**Lab 3 – Windows Forms Application**

**Guide**

Contents

[1. Create Data Transfer Layer (folder DTL), including the following classes: 2](#_Toc436394369)

[2. Create Database Access Layer (folder DAL), including the following classes: 2](#_Toc436394370)

[3. Create GUI layer (folder GUI) 4](#_Toc436394371)

[3.1. MainGUI form 4](#_Toc436394372)

[3.1.1. MenuStrip control named menuStrip1 4](#_Toc436394373)

[3.1.2. Panel control named panel1 4](#_Toc436394374)

[3.2. BookGUI form 5](#_Toc436394375)

[3.3. BorrowGUI form 13](#_Toc436394376)

[3.4. ReturnGUI form 18](#_Toc436394377)

# Create Data Transfer Layer (folder DTL), including the following classes:

1. Borrower
2. Book
3. Copy
4. CirculatedCopy
5. Reservation

# Create Database Access Layer (folder DAL), including the following classes:

1. DAO
2. BorrowerDAO
3. BookDAO
4. CopyDAO
5. CirculatedCopyDAO
6. ReservationDAO

Note:

* Class DAO contains 2 methods: GetDataTable(string cmd) and Update(SqlCommand cmd)
* Remaining classes contains methods (by using class DAO): GetDataTable(), Insert(), Update(), Delete()… to select a table, insert, update, and delete a row of a table in database Library respectively.

For example:

class DAO

{

static string strConn = ConfigurationManager.ConnectionStrings["LibraryConnectionString"].ConnectionString;

static public DataTable GetDataTable(string sqlSelect)

{

try

{

SqlConnection conn = new SqlConnection(strConn);

SqlDataAdapter da = new SqlDataAdapter(sqlSelect, conn);

DataSet ds = new DataSet();

da.Fill(ds);

return ds.Tables[0];

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return null;

}

}

static public bool UpdateTable(SqlCommand cmd)

{

try

{

SqlConnection conn = new SqlConnection(strConn);

cmd.Connection = conn;

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

return true;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return false;

}

}

}

class BookDAO

{

public static DataTable GetDataTable()

{

string cmd = "select \* from Book";

return DAO.GetDataTable(cmd);

}

public static int GetBookNumberMax()

{

DataTable dt = GetDataTable();

if (dt.Rows.Count == 0) return 0;

else return (int)(dt.Compute("max(bookNumber)", ""));

}

public static bool Insert(Book b)

{

SqlCommand cmd = new SqlCommand("insert into book(bookNumber, title, authors, publisher)" +

"values (@bookNumber, @title, @authors, @publisher)");

return DAO.UpdateTable(cmd);

}

public static bool Update(Book b)

{

SqlCommand cmd = new SqlCommand("update Book set title=@title, authors = @authors, publisher = @publisher where bookNumber=@bookNumber");

cmd.Parameters.AddWithValue("@bookNumber", b.BookNumber);

cmd.Parameters.AddWithValue("@title", b.Title);

cmd.Parameters.AddWithValue("@authors", b.Authors);

cmd.Parameters.AddWithValue("@publisher", b.Publisher);

return DAO.UpdateTable(cmd);

}

public static Boolean Delete(int bookNumber)

{

SqlCommand cmd = new SqlCommand("delete Book where bookNumber=@bookNumber");

cmd.Parameters.AddWithValue("@bookNumber", bookNumber);

return DAO.UpdateTable(cmd);

}

}

# Create GUI layer (folder GUI)

## MainGUI form

Use two controls: MenuStrip and Panel:

### MenuStrip control named menuStrip1

1. Add menu items: Book, Member, Borrow, Return, Reserve
2. Set property ImageScalingSize of menuStrip1: 50, 50.
3. Set property Image of each menu item to according icon file.

### Panel control named panel1

To show a form in the panel1 (for example BookGUI form) by using the following statement:

embed(panel1, new BookGUI());

Method embed() is shown as below

private void embed(Panel panel, Form f)

{

panel.Controls.Clear();

f.FormBorderStyle = FormBorderStyle.None;

f.TopLevel = false;

f.Show();

panel.Controls.Add(f);

}

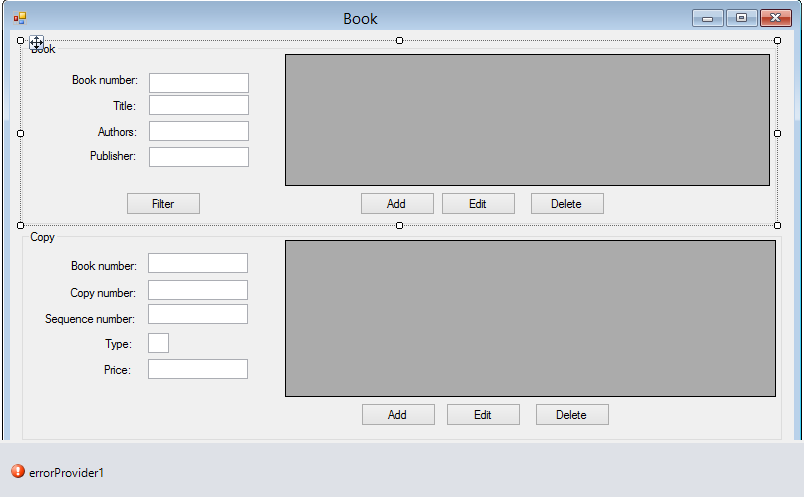
## BookGUI form

btnDelete

btnFilter

btnAdd

btnEdit



txtPrice

txtType

txtSequenceNumber

txtCopyNumber

txtCopyBookNumber

txtPublisher

txtAuthors

txtTitle

txtBookNumber

dataGridView2

dataGridView1

btnCopyDelete

btnCopyEdit

btnCopyAdd

public partial class BookGUI : Form

{

DataView dv, dvc;

Book b;

Copy c;

public BookGUI()

{

InitializeComponent();

View();

displayButtons(1);

displayCopyButtons(1);

}

// get books and display

private void View()

{

DataTable dt = BookDAO.GetDataTable();

dv = new DataView(dt);

dataGridView1.DataSource = dv;

}

// filter the list by bookNumber

private void btnFilter\_Click(object sender, EventArgs e)

{

int bookNumber;

try

{

bookNumber = int.Parse(txtBookNumber.Text);

}

catch

{

MessageBox.Show("Book number must be integer (empty for all books)!");

if (txtBookNumber.Text != "") return;

else bookNumber = -1;

}

if (bookNumber > -1) dv.RowFilter = "bookNumber = " + bookNumber.ToString();

}

private void btnAdd\_Click(object sender, EventArgs e)

{

Button btn = (Button) sender;

if (btn.Text == "Add")

{

b = new Book(BookDAO.GetBookNumberMax() + 1, "", "", "");

displayBook();

displayButtons(2);

}

else

{

b.Title = txtTitle.Text;

b.Authors = txtAuthors.Text;

b.Publisher = txtPublisher.Text;

BookDAO.Insert(b);

// Add to DataTable

DataTable dt = dv.Table;

DataRow newRow = dt.NewRow();

newRow["bookNumber"] = b.BookNumber;

newRow["title"] = b.Title;

newRow["authors"] = b.Authors;

newRow["publisher"] = b.Publisher;

dt.Rows.Add(newRow);

dv.RowFilter = "";

displayButtons(1);

}

}

private void displayButtons(int state)

{

switch (state)

{

// allow to filter/Add/Edit/Delete book

case 1:

txtBookNumber.Enabled = true;

txtBookNumber.Focus();

txtTitle.Enabled = false;

txtAuthors.Enabled = false;

txtPublisher.Enabled = false;

btnFilter.Enabled = true;

btnAdd.Text = "Add";

btnAdd.Enabled = true;

btnEdit.Text = "Edit";

btnEdit.Enabled = true;

btnDelete.Enabled = true;

break;

// Allow to add a book

case 2:

txtBookNumber.Enabled = false;

txtTitle.Enabled = true;

txtAuthors.Enabled = true;

txtPublisher.Enabled = true;

txtTitle.Focus();

btnAdd.Text = "Save";

btnAdd.Enabled = true;

btnFilter.Enabled = false;

btnEdit.Enabled = false;

btnDelete.Enabled = false;

break;

// Allow to edit a book

case 3:

txtBookNumber.Enabled = false;

txtTitle.Enabled = true;

txtAuthors.Enabled = true;

txtPublisher.Enabled = true;

txtTitle.Focus();

btnEdit.Text = "Save";

btnEdit.Enabled = true;

btnFilter.Enabled = false;

btnAdd.Enabled = false;

btnDelete.Enabled = false;

break;

}

}

private void displayBook()

{

txtBookNumber.Text = b.BookNumber.ToString();

txtTitle.Text = b.Title;

txtPublisher.Text = b.Publisher;

txtAuthors.Text = b.Authors;

}

private void btnEdit\_Click(object sender, EventArgs e)

{

Button btn = (Button)sender;

if (btn.Text == "Edit")

{

if (!isSelected()) return;

txtBookNumber.Text = dataGridView1.SelectedRows[0].Cells["bookNumber"].Value.ToString();

txtTitle.Text = dataGridView1.SelectedRows[0].Cells["title"].Value.ToString();

txtAuthors.Text = dataGridView1.SelectedRows[0].Cells["authors"].Value.ToString();

txtPublisher.Text = dataGridView1.SelectedRows[0].Cells["publisher"].Value.ToString();

displayButtons(3);

}

else

{

b = new Book(int.Parse(txtBookNumber.Text), txtTitle.Text, txtAuthors.Text, txtPublisher.Text);

BookDAO.Update(b);

// Update in DataTable

DataTable dt = dv.Table;

DataRow[] result = dt.Select("bookNumber = " + b.BookNumber);

DataRow row = result[0];

row["title"] = b.Title;

row["authors"] = b.Authors;

row["publisher"] = b.Publisher;

dv.RowFilter = "";

displayButtons(1);

}

}

private void btnDelete\_Click(object sender, EventArgs e)

{

if (!isSelected()) return;

int bookNumber = (int) dataGridView1.SelectedRows[0].Cells["bookNumber"].Value;

DialogResult dr = MessageBox.Show(String.Format("Do you want to delete this book number {0}?", bookNumber), "Confirm deteting", MessageBoxButtons.YesNo);

if (dr == DialogResult.No) return;

BookDAO.Delete(bookNumber);

// Delete in DataTable

DataTable dt = dv.Table;

DataRow[] result = dt.Select("bookNumber = " + bookNumber);

result[0].Delete();

}

private bool isSelected()

{

if (dataGridView1.SelectedRows.Count == 0)

{

MessageBox.Show("You must select a book in the list of books!");

return false;

}

if (dataGridView1.SelectedRows[0].Cells["bookNumber"].Value == null)

{

MessageBox.Show("You must select a book in the list of books!");

return false;

}

return true;

}

private void ViewCopies()

{

if (dataGridView1.SelectedRows.Count == 0) return;

int bookNumber = (int)dataGridView1.SelectedRows[0].Cells["bookNumber"].Value;

txtCopyBookNumber.Text = bookNumber.ToString();

dvc = new DataView(CopyDAO.GetDataTable(););

dvc.RowFilter = "bookNumber = " + bookNumber;

dataGridView2.DataSource = dvc;

}

private bool isCopySelected()

{

if (dataGridView2.SelectedRows.Count == 0)

{

MessageBox.Show("You must select a copy in the list of copies!");

return false;

}

if (dataGridView2.SelectedRows[0].Cells["copyNumber"].Value == null)

{

MessageBox.Show("You must select a copy in the list of copies!");

return false;

}

return true;

}

private void displayCopyButtons(int stateCopy)

{

switch (stateCopy)

{

// Allow to Add/Edit/delete copy

case 1:

txtType.Enabled = false;

txtPrice.Enabled = false;

btnCopyDelete.Enabled = true;

btnCopyAdd.Text = "Add";

btnCopyAdd.Enabled = true;

btnCopyEdit.Text = "Edit";

btnCopyEdit.Enabled = true;

break;

// Allow to add a copy

case 2:

txtType.Enabled = true;

txtPrice.Enabled = true;

txtPrice.Focus();

btnCopyAdd.Text = "Save";

btnCopyAdd.Enabled = true;

btnCopyEdit.Text = "Edit";

btnCopyEdit.Enabled = false;

btnCopyDelete.Enabled = false;

break;

// Allow to edit a copy

case 3:

txtType.Enabled = true;

txtPrice.Enabled = true;

txtPrice.Focus();

btnCopyAdd.Text = "Add";

btnCopyAdd.Enabled = false;

btnCopyEdit.Text = "Save";

btnCopyEdit.Enabled = true;

btnCopyDelete.Enabled = false;

break;

}

}

private void displayCopy()

{

txtCopyBookNumber.Text = c.BookNumber.ToString();

txtCopyNumber.Text = c.CopyNumber.ToString();

txtSequenceNumber.Text = c.SequenceNumber.ToString();

txtType.Text = c.Type.ToString();

txtPrice.Text = c.Price.ToString();

}

private void btnCopyAdd\_Click(object sender, EventArgs e)

{

Button btn = (Button) sender;

if (btn.Text == "Add")

{

if (!isSelected()) return;

txtCopyBookNumber.Text = dataGridView1.SelectedRows[0].Cells["bookNumber"].Value.ToString();

c = new Copy(int.Parse(txtCopyBookNumber.Text), CopyDAO.GetCopyNumberMax() + 1, CopyDAO.GetSequenceNumberMax(int.Parse(txtCopyBookNumber.Text)) + 1, 'A', 0);

displayCopy();

displayCopyButtons(2);

}

else

{

c.Price = double.Parse(txtPrice.Text);

CopyDAO.Insert(c);

ViewCopies();

displayCopyButtons(1);

}

}

private void btnCopyEdit\_Click(object sender, EventArgs e)

{

Button btn = (Button)sender;

if (btn.Text == "Edit")

{

if (!isCopySelected()) return;

if (txtType.Text[0] == 'B')

{

MessageBox.Show("Can't edit borrowed copy!");

return;

}

displayCopyButtons(3);

}

else

{

c = new Copy(int.Parse(txtCopyBookNumber.Text), int.Parse(txtCopyNumber.Text), int.Parse(txtSequenceNumber.Text), txtType.Text[0], double.Parse(txtPrice.Text));

CopyDAO.Update(c);

ViewCopies();

displayCopyButtons(1);

}

}

private void btnCopyDelete\_Click(object sender, EventArgs e)

{

if (!isCopySelected()) return;

int copyNumber = (int)dataGridView2.SelectedRows[0].Cells["copyNumber"].Value;

DialogResult dr = MessageBox.Show(String.Format("Do you want to delete this copy number {0}?", copyNumber), "Confirm deteting", MessageBoxButtons.YesNo);

if (dr == DialogResult.No) return;

CopyDAO.Delete(copyNumber);

ViewCopies();

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0) return;

txtBookNumber.Text = dataGridView1.Rows[e.RowIndex].Cells["bookNumber"].Value.ToString();

txtCopyBookNumber.Text = txtBookNumber.Text;

txtTitle.Text = dataGridView1.Rows[e.RowIndex].Cells["title"].Value.ToString();

txtAuthors.Text = dataGridView1.Rows[e.RowIndex].Cells["authors"].Value.ToString();

txtPublisher.Text = dataGridView1.Rows[e.RowIndex].Cells["publisher"].Value.ToString();

ViewCopies();

}

private void dataGridView2\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0) return;

txtCopyBookNumber.Text = dataGridView2.Rows[e.RowIndex].Cells["bookNumber"].Value.ToString();

txtCopyNumber.Text = dataGridView2.Rows[e.RowIndex].Cells["copyNumber"].Value.ToString();

txtSequenceNumber.Text = dataGridView2.Rows[e.RowIndex].Cells["sequenceNumber"].Value.ToString();

txtType.Text = dataGridView2.Rows[e.RowIndex].Cells["type"].Value.ToString();

txtPrice.Text = dataGridView2.Rows[e.RowIndex].Cells["price"].Value.ToString();

}

private void txtType\_Validating(object sender, CancelEventArgs e)

{

if (txtType.Text.Length == 0)

{

// Cancel the event and select the text to be corrected by the user.

e.Cancel = true;

txtType.Select(0, txtType.Text.Length);

// Set the ErrorProvider error with the text to display.

this.errorProvider1.SetError(txtType, "Must be 'A' or 'R'");

return;

}

if (txtType.Text[0] != 'A' && txtType.Text[0] != 'R')

{

// Cancel the event and select the text to be corrected by the user.

e.Cancel = true;

txtType.Select(0, txtType.Text.Length);

// Set the ErrorProvider error with the text to display.

this.errorProvider1.SetError(txtType, "Must be 'A' or 'R'");

return;

}

e.Cancel = false;

}

private void txtType\_Validated(object sender, EventArgs e)

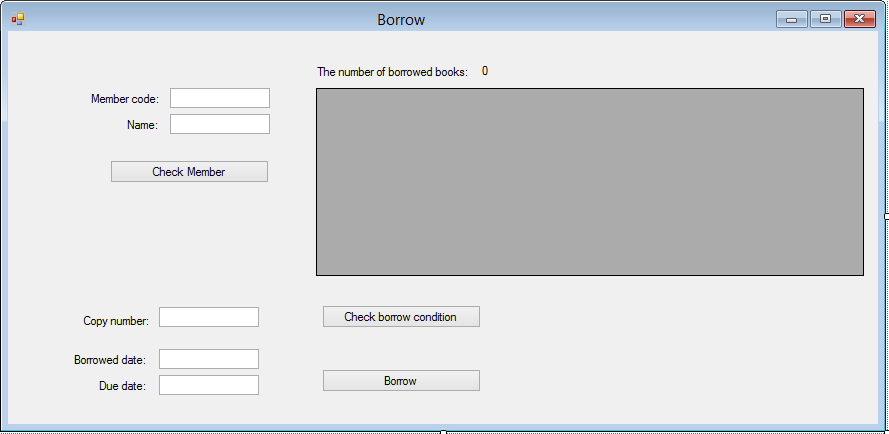
{

errorProvider1.SetError(txtType, "");

}

}

## BorrowGUI form



lblNumberBorrowedBooks

btnBorrow

btnCondition

btnMember

txtDueDate

txtBorrowedDate

txtCopyNumber

txtName

txtBorrowerNumber

dataGridView1

public partial class BorrowGUI : Form

{

DataView dv;

Copy c;

Reservation r;

enum errorBorrow

{

OK,

Connect,

CopyNotExist,

CopyReferenced,

CopyBorrowed,

CopyReserved

};

public BorrowGUI()

{

InitializeComponent();

txtBorrowedDate.Text = DateTime.Now.ToString("dd/MM/yyyy");

txtDueDate.Text = DateTime.Now.AddDays(14).ToString("dd/MM/yyyy");

displayButtons(1);

}

private void btnMember\_Click(object sender, EventArgs e)

{

int borrowerNumber;

try

{

borrowerNumber = int.Parse(txtBorrowerNumber.Text);

}

catch

{

MessageBox.Show("Borrower number must be a integer!");

txtBorrowerNumber.Focus();

return;

}

Borrower b = BorrowerDAO.GetBorrower(borrowerNumber);

if (b == null)

{

MessageBox.Show(String.Format("Borrower number {0} does't exist!", borrowerNumber));

txtBorrowerNumber.Focus();

return;

}

txtName.Text = b.Name;

View();

int n = dataGridView1.Rows.Count;

MessageBox.Show("Number = " + n);

if (dataGridView1.Rows.Count >= 5)

{

MessageBox.Show("You borrowed 5 books, so you can't borrow any more!");

txtBorrowerNumber.Focus();

return;

}

displayButtons(2);

r = null;

c = null;

}

private void btnCondition\_Click(object sender, EventArgs e)

{

int copyNumber;

try

{

copyNumber = int.Parse(txtCopyNumber.Text);

}

catch

{

MessageBox.Show("Copy number must be a integer!");

txtCopyNumber.Focus();

return;

}

errorBorrow error = checkCondition(int.Parse(txtBorrowerNumber.Text), int.Parse(txtCopyNumber.Text), out c, out r);

string str = "";

switch (error)

{

case errorBorrow.CopyNotExist:

str = "No this copy number, so you can't borrow!";

txtCopyNumber.Focus();

break;

case errorBorrow.CopyReferenced:

str = "It is referenced, so you can't borrow!";

txtCopyNumber.Focus();

break;

case errorBorrow.CopyBorrowed:

str = "It is borrwed, so you can't borrow!";

txtCopyNumber.Focus();

break;

case errorBorrow.CopyReserved:

str = "It is reserved by other, so you can't borrow!";

txtCopyNumber.Focus();

break;

case errorBorrow.Connect:

str = "Can't connect to Database, so you can't borrow!";

break;

}

if (str != "")

{

MessageBox.Show(str);

return;

}

displayButtons(3);

}

private errorBorrow checkCondition(int borrowerNumber, int copyNumber, out Copy c, out Reservation r)

{

c = null;

r = null;

c = CopyDAO.GetCopy(copyNumber);

if (c == null) return errorBorrow.CopyNotExist;

if (c.Type == 'R') return errorBorrow.CopyReferenced;

if (c.Type == 'B') return errorBorrow.CopyBorrowed;

// Check if this book is reserved by others and you are not at the first of the reservation list

int bookNumber = c.BookNumber;

r = ReserveDAO.GetFirstReservation(bookNumber);

if(r != null && r.BorrowerNumber != borrowerNumber) return errorBorrow.CopyReserved;

return errorBorrow.OK;

}

private void btnBorrow\_Click(object sender, EventArgs e)

{

try

{

DateTime.ParseExact(txtBorrowedDate.Text, "dd/MM/yyyy", null);

}

catch

{

MessageBox.Show("Format of date is not valid!");

return;

}

CirculatedCopy cc = new CirculatedCopy(int.Parse(txtCopyNumber.Text), int.Parse(txtBorrowerNumber.Text),

DateTime.ParseExact(txtBorrowedDate.Text, "dd/MM/yyyy", null), DateTime.ParseExact(txtDueDate.Text, "dd/MM/yyyy", null));

borrow(cc, c, r);

View();

displayButtons(1);

}

private void borrow(CirculatedCopy cc, Copy c, Reservation r)

{

// Insert borrow record

if (cc == null) return;

CirculatedCopyDAO.Insert(cc);

// update type = 'B' at Copy

if (c == null) return;

c.Type = 'B';

CopyDAO.Update(c);

// update status = 'A' at Reservation

if (r == null) return;

r.Status = 'A';

ReserveDAO.Update(r);

}

// get all borrowed copies by this borrower and display

private void View()

{

DataTable dt = CirculatedCopyDAO.GetBorrowedCopies(int.Parse(txtBorrowerNumber.Text));

dataGridView1.DataSource = dt;

lblNumberBorrowedBooks.Text = dt.Rows.Count.ToString();

this.dataGridView1.Columns["borrowedDate"].DefaultCellStyle.Format = "dd/MM/yyyy";

this.dataGridView1.Columns["dueDate"].DefaultCellStyle.Format = "dd/MM/yyyy";

this.dataGridView1.Columns["returnedDate"].DefaultCellStyle.Format = "dd/MM/yyyy";

}

private void displayButtons(int state)

{

switch (state)

{

// to enter member and check member

case 1:

txtBorrowerNumber.Enabled = true;

btnMember.Enabled = true;

txtCopyNumber.Enabled = false;

btnCondition.Enabled = false;

txtBorrowedDate.Enabled = false;

btnBorrow.Enabled = false;

break;

// to check condition

case 2:

txtBorrowerNumber.Enabled = false;

btnMember.Enabled = false;

txtCopyNumber.Enabled = true;

btnCondition.Enabled = true;

txtBorrowedDate.Enabled = false;

btnBorrow.Enabled = false;

break;

// to borrow

case 3:

txtBorrowerNumber.Enabled = false;

btnMember.Enabled = false;

txtCopyNumber.Enabled = false;

btnCondition.Enabled = false;

txtBorrowedDate.Enabled = true;

btnBorrow.Enabled = true;

break;

}

}

private void txtBorrowedDate\_Validated(object sender, EventArgs e)

{

DateTime dueDate = DateTime.ParseExact(txtBorrowedDate.Text, "dd/MM/yyyy", null).AddDays(14);

txtDueDate.Text = dueDate.ToString("dd/MM/yyyy");

}

private void txtBorrowedDate\_Validating(object sender, CancelEventArgs e)

{

try

{

DateTime.ParseExact(txtBorrowedDate.Text, "dd/MM/yyyy", null);

e.Cancel = false;

}

catch

{

MessageBox.Show("Format of date is not valid!");

txtBorrowedDate.Focus();

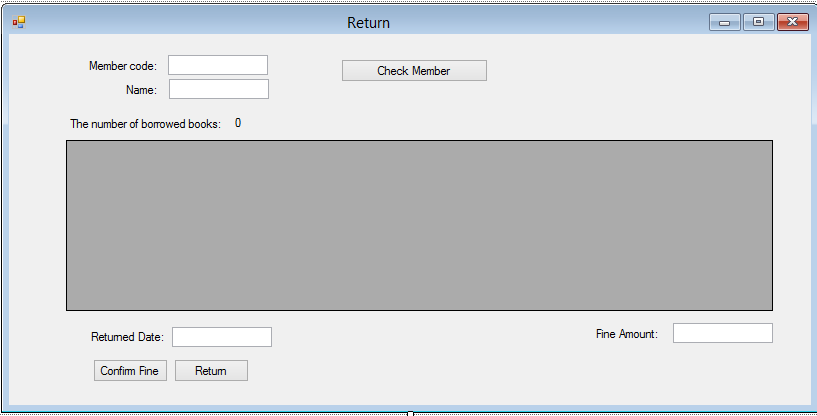
e.Cancel = true;

}

}

}

## ReturnGUI form



public partial class ReturnGUI : Form

{

DataView dv;

public ReturnGUI()

{

InitializeComponent();

displayButtons(1);

txtReturnedDate.Text = DateTime.Now.AddDays(20).ToString("dd/MM/yyyy");

}

private void btnMember\_Click(object sender, EventArgs e)

{

int borrowerNumber;

try

{

borrowerNumber = int.Parse(txtBorrowerNumber.Text);

}

catch

{

MessageBox.Show("Borrower number must be a integer!");

txtBorrowerNumber.Focus();

return;

}

Borrower b = BorrowerDAO.GetBorrower(borrowerNumber);

if (b == null)

{

MessageBox.Show(String.Format("Borrower number {0} does't exist!", borrowerNumber));

txtBorrowerNumber.Focus();

return;

}

txtName.Text = b.Name;

View();

displayButtons(2);

}

private void btnFine\_Click(object sender, EventArgs e)

{

if(!isSelected()) return;

double fine =0;

foreach(DataGridViewRow dr in dataGridView1.SelectedRows)

{

fine += (int) (DateTime.Now.AddDays(20) - Convert.ToDateTime(dr.Cells["dueDate"].Value.ToString())).TotalDays;

}

txtFineAmount.Text = fine.ToString("C2");

displayButtons(3);

}

private void btnReturn\_Click(object sender, EventArgs e)

{

double fine, fineTotal = 0;

try{

DateTime.ParseExact(txtReturnedDate.Text, "dd/MM/yyyy", null);

}

catch{

MessageBox.Show("Format of date must be 'dd/MM/yyyy'!");

return;

}

CirculatedCopy cc;

foreach (DataGridViewRow dr in dataGridView1.SelectedRows)

{

fine = 0;

if (DateTime.ParseExact(txtReturnedDate.Text, "dd/MM/yyyy", null) > Convert.ToDateTime(dr.Cells["dueDate"].Value.ToString()))

fine = (int)(DateTime.Now.AddDays(20) - Convert.ToDateTime(dr.Cells["dueDate"].Value.ToString())).TotalDays;

fineTotal += fine;

cc = new CirculatedCopy(int.Parse(dr.Cells["copyNumber"].Value.ToString()), int.Parse(dr.Cells["borrowerNumber"].Value.ToString()),

Convert.ToDateTime(dr.Cells["borrowedDate"].Value.ToString()), Convert.ToDateTime(dr.Cells["dueDate"].Value.ToString()));

cc.Id = int.Parse(dr.Cells["ID"].Value.ToString());

cc.ReturnedDate = DateTime.ParseExact(txtReturnedDate.Text, "dd/MM/yyyy", null);

cc.FineAmount = fine;

Return(cc);

}

View();

displayButtons(1);

}

private void Return(CirculatedCopy cc)

{

CirculatedCopyDAO.Update(cc);

Copy c = CopyDAO.GetCopy(cc.CopyNumber);

// update type = 'A' in copy

c.Type = 'A';

CopyDAO.Update(c);

}

private bool isSelected()

{

if (dataGridView1.SelectedRows.Count == 0)

{

MessageBox.Show("Don't have any book to return!");

return false;

}

if (dataGridView1.SelectedRows[0].Cells["copyNumber"].Value == null)

{

MessageBox.Show("You must select a book in the list to return!");

return false;

}

return true;

}

private void txtReturnedDate\_Validating(object sender, CancelEventArgs e)

{

try

{

DateTime.ParseExact(txtReturnedDate.Text, "dd/MM/yyyy", null);

e.Cancel = false;

}

catch

{

MessageBox.Show("Format date must be 'dd/MM/yyyy'!");

txtReturnedDate.Focus();

e.Cancel = true;

}

}

// Get borrowed copies by this borrower and display

private void View()

{

DataTable dt = CirculatedCopyDAO.GetBorrowedCopies(int.Parse(txtBorrowerNumber.Text));

dataGridView1.DataSource = dt;

this.dataGridView1.Columns["borrowedDate"].DefaultCellStyle.Format = "dd/MM/yyyy";

this.dataGridView1.Columns["dueDate"].DefaultCellStyle.Format = "dd/MM/yyyy";

this.dataGridView1.Columns["returnedDate"].DefaultCellStyle.Format = "dd/MM/yyyy";

lblNumberBorrowedBooks.Text = dt.Rows.Count.ToString();

}

private void displayButtons(int state)

{

switch (state)

{

// to enter member and check Member

case 1:

txtBorrowerNumber.Enabled = true;

btnMember.Enabled = true;

dataGridView1.Enabled = false;

btnFine.Enabled = false;

txtReturnedDate.Enabled = false;

btnReturn.Enabled = false;

break;

// to Confirm fine

case 2:

txtBorrowerNumber.Enabled = false;

btnMember.Enabled = false;

dataGridView1.Enabled = true;

btnFine.Enabled = true;

txtReturnedDate.Enabled = false;

btnReturn.Enabled = false;

break;

// to Return

case 3:

txtBorrowerNumber.Enabled = false;

btnMember.Enabled = false;

btnFine.Enabled = false;

dataGridView1.Enabled = false;

txtReturnedDate.Enabled = true;

btnReturn.Enabled = true;

break;

}

}

}