
                MessageBox.Show("The new password must be different from the current one.
Please try again.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtNewPassword.Focus();
                return false;
            }

            if (string.IsNullOrEmpty(retypedPassword))
            {
                MessageBox.Show("Please retype your new password.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtRetypePassword.Focus();
                return false;
            }

            if (retypedPassword != newPassword)
            {
                MessageBox.Show("The new and the retyped passwords must match. Please try
again.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtRetypePassword.Focus();
                return false;
            }

            return true;
        }
    }
}
```

## ClientForm

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

namespace Solution_HitechSystem.GUI
{
    public partial class ClientForm : Form
    {
        private MainForm mainForm;

        private Client client = new Client();
        private string clientId     = default(int).ToString();
        private string clientName   = string.Empty;
        private string address      = string.Empty;
        private string city         = string.Empty;
        private string postalCode   = string.Empty;
        private string phoneNumber  = string.Empty;
        private string faxNumber    = string.Empty;
        private string bankNumber   = string.Empty;
        private string branchNumber = string.Empty;
        private string bankAccount  = string.Empty;
        private double creditLimit  = default(double);

        public ClientForm()
        {
            InitializeComponent();
        }

        public ClientForm(MainForm mainForm)
        {
            InitializeComponent();
            this.mainForm = mainForm;
        }

        private void ClientForm_Load(object sender, EventArgs e)
        {
            ClearFields();
            LoadClients();
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            if (!ValidateFields())
            {
                return;
            }

            this.client = new Client(this.clientId, this.clientName,
                this.address, this.city, this.postalCode, this.phoneNumber, this.faxNumber,
                this.creditLimit, this.bankNumber, this.branchNumber, this.bankAccount);

            if (ClientsCollection.AddClient(this.client))
            {
                this.grdClients.Rows.Add(this.clientId, this.clientName, this.phoneNumber,
this.faxNumber,
                    this.address, this.city, this.postalCode,
                    this.bankNumber, this.branchNumber, this.bankAccount, this.creditLimit);
                ClearFields();
            }
            else
            {
                MessageBox.Show("This Client Id already exists.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                return;
            }
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            if (!ValidateFields())
            {
                return;
            }

            // Updating the properties of the Client this.client.
            this.client.ClientId = this.clientId;
            this.client.Name = this.clientName;
            this.client.PhoneNumber = this.phoneNumber;
            this.client.FaxNumber = this.faxNumber;
            this.client.Address = this.address;
            this.client.City = this.city;
            this.client.PostalCode = this.postalCode;
            this.client.BankNumber = this.bankNumber;
            this.client.BranchNumber = this.branchNumber;
            this.client.BankAccount = this.bankAccount;
            this.client.CreditLimit = this.creditLimit;

            // At this point this.client refers to an object inside ClientsCollection.
            // So the property updates above have also updated the Client object inside
            // that collection (refer to grdClients_CellMouseDoubleClick).

            // Grabbing the selected row.
            int selectedRow =
this.grdClients.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            DataGridViewRow row = this.grdClients.Rows[selectedRow];

            // Updating the row data.
            row.Cells["colClientId"].Value = this.client.ClientId;
            row.Cells["colClientName"].Value = this.client.Name;
            row.Cells["colPhoneNumber"].Value = this.client.PhoneNumber;
            row.Cells["colFaxNumber"].Value = this.client.FaxNumber;
            row.Cells["colAddress"].Value = this.client.Address;
            row.Cells["colCity"].Value = this.client.City;
            row.Cells["colPostalCode"].Value = this.client.PostalCode;
            row.Cells["colBankNumber"].Value = this.client.BankNumber;
            row.Cells["colBranchNumber"].Value = this.client.BranchNumber;
            row.Cells["colBankAccount"].Value = this.client.BankAccount;
            row.Cells["colCreditLimit"].Value = this.client.CreditLimit;
            ClearFields();
        }

        private void btnRemove_Click(object sender, EventArgs e)
        {
            GetSelection();

            //TODO: check if the client has already placed any order.

            // Grabbing the selected row and removing it.
            int selectedRow = 
this.grdClients.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            if (selectedRow >= 0)
            {
                // Removing the selected row.
                this.grdClients.Rows.RemoveAt(selectedRow);
                // Removing the client from ClientsCollection.
                ClientsCollection.RemoveClient(this.client);
            }

            ClearFields();
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            ClearFields();
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            HiTechDB.SaveData();
        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void grdClients_CellMouseDoubleClick(object sender, 
DataGridViewCellMouseEventArgs e)
        {
            // Ignoring double-clicks on the fixed cells.
            if (e.RowIndex >= 0)
            {
                GetSelection();
            }
        }

        private void ClearFields()
        {
            // Clearing the textboxes and maskedtextboxes.
            foreach (Control ctrl in grpClientInformation.Controls)
            {
                if (ctrl is TextBox)
                {
                    (ctrl as TextBox).Clear();
                }
                if (ctrl is MaskedTextBox)
                {
                    (ctrl as MaskedTextBox).Clear();
                }
            }

            this.txtClientId.Text = ClientsCollection.NextClientId();

            if (this.Visible)
            {
                this.txtClientName.Focus();
            }
        }

        private void GetSelection()
        {
            // Retrieves client information from the selected row in grdClients.
            // The information is loaded into the non-UI fields (declared at the
            // beginning of this class) and is shown in the UI fields so the
            // logged in user can update/remove it.

            int selectedRow =
this.grdClients.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            if (selectedRow >= 0)
            {
                DataGridViewRow row = this.grdClients.Rows[selectedRow];

                // Retrieving client information from the selected row.
                this.clientId = row.Cells["colClientId"].Value.ToString();
                this.clientName = row.Cells["colClientName"].Value.ToString();
                this.phoneNumber = row.Cells["colPhoneNumber"].Value.ToString();
                this.faxNumber = row.Cells["colFaxNumber"].Value.ToString();
                this.address = row.Cells["colAddress"].Value.ToString();
                this.city = row.Cells["colCity"].Value.ToString();
                this.postalCode = row.Cells["colPostalCode"].Value.ToString();
                this.bankNumber = row.Cells["colBankNumber"].Value.ToString();
                this.branchNumber = row.Cells["colBranchNumber"].Value.ToString();
                this.bankAccount = row.Cells["colBankAccount"].Value.ToString();
                this.creditLimit = 0;
                bool parseOKCreditLimit =
double.TryParse(row.Cells["colCreditLimit"].Value.ToString(), out this.creditLimit);

                // Retrieving the selected client from ClientsCollection
                // and storing it in the Client this.client.
                this.client = ClientsCollection.GetClient(this.clientId);

                // Showing the client information in the UI fields.
                this.txtClientId.Text = this.clientId;
                this.txtClientName.Text = this.clientName;
                this.mskPhoneNumber.Text = this.phoneNumber;
                this.mskFaxNumber.Text = this.faxNumber;
                this.txtAddress.Text = this.address;
                this.txtCity.Text = this.city;
                this.mskPostalCode.Text = this.postalCode;
                this.mskBankNumber.Text = this.bankNumber;
                this.mskBranchNumber.Text = this.branchNumber;
                this.mskBankAccount.Text = this.bankAccount;
                this.txtCreditLimit.Text = this.creditLimit.ToString();
            }
        }

        private void LoadClients()
        {
            GenericList<string, Client> clientList = ClientsCollection.GetCollection();
            this.grdClients.Rows.Clear();

            for (int i = 0; i < clientList.GetCount(); i++)
            {
                Client client = clientList[i];

                // Creating a new row first as it will include the columns created at design-
time.
                int rowId = this.grdClients.Rows.Add();

                // Grabbing the new row.
                DataGridViewRow row = this.grdClients.Rows[rowId];

                // Adding the data.
                row.Cells["colClientId"].Value = client.ClientId;
                row.Cells["colClientName"].Value = client.Name;
                row.Cells["colPhoneNumber"].Value = client.PhoneNumber;
                row.Cells["colFaxNumber"].Value = client.FaxNumber;
                row.Cells["colAddress"].Value = client.Address;
                row.Cells["colCity"].Value = client.City;
                row.Cells["colPostalCode"].Value = client.PostalCode;
                row.Cells["colBankNumber"].Value = client.BankNumber;
                row.Cells["colBranchNumber"].Value = client.BranchNumber;
                row.Cells["colBankAccount"].Value = client.BankAccount;
                row.Cells["colCreditLimit"].Value = client.CreditLimit;
            }
        }

        private bool ValidateFields()
        {
            // Loading the provided client information into the
            // non-UI fields (declared at the beginning of this class).
            this.clientId = this.txtClientId.Text.Trim();
            this.clientName = this.txtClientName.Text.Trim();
            this.phoneNumber = this.mskPhoneNumber.Text.Trim();
            this.faxNumber = this.mskFaxNumber.Text.Trim();
            this.address = this.txtAddress.Text.Trim();
            this.city = this.txtCity.Text.Trim();
            this.postalCode = this.mskPostalCode.Text.Trim();
            this.bankNumber = this.mskBankNumber.Text.Trim();
            this.branchNumber = this.mskBranchNumber.Text.Trim();
            this.bankAccount = this.mskBankAccount.Text.Trim();
            this.creditLimit = 0;
            bool parseOKCreditLimit = double.TryParse(this.txtCreditLimit.Text.Trim(), out
this.creditLimit);

            if (string.IsNullOrEmpty(this.clientName))
            {
                MessageBox.Show("Please enter the client's name.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtClientName.Focus();
                return false;
            }

            if (string.IsNullOrEmpty(this.phoneNumber))
            {
                MessageBox.Show("Please enter the client's phone number (format: XXX XXX
XXXX).", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.mskPhoneNumber.Focus();
                return false;
            }
            else
            {
                // this.phoneNumber has only the digits.
                Match m = Regex.Match(this.phoneNumber, @"^\d{10}$");
                if (!m.Success)
                {
                    MessageBox.Show("The client's phone number is in an incorrect format.
Please use: XXX XXX XXXX", this.Text,
                        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    this.mskPhoneNumber.Focus();
                    return false;
                }
            }

            // Fax number can be empty, but if provided must follow the correct phone format.
            if (!string.IsNullOrEmpty(this.faxNumber))
            {
                // this.faxNumber has only the digits.
                Match m = Regex.Match(this.faxNumber, @"^\d{10}$");
                if (!m.Success)
                {
                    MessageBox.Show("The client's fax number is in an incorrect format. Please
use: XXX XXX XXXX", this.Text,
                        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    this.mskFaxNumber.Focus();
                    return false;
                }
            }

            if (string.IsNullOrEmpty(this.address))
            {
                MessageBox.Show("Please enter the client's address.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtAddress.Focus();
                return false;
            }

            if (string.IsNullOrEmpty(this.city))
            {
                MessageBox.Show("Please enter the client's city.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtCity.Focus();
                return false;
            }

            if (string.IsNullOrEmpty(this.postalCode))
            {
                MessageBox.Show("Please enter the client's postal code (format A1A 1A1, where
A is a letter and 1 is a digit).", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.mskPostalCode.Focus();
                return false;
            }
            else
            {
                // Canadian postal code regular expression validation taken from:
                // http://stackoverflow.com/questions/1146202/canadian-postal-code-validation
                Match m = Regex.Match(this.postalCode, @"^[ABCEGHJKLMNPRSTVXY][0-
9][ABCEGHJKLMNPRSTVWXYZ][0-9][ABCEGHJKLMNPRSTVWXYZ][0-9]$");
                if (!m.Success)
                {
                    MessageBox.Show("This is not a valid postal code. Please try again.",
this.Text,
                        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    this.mskPostalCode.Focus();
                    return false;
                }
            }

            if (string.IsNullOrEmpty(this.bankNumber))
            {
                MessageBox.Show("Please enter the client's bank number.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.mskBankNumber.Focus();
                return false;
            }
            else
            {
                Match m = Regex.Match(this.faxNumber, @"^\d{3}$");
                if (!m.Success)
                {
                    MessageBox.Show("The bank number should be composed of 3 digits. Please
try again.", this.Text,
                        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    this.mskBankNumber.Focus();
                    return false;
                }
            }

            if (string.IsNullOrEmpty(this.branchNumber))
            {
                MessageBox.Show("Please enter the client's branch (transit) number.",
this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.mskBranchNumber.Focus();
                return false;
            }
            {
                Match m = Regex.Match(this.faxNumber, @"^\d{5}$");
                if (!m.Success)
                {
                    MessageBox.Show("The branch number should be composed of 5 digits. Please
try again.", this.Text,
                        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    this.mskBranchNumber.Focus();
                    return false;
                }
            }

            if (string.IsNullOrEmpty(this.bankAccount))
            {
                MessageBox.Show("Please enter the client's account number.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.mskBankAccount.Focus();
                return false;
            }

            if (!parseOKCreditLimit || (this.creditLimit == 0))
            {
                MessageBox.Show("Please enter a valid non-zero credit limit for this client.",
this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtCreditLimit.Focus();
                return false;
            }

            return true;
        }
    }
}
```

## EmployeeForm

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

namespace Solution_HitechSystem.GUI
{
    public partial class EmployeeForm : Form
    {
        private MainForm mainForm;

        private Employee employee = new Employee();
        private string employeeId = default(int).ToString();
        private string firstName = string.Empty;
        private string lastName = string.Empty;
        private string jobTitle = string.Empty;

        public EmployeeForm()
        {
            InitializeComponent();
        }

        public EmployeeForm(MainForm mainForm)
        {
            InitializeComponent();
            this.mainForm = mainForm;
        }

        private void EmployeeForm_Load(object sender, EventArgs e)
        {
            ClearFields();
            LoadEmployees();
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            if (!ValidateFields())
            {
                return;
            }

            this.employee = new Employee(this.employeeId, this.firstName, this.lastName,
this.jobTitle);
            if (EmployeesCollection.AddEmployee(this.employee))
            {
                this.grdEmployees.Rows.Add(this.employeeId, this.firstName, this.lastName,
this.jobTitle);
                ClearFields();
            }
            else
            {
                MessageBox.Show("This Employee Id already exists.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                return;
            }
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            if (!ValidateFields())
            {
                return;
            }

            // Updating the properties of the Employee this.employee.
            this.employee.FirstName = this.firstName;
            this.employee.LastName = this.lastName;
            this.employee.JobTitle = this.jobTitle;

            // At this point this.employee refers to an object inside EmployeesCollection.
            // So the property updates above have also updated the Employee object inside
            // that collection (refer to grdEmployees_CellMouseDoubleClick).

            // Grabbing the selected row.
            int selectedRow =
this.grdEmployees.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            DataGridViewRow row = this.grdEmployees.Rows[selectedRow];

            // Updating the row data.
            row.Cells["colEmployeeId"].Value = this.employee.EmployeeId;
            row.Cells["colFirstName"].Value = this.employee.FirstName;
            row.Cells["colLastName"].Value = this.employee.LastName;
            row.Cells["colJobTitle"].Value = this.employee.JobTitle;

            ClearFields();
        }

        private void btnRemove_Click(object sender, EventArgs e)
        {
            GetSelection();

            // Checking if the employee being removed has a user in the system.
            if (UsersCollection.GetUserByEmployee(this.employee.EmployeeId) != null)
            {
                MessageBox.Show("You cannot remove this employee because there is a user
assigned to her/him.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                return;
            }

            // Grabbing the selected row and removing it.
            int selectedRow = 
this.grdEmployees.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            if (selectedRow >= 0)
            {
                // Removing the selected row.
                this.grdEmployees.Rows.RemoveAt(selectedRow);
                // Removing the employee from EmployeesCollection.
                EmployeesCollection.RemoveEmployee(this.employee);
            }

            ClearFields();
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            HiTechDB.SaveData();
        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void LoadEmployees()
        {
            GenericList<string, Employee> employeeList = EmployeesCollection.GetCollection();

            this.grdEmployees.Rows.Clear();

            for (int i = 0; i < employeeList.GetCount(); i++)
            {
                Employee employee = employeeList[i];

                // Creating a new row first as it will include the columns created at design-
time.
                int rowId = this.grdEmployees.Rows.Add();

                // Grabbing the new row.
                DataGridViewRow row = this.grdEmployees.Rows[rowId];

                // Adding the data.
                row.Cells["colEmployeeId"].Value = employee.EmployeeId;
                row.Cells["colFirstName"].Value = employee.FirstName;
                row.Cells["colLastName"].Value = employee.LastName;
                row.Cells["colJobTitle"].Value = employee.JobTitle;
            }
        }

        private void ClearFields()
        {
            this.txtEmployeeId.Text = EmployeesCollection.NextEmployeeId();
            this.txtFirstName.Clear();
            this.txtLastName.Clear();
            this.txtJobTitle.Clear();

            if (this.Visible)
            {
                this.txtFirstName.Focus();
            }
        }

        private bool ValidateFields()
        {
            // Loading the provided employee information into the
            // non-UI fields (declared at the beginning of this class).
            this.employeeId = this.txtEmployeeId.Text.Trim();
            this.firstName = this.txtFirstName.Text.Trim();
            this.lastName = this.txtLastName.Text.Trim();
            this.jobTitle = this.txtJobTitle.Text.Trim();

            if (string.IsNullOrEmpty(this.firstName))
            {
                MessageBox.Show("Please enter the employee's first name.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtFirstName.Focus();
                return false;
            }

            if (string.IsNullOrEmpty(this.lastName))
            {
                MessageBox.Show("Please enter the employee's last name.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtLastName.Focus();
                return false;
            }

            if (string.IsNullOrEmpty(this.jobTitle))
            {
                MessageBox.Show("Please enter the employee's job title.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtJobTitle.Focus();
                return false;
            }

            return true;
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            ClearFields();
        }

        private void GetSelection()
        {
            // Retrieves employee information from the selected row in grdEmployees.
            // The information is loaded into the non-UI fields (declared at the
            // beginning of this class) and is shown in the UI fields so the
            // logged in user can update/remove it.

            int selectedRow = 
this.grdEmployees.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            if (selectedRow >= 0)
            {
                DataGridViewRow row = this.grdEmployees.Rows[selectedRow];

                // Retrieving employee information from the selected row.
                this.employeeId = row.Cells["colEmployeeId"].Value.ToString();
                this.firstName = row.Cells["colFirstName"].Value.ToString();
                this.lastName = row.Cells["colLastName"].Value.ToString();
                this.jobTitle = row.Cells["colJobTitle"].Value.ToString();

                // Retrieving the selected employee from EmployeesCollection
                // and storing it in the Employee this.employee.
                this.employee = EmployeesCollection.GetEmployee(this.employeeId);

                // Showing the employee information in the UI fields.
                this.txtEmployeeId.Text = this.employeeId;
                this.txtFirstName.Text = this.firstName;
                this.txtLastName.Text = this.lastName;
                this.txtJobTitle.Text = this.jobTitle;
            }
        }

        private void grdEmployees_CellMouseDoubleClick(object sender,
DataGridViewCellMouseEventArgs e)
        {
            // Ignoring double-clicks on the fixed cells.
            if (e.RowIndex >= 0)
            {
                GetSelection();
            }
        }
    }
}
```

## LoginForm

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

namespace Solution_HitechSystem.GUI
{
    public partial class LoginForm : Form
    {
        // This variable holds the user who logged in to the system.
        private User user = null;

        public User UserLoggedIn
        {
            get { return user; }
        }

        public LoginForm()
        {
            InitializeComponent();
        }
        private void btnExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void btnLogin_Click(object sender, EventArgs e)
        {
            string username = this.txtUsername.Text.Trim();
            string password = this.txtPassword.Text.Trim();

            if (username.Length == 0)
            {
                MessageBox.Show("Please enter your username.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtUsername.Focus();
                return;
            }

            if (password.Length == 0)
            {
                MessageBox.Show("Please enter your password.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtPassword.Focus();
                return;
            }

            // Retrieving the user by the username.
            this.user = UsersCollection.GetUser(username);

            if (this.user == null)
            {
                MessageBox.Show($"Username '{username}' does not exist.", this.Text,
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtUsername.Focus();
                return;
            }

            if (this.user.CheckPassword(password))
            {
                this.Hide();
                return;
            }
            else
            {
                MessageBox.Show("You entered a wrong password. Try again.", this.Text,
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtPassword.Focus();
                return;
            }
        }
    }
}
```

## MainForm

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

namespace Solution_HitechSystem.GUI
{
    public partial class MainForm : Form
    {
        // User logged into the system.
        private User userLoggedIn;

        public MainForm()
        {
            InitializeComponent();
        }

        public MainForm(User user)
        {
            InitializeComponent();

            this.userLoggedIn = user;
            this.lblStatusBar.Text = $"User: {this.userLoggedIn.Username}";
            // Enablind/disabling the buttons according to user's permissions.
            CheckPermissions();
        }

        public User UserLoggedIn
        {
            get { return userLoggedIn; }
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void CheckPermissions()
        {
            // Retrieving the user's permissions.
            Permission permissions = 
PermissionsCollection.GetPermission(this.userLoggedIn.Username);

            this.btnEmployees.Enabled = (permissions.EmployeeSaveUpdateDelete || 
permissions.EmployeeSearchList);
            this.btnUsers.Enabled = (permissions.UserSaveUpdateDelete || 
permissions.UserSearchList);
            this.btnClients.Enabled = (permissions.ClientSaveUpdateDelete || 
permissions.ClientSearchList);
```

```
            this.btnOrders.Enabled = (permissions.ClientOrderSaveUpdateCancel ||
permissions.ClientOrderSearchList);
            this.btnProducts.Enabled = (permissions.ProductSaveUpdateDelete ||
permissions.ProductSearchList);
        }

        private void btnEmployees_Click(object sender, EventArgs e)
        {
            EmployeeForm frm = new EmployeeForm(this);
            frm.ShowDialog();
        }

        private void btnUsers_Click(object sender, EventArgs e)
        {
            UserForm frm = new UserForm(this);
            frm.ShowDialog();
        }

        private void btnChangePassword_Click(object sender, EventArgs e)
        {
            ChangePasswordForm frm = new ChangePasswordForm(this);
            frm.ShowDialog();
        }

        private void btnClients_Click(object sender, EventArgs e)
        {
            ClientForm frm = new ClientForm(this);
            frm.ShowDialog();
        }

        private void btnProducts_Click(object sender, EventArgs e)
        {
            ProductForm frm = new ProductForm(this);
            frm.ShowDialog();
        }
    }
}
```

## ProductForm

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

namespace Solution_HitechSystem.GUI
{
    public partial class ProductForm : Form
    {
        private MainForm mainForm;

        public ProductForm()
        {
            InitializeComponent();
        }

        public ProductForm(MainForm mainForm)
        {
            InitializeComponent();
            this.mainForm = mainForm;
        }

        private void ProductForm_Load(object sender, EventArgs e)
        {

        }

        private void btnAuthors_Click(object sender, EventArgs e)
        {
            AuthorForm frm = new AuthorForm(this);
            frm.ShowDialog();
        }

        private void btnCategories_Click(object sender, EventArgs e)
        {
            CategoryForm frm = new CategoryForm(this);
            frm.ShowDialog();
        }

        private void btnSuppliers_Click(object sender, EventArgs e)
        {
            SupplierForm frm = new SupplierForm(this);
            frm.ShowDialog();
        }

        private void btnBooks_Click(object sender, EventArgs e)
        {
            BookForm frm = new BookForm(this);
            frm.ShowDialog();
        }
    }
}
```

## SupplierForm

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

namespace Solution_HitechSystem.GUI
{
    public partial class SupplierForm : Form
    {
        private ProductForm productForm;

        private Supplier supplier = new Supplier();
        private string supplierId = default(int).ToString();
        private string supplierName = string.Empty;
        private string address = string.Empty;
        private string city = string.Empty;
        private string postalCode = string.Empty;
        private string phoneNumber = string.Empty;

        public SupplierForm()
        {
            InitializeComponent();
        }

        public SupplierForm(ProductForm productForm)
        {
            InitializeComponent();
            this.productForm = productForm;
        }

        private void SupplierForm_Load(object sender, EventArgs e)
        {
            ClearFields();
            LoadSuppliers();
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            if (!ValidateFields())
            {
                return;
            }

            this.supplier = new Supplier(this.supplierId, this.supplierName,
                this.address, this.city, this.postalCode, this.phoneNumber);

            if (SuppliersCollection.AddSupplier(this.supplier))
            {
                this.grdSuppliers.Rows.Add(this.supplierId, this.supplierName,
this.phoneNumber,
                    this.address, this.city, this.postalCode);
                ClearFields();
            }
            else
            {
                MessageBox.Show("This Supplier Id already exists.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                return;
            }
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            if (!ValidateFields())
            {
                return;
            }

            // Updating the properties of the Supplier this.supplier.
            this.supplier.SupplierId = this.supplierId;
            this.supplier.Name = this.supplierName;
            this.supplier.PhoneNumber = this.phoneNumber;
            this.supplier.Address = this.address;
            this.supplier.City = this.city;
            this.supplier.PostalCode = this.postalCode;

            // At this point this.supplier refers to an object inside SuppliersCollection.
            // So the property updates above have also updated the Supplier object inside
            // that collection (refer to grdSuppliers_CellMouseDoubleClick).

            // Grabbing the selected row.
            int selectedRow =
this.grdSuppliers.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            DataGridViewRow row = this.grdSuppliers.Rows[selectedRow];

            // Updating the row data.
            row.Cells["colSupplierId"].Value = this.supplier.SupplierId;
            row.Cells["colSupplierName"].Value = this.supplier.Name;
            row.Cells["colPhoneNumber"].Value = this.supplier.PhoneNumber;
            row.Cells["colAddress"].Value = this.supplier.Address;
            row.Cells["colCity"].Value = this.supplier.City;
            row.Cells["colPostalCode"].Value = this.supplier.PostalCode;

            ClearFields();
        }

        private void btnRemove_Click(object sender, EventArgs e)
        {
            GetSelection();

            //TODO: check if the supplier is assigned to any product (book/software).

            // Grabbing the selected row and removing it.
            int selectedRow =
this.grdSuppliers.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            if (selectedRow >= 0)
            {
                // Removing the selected row.
                this.grdSuppliers.Rows.RemoveAt(selectedRow);
                // Removing the supplier from SuppliersCollection.
                SuppliersCollection.RemoveSupplier(this.supplier);
            }

            ClearFields();
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            ClearFields();
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            HiTechDB.SaveData();
        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        private void grdSuppliers_CellMouseDoubleClick(object sender,
DataGridViewCellMouseEventArgs e)
        {
            // Ignoring double-clicks on the fixed cells.
            if (e.RowIndex >= 0)
            {
                GetSelection();
            }
        }

        private void ClearFields()
        {
            // Clearing the textboxes and maskedtextboxes.
            foreach (Control ctrl in grpSupplierInformation.Controls)
            {
                if (ctrl is TextBox)
                {
                    (ctrl as TextBox).Clear();
                }
                if (ctrl is MaskedTextBox)
                {
                    (ctrl as MaskedTextBox).Clear();
                }
            }

            this.txtSupplierId.Text = SuppliersCollection.NextSupplierId();

            if (this.Visible)
            {
                this.txtSupplierName.Focus();
            }
        }

        private void GetSelection()
        {
            // Retrieves supplier information from the selected row in grdSuppliers.
            // The information is loaded into the non-UI fields (declared at the
            // beginning of this class) and is shown in the UI fields so the
            // logged in user can update/remove it.

            int selectedRow =
this.grdSuppliers.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            if (selectedRow >= 0)
            {
                DataGridViewRow row = this.grdSuppliers.Rows[selectedRow];

                // Retrieving supplier information from the selected row.
                this.supplierId = row.Cells["colSupplierId"].Value.ToString();
                this.supplierName = row.Cells["colSupplierName"].Value.ToString();
                this.phoneNumber = row.Cells["colPhoneNumber"].Value.ToString();
                this.address = row.Cells["colAddress"].Value.ToString();
                this.city = row.Cells["colCity"].Value.ToString();
                this.postalCode = row.Cells["colPostalCode"].Value.ToString();

                // Retrieving the selected supplier from SuppliersCollection
                // and storing it in the Supplier this.supplier.
                this.supplier = SuppliersCollection.GetSupplier(this.supplierId);

                // Showing the supplier information in the UI fields.
                this.txtSupplierId.Text = this.supplierId;
                this.txtSupplierName.Text = this.supplierName;
                this.mskPhoneNumber.Text = this.phoneNumber;
                this.txtAddress.Text = this.address;
                this.txtCity.Text = this.city;
                this.mskPostalCode.Text = this.postalCode;
            }
        }

        private void LoadSuppliers()
        {
            GenericList<string, Supplier> supplierList = SuppliersCollection.GetCollection();

            this.grdSuppliers.Rows.Clear();

            for (int i = 0; i < supplierList.GetCount(); i++)
            {
                Supplier supplier = supplierList[i];

                // Creating a new row first as it will include the columns created at design-
time.
                int rowId = this.grdSuppliers.Rows.Add();

                // Grabbing the new row.
                DataGridViewRow row = this.grdSuppliers.Rows[rowId];

                // Adding the data.
                row.Cells["colSupplierId"].Value = supplier.SupplierId;
                row.Cells["colSupplierName"].Value = supplier.Name;
                row.Cells["colPhoneNumber"].Value = supplier.PhoneNumber;
                row.Cells["colAddress"].Value = supplier.Address;
                row.Cells["colCity"].Value = supplier.City;
                row.Cells["colPostalCode"].Value = supplier.PostalCode;
            }
        }

        private bool ValidateFields()
        {
            // Loading the provided supplier information into the
            // non-UI fields (declared at the beginning of this class).
            this.supplierId = this.txtSupplierId.Text.Trim();
            this.supplierName = this.txtSupplierName.Text.Trim();
            this.phoneNumber = this.mskPhoneNumber.Text.Trim();
            this.address = this.txtAddress.Text.Trim();
            this.city = this.txtCity.Text.Trim();
            this.postalCode = this.mskPostalCode.Text.Trim();

            if (string.IsNullOrEmpty(this.supplierName))
            {
                MessageBox.Show("Please enter the supplier's name.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtSupplierName.Focus();
                return false;
            }

            if (string.IsNullOrEmpty(this.phoneNumber))
            {
                MessageBox.Show("Please enter the supplier's phone number (format: XXX XXX
XXXX).", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.mskPhoneNumber.Focus();
                return false;
            }
            else
            {
                // this.phoneNumber has only the digits.
                Match m = Regex.Match(this.phoneNumber, @"^\d{10}$");
                if (!m.Success)
                {
                    MessageBox.Show("The supplier's phone number is in an incorrect format.
Please use: XXX XXX XXXX", this.Text,
                        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    this.mskPhoneNumber.Focus();
                    return false;
                }
            }

            if (string.IsNullOrEmpty(this.address))
            {
                MessageBox.Show("Please enter the supplier's address.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtAddress.Focus();
                return false;
            }

            if (string.IsNullOrEmpty(this.city))
            {
                MessageBox.Show("Please enter the supplier's city.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtCity.Focus();
                return false;
            }

            if (string.IsNullOrEmpty(this.postalCode))
            {
                MessageBox.Show("Please enter the supplier's postal code (format A1A 1A1,
where A is a letter and 1 is a digit).", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.mskPostalCode.Focus();
                return false;
            }
            else
            {
                // Canadian postal code regular expression validation taken from:
                // http://stackoverflow.com/questions/1146202/canadian-postal-code-validation
                Match m = Regex.Match(this.postalCode, @"^[ABCEGHJKLMNPRSTVXY][0-
9][ABCEGHJKLMNPRSTVWXYZ][0-9][ABCEGHJKLMNPRSTVWXYZ][0-9]$");
                if (!m.Success)
                {
                    MessageBox.Show("This is not a valid postal code. Please try again.",
this.Text,
                        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    this.mskPostalCode.Focus();
                    return false;
                }
            }

            return true;
        }
    }
}
```

## UserForm

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

namespace Solution_HitechSystem.GUI
{
    public partial class UserForm : Form
    {
        private MainForm mainForm;

        private User user = new User();
        private Permission permission = new Permission();
        private string username = string.Empty;
        private string employeeId = default(int).ToString();
        // To control the edition modes.
        private bool isInAdditionMode = false;
        private bool isInUpdateDeletionMode = false;

        public UserForm()
        {
            InitializeComponent();
        }

        public UserForm(MainForm mainForm)
        {
            InitializeComponent();
            this.mainForm = mainForm;
        }

        // Helper class to handle Combobox items.
        // From http://stackoverflow.com/questions/3063320/combobox-adding-text-and-value-to-an-item-no-binding-source
        class ComboboxItem
        {
            public string Text { get; set; }
            public object Value { get; set; }

            public override string ToString()
            {
                return Text;
            }
        }

        private void UserForm_Load(object sender, EventArgs e)
        {
            ClearFields();
            LoadUsers();
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            ClearFields();
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            if (!this.isInAdditionMode)
            {
                return;
            }

            if (!ValidateFields())
            {
                return;
            }

            // All users are created with the password equal to the username.
            this.user = new User(this.username, this.username, "", this.employeeId);

            // Retrieving the employee associated with the user in order to show the full
name.
            Employee employee = EmployeesCollection.GetEmployee(this.employeeId);

            if (UsersCollection.AddUser(this.user))
            {
                // The user was successfully added to UsersCollection.

                // Creating the Permission object.
                this.permission = new Permission(this.username,
                    chkEmployeeSaveUpdateDelete.Checked,
                    chkEmployeeSearchList.Checked,
                    chkUserSaveUpdateDelete.Checked,
                    chkUserSearchList.Checked,
                    chkClientSaveUpdateDelete.Checked,
                    chkClientSearchList.Checked,
                    chkProductSaveUpdateDelete.Checked,
                    chkProductSearchList.Checked,
                    chkClientsOrdersSaveUpdateCancel.Checked,
                    chkClientsOrdersSearchList.Checked);
                // Adding the Permission objec to PermissionsCollection.
                PermissionsCollection.AddPermission(this.permission);

                // Adding the row to grdUsers.
                this.grdUsers.Rows.Add(this.username, this.employeeId, employee.FullName,
                    chkEmployeeSaveUpdateDelete.Checked,
                    chkEmployeeSearchList.Checked,
                    chkUserSaveUpdateDelete.Checked,
                    chkUserSearchList.Checked,
                    chkClientSaveUpdateDelete.Checked,
                    chkClientSearchList.Checked,
                    chkProductSaveUpdateDelete.Checked,
                    chkProductSearchList.Checked,
                    chkClientsOrdersSaveUpdateCancel.Checked,
                    chkClientsOrdersSearchList.Checked);

                ClearFields();
            }
            else
            {
                MessageBox.Show("This Username already exists.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                return;
            }
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            // Once a user is created the only editing allowed is changing permissions.
            // In order to change the username or assign a different employee to it
            // the user must be removed and created again.

            if (!this.isInUpdateDeletionMode)
            {
                return;
            }

            // If the user being updated is the one logged into the system.
            if (this.user.Username == this.mainForm.UserLoggedIn.Username)
            {
                // If this is the case then the permissions to use this screen cannot be
unchecked.
                if ((!chkUserSaveUpdateDelete.Checked) || (!chkUserSearchList.Checked))
                {
                    MessageBox.Show("You cannot remove your permissions to
Save/Update/Delete/Search/List users.", this.Text,
                        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    return;
                }
            }

            // Updating the properties of the Permission object this.permission.
            // The Permission object this.permission holds a reference to an object
            // inside PermissionsCollection (refer to grdUsers_CellMouseDoubleClick).
            // So the property updates below will also update the Permission object inside
that collection.
            this.permission.EmployeeSaveUpdateDelete =
this.chkEmployeeSaveUpdateDelete.Checked;
            this.permission.EmployeeSearchList = this.chkEmployeeSearchList.Checked;
            this.permission.UserSaveUpdateDelete = this.chkUserSaveUpdateDelete.Checked;
            this.permission.UserSearchList = this.chkUserSearchList.Checked;
            this.permission.ClientSaveUpdateDelete = this.chkClientSaveUpdateDelete.Checked;
            this.permission.ClientSearchList = this.chkClientSearchList.Checked;
            this.permission.ProductSaveUpdateDelete = this.chkProductSaveUpdateDelete.Checked;
            this.permission.ProductSearchList = this.chkProductSearchList.Checked;
            this.permission.ClientOrderSaveUpdateCancel =
this.chkClientsOrdersSaveUpdateCancel.Checked;
            this.permission.ClientOrderSearchList = this.chkClientsOrdersSearchList.Checked;

            // Grabbing the selected row.
            int selectedRow =
this.grdUsers.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            DataGridViewRow row = this.grdUsers.Rows[selectedRow];

            // Updating the row data.
            row.Cells["colEmployeeSaveUpdateDelete"].Value =
this.permission.EmployeeSaveUpdateDelete;
            row.Cells["colEmployeeSearchList"].Value = this.permission.EmployeeSearchList;
            row.Cells["colUserSaveUpdateDelete"].Value = this.permission.UserSaveUpdateDelete;
            row.Cells["colUserSearchList"].Value = this.permission.UserSearchList;
            row.Cells["colClientSaveUpdateDelete"].Value = 
this.permission.ClientSaveUpdateDelete;
            row.Cells["colClientSearchList"].Value = this.permission.ClientSearchList;
            row.Cells["colProductSaveUpdateDelete"].Value = 
this.permission.ProductSaveUpdateDelete;
            row.Cells["colProductSearchList"].Value = this.permission.ProductSearchList;
            row.Cells["colClientOrderSaveUpdateCancel"].Value = 
this.permission.ClientOrderSaveUpdateCancel;
            row.Cells["colClientOrderSearchList"].Value = 
this.permission.ClientOrderSearchList;

            ClearFields();
        }

        private void btnRemove_Click(object sender, EventArgs e)
        {
            SetUpdateDeletionMode();

            // Checking if the user being removed is the one logged into the system.
            if (this.user.Username == this.mainForm.UserLoggedIn.Username)
            {
                MessageBox.Show("You cannot remove yourself from the system.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                return;
            }

            // Grabbing the selected row and removing it.
            int selectedRow = 
this.grdUsers.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            if (selectedRow >= 0)
            {
                // Removing the selected row.
                this.grdUsers.Rows.RemoveAt(selectedRow);
                // Removing the user and the associated permissions from
                // UsersCollection and PermissionsCollection respectively.
                UsersCollection.RemoveUser(this.user);
                PermissionsCollection.RemovePermission(this.permission);
            }

            ClearFields();
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            HiTechDB.SaveData();
        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void grdUsers_CellMouseDoubleClick(object sender, 
DataGridViewCellMouseEventArgs e)
        {
            // Ignoring double-clicks on the fixed cells.
            if (e.RowIndex >= 0)
            {
                SetUpdateDeletionMode();
            }
        }

        private void ClearFields()
        {
            SetAdditionMode();
            LoadEmployees();

            // Clearing the permission checkboxes.
            foreach (Control ctrl in grpPermissions.Controls)
            {
                if (ctrl is CheckBox)
                {
                    (ctrl as CheckBox).Checked = false;
                }
            }

            if (this.Visible)
            {
                this.txtUsername.Focus();
            }
        }

        private void GetSelection()
        {
            // Retrieves user and permission information from the selected row in grdUsers.
            // The information is loaded into the non-UI fields (declared at the
            // beginning of this class) and is shown in the UI fields so the
            // logged in user can update/remove it.

            int selectedRow =
this.grdUsers.Rows.GetFirstRow(DataGridViewElementStates.Selected);
            if (selectedRow >= 0)
            {
                DataGridViewRow row = this.grdUsers.Rows[selectedRow];

                // Retrieving user information from the selected row.
                this.username = row.Cells["colUsername"].Value.ToString();
                this.employeeId = row.Cells["colEmployeeId"].Value.ToString();
                string employeeFullName = row.Cells["colEmployeeFullName"].Value.ToString();

                // Retrieving the selected user from UsersCollection
                // and storing it in the User this.user.
                this.user = UsersCollection.GetUser(this.username);

                // Retrieving the selected user's permissions from PermissionsCollection
                // and storing it in the Permission this.permission.
                this.permission =
PermissionsCollection.GetPermissionByUsername(this.username);

                // Showing the user information in the fields.
                this.txtUsername.Text = this.username;
                this.cmbEmployee.Text = employeeFullName;
                // Showing the user's permissions.
                // The permissions are being loaded from Permission object instantiated above
                // because it is easier than loading them from grdUsers.
                this.chkEmployeeSaveUpdateDelete.Checked =
this.permission.EmployeeSaveUpdateDelete;
                this.chkEmployeeSearchList.Checked = this.permission.EmployeeSearchList;
                this.chkUserSaveUpdateDelete.Checked = this.permission.UserSaveUpdateDelete;
                this.chkUserSearchList.Checked = this.permission.UserSearchList;
                this.chkClientSaveUpdateDelete.Checked =
this.permission.ClientSaveUpdateDelete;
                this.chkClientSearchList.Checked = this.permission.ClientSearchList;
                this.chkProductSaveUpdateDelete.Checked =
this.permission.ProductSaveUpdateDelete;
                this.chkProductSearchList.Checked = this.permission.ProductSearchList;
                this.chkClientsOrdersSaveUpdateCancel.Checked =
this.permission.ClientOrderSaveUpdateCancel;
                this.chkClientsOrdersSearchList.Checked =
this.permission.ClientOrderSearchList;
            }
        }

        private void LoadEmployees()
        {
            GenericList<string, Employee> employeeList =
EmployeesCollection.GetEmployeesWithNoUserAssigned();

            this.cmbEmployee.Items.Clear();
            this.cmbEmployee.DisplayMember = "Text";
            this.cmbEmployee.ValueMember = "Value";

            for (int i = 0; i < employeeList.GetCount(); i++)
            {
                Employee employee = employeeList[i];

                ComboboxItem item = new ComboboxItem();
                item.Value = employee.EmployeeId;
                item.Text = employee.FullName;

                this.cmbEmployee.Items.Add(item);
            }
        }

        private void LoadUsers()
        {
            GenericList<string, User> userList = UsersCollection.GetCollection();

            this.grdUsers.Rows.Clear();

            for (int i = 0; i < userList.GetCount(); i++)
            {
                User user = userList[i];
                Employee employee = EmployeesCollection.GetEmployee(user.EmployeeId);
                Permission permission =
PermissionsCollection.GetPermissionByUsername(user.Username);

                // Creating a new row first as it will include the columns created at design-
time.
                int rowId = this.grdUsers.Rows.Add();

                // Grabbing the new row.
                DataGridViewRow row = this.grdUsers.Rows[rowId];

                // Adding the data.
                row.Cells["colUsername"].Value = user.Username;
                row.Cells["colEmployeeId"].Value = employee.EmployeeId;
                row.Cells["colEmployeeFullName"].Value = employee.FullName;
                row.Cells["colEmployeeSaveUpdateDelete"].Value =
permission.EmployeeSaveUpdateDelete;
                row.Cells["colEmployeeSearchList"].Value = permission.EmployeeSearchList;
                row.Cells["colUserSaveUpdateDelete"].Value = permission.UserSaveUpdateDelete;
                row.Cells["colUserSearchList"].Value = permission.UserSearchList;
                row.Cells["colClientSaveUpdateDelete"].Value =
permission.ClientSaveUpdateDelete;
                row.Cells["colClientSearchList"].Value = permission.ClientSearchList;
                row.Cells["colProductSaveUpdateDelete"].Value =
permission.ProductSaveUpdateDelete;
                row.Cells["colProductSearchList"].Value = permission.ProductSearchList;
                row.Cells["colClientOrderSaveUpdateCancel"].Value =
permission.ClientOrderSaveUpdateCancel;
                row.Cells["colClientOrderSearchList"].Value =
permission.ClientOrderSearchList;
            }
        }

        private void SetAdditionMode()
        {
            this.isInAdditionMode = true;
            this.isInUpdateDeletionMode = false;

            this.txtUsername.Clear();
            this.txtUsername.ReadOnly = false;

            this.cmbEmployee.Text = string.Empty;
            this.cmbEmployee.DropDownStyle = ComboBoxStyle.DropDownList;
            this.cmbEmployee.Enabled = true;
        }

        private void SetUpdateDeletionMode()
        {
            this.isInAdditionMode = false;
            this.isInUpdateDeletionMode = true;

            this.txtUsername.ReadOnly = true;

            this.cmbEmployee.DropDownStyle = ComboBoxStyle.DropDown;
            this.cmbEmployee.Enabled = false;

            GetSelection();
        }

        private bool ValidateFields()
        {
            this.username = this.txtUsername.Text.Trim();

            if (string.IsNullOrEmpty(this.username))
            {
                MessageBox.Show("Please enter the username.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.txtUsername.Focus();
                return false;
            }

            if (this.cmbEmployee.SelectedIndex == -1)
            {
                MessageBox.Show("Please assign an employee to this user.", this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                this.cmbEmployee.Focus();
                return false;
            }
            this.employeeId = ((ComboboxItem)this.cmbEmployee.SelectedItem).Value.ToString();

            return true;
        }
    }
}
```