# Code Complexities

To highlight the most significant code in my program, I have decided to compile ‘complex code’ in the following pages.

## Template Form / Inheritable Forms

Throughout my usage of forms, I incorporated a template form that would be used as the base class for my other forms.

public partial class cformAddEditBooking : cform

{

}

The above code is an example of one of my child class forms. The cform retains functionality, mainly the back button, menu strip and exit button, which means each inherited form would inherit these functions. Therefore, I cut back on repetitive code that would had needed to be present on each form if I didn’t use a template form.

## Customised Exceptions

### AddEditBooking

* Thrown if office selected is in use (urgent booking)

class AddEditBooking : Exception

{

public AddEditBooking()

{

}

public AddEditBooking(string message): base(message)

{

}

}

### AddEditCustomer

* Thrown if customer information not filled

class AddEditCustomer : Exception

{

public AddEditCustomer()

{

}

public AddEditCustomer(string message): base(message)

{

}

}

## SQL Inner joins

The extensive use of inner-joins where possible allowed me to cut-down on sql methods to achieve certain functionality need

public List<AppointmentBooking> getAllAppointmentBookings(DateTime date, string filter)

{

List<AppointmentBooking> Bookings = new List<AppointmentBooking>();

db.Cmd = db.Con.CreateCommand();

db.Cmd.CommandText = "SELECT AppointmentBooking.BookingId, AppointmentBooking.Date, AppointmentBooking.TimeSlot, Customer.Forename, Customer.Surname, Office.RoomNo, Staff.Forename, Staff.Surname, Service.Name, AppointmentBooking.Status, AppointmentBooking.Deleted, AppointmentBooking.Priority FROM " +

"(AppointmentBooking " +

"INNER JOIN Customer ON AppointmentBooking.CustomerId = Customer.CustomerId " +

"INNER JOIN Office ON AppointmentBooking.OfficeId = Office.OfficeId " +

"INNER JOIN StaffService ON AppointmentBooking.StaffServiceId = StaffService.StaffServiceId) " +

"INNER JOIN (StaffService AS StaffAlias INNER JOIN Staff ON StaffAlias.StaffId = Staff.StaffId) ON StaffAlias.StaffServiceId = StaffService.StaffServiceId " +

"INNER JOIN (StaffService AS ServiceAlias INNER JOIN Service ON ServiceAlias.ServiceId = Service.ServiceId) ON ServiceAlias.StaffServiceId = StaffService.StaffServiceId " +

"WHERE MONTH(AppointmentBooking.Date) = " + date.Month + " AND YEAR(AppointmentBooking.Date) = " + date.Year + " AND AppointmentBooking.Deleted = 0" + filter +

" ORDER BY AppointmentBooking.Date, AppointmentBooking.TimeSlot";

//ADD ORDER BY TIME SLOT

db.Reader = db.Cmd.ExecuteReader();

while (db.Reader.Read())

{

Bookings.Add(getAppointmentBookingFromReader(db.Reader));

}

db.Reader.Close();

return Bookings;

}

## Age Calculation

The following method is included within the utility object which is used in a multitude of ways, for example, checking the customer is of age or display the age of the customer in the view customer form.

public int calcAge(DateTime DOB)

{

int age = Convert.ToInt32(DateTime.Now.Year - DOB.Year);

if (DateTime.Now.DayOfYear < DOB.DayOfYear)

{

age -= 1;

}

return age;

}

## Postcode Check

By using regex, I was able to ensure that postcodes met a specific UK format. The specific method is a return type of bool, which means I am able to make use of this information.

public bool checkPostcode(string postcode)

{

return (new Regex(@"(GIR 0AA)|((([A-Z-[QVX]][0-9][0-9]?)|(([A-Z-[QVX]][A-Z-[IJZ]][0-9][0-9]?)|(([A-Z-[QVX]][0-9][A-HJKSTUW])|([A-Z-[QVX]][A-Z-[IJZ]][0-9][ABEHMNPRVWXY])))) [0-9][A-Z-[CIKMOV]]{2})")).IsMatch(postcode);

}

## Email Check

When validating the email, when a message is not thrown, the email is passed with the Boolean: true. However, when the email is false, an error occurs and to remedy this, I return false – ending the method and indicating the invalidity of the email.

public bool checkEmail(string email)

{

try

{

var validate = new System.Net.Mail.MailAddress(email);

return true;

}

catch

{

return false;

}

}

## Automatic Status Update

In the program.cs, I imbedded the status update code as it’s the main entry for the application. With this code, when a booking is passed todays date, the booking is marked as missed – unless it’s checked in.

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Database db = new Database();

if (!db.connect())

{

MessageBox.Show("Database Connection not opened.", "Error");

Application.Exit();

}

AdviceBookingUtil advBookingUtil = new AdviceBookingUtil(db);

MessageBox.Show("Any upcoming bookings that have passed todays date will be marked as missed.", "Warning");

advBookingUtil.AdviceDBA.BookingDBAccess.updateBookings();

advBookingUtil.FirstStartup = true;

Application.Run(new cformMenu(advBookingUtil));

}

## Schedule

I used a DataTable and nested for loops to efficient fill each row and column of the schedule. In addition, with the use of if-statements, I skipped a column.

public DataTable viewOfficeDiaryDataTable(List<List<string>> bookings)

{

DataTable dT = new DataTable();

DataRow dTRow;

dT.Columns.Add("10:00 - 11:00");

dT.Columns.Add("11:00 - 12:00");

dT.Columns.Add("Lunch");

dT.Columns.Add("13:00 - 14:00");

dT.Columns.Add("14:00 - 15:00");

for (int row = 1; row < 5; row++)

{

dTRow = dT.NewRow();

for (int col = 0; col < 5; col++)

{

if(col != 2)

{

int column = col;

if (col > 2)

{

column -= 1;

}

foreach (List<string> booking in bookings)

{

if (Convert.ToInt32(booking[0]) == row && Convert.ToInt32(booking[1]) == column + 1)

{

dTRow[col] = booking[2];

break;

}

}

//if (dTRow[col].ToString() == string.Empty)

//{

// dTRow[col] = "Open Slot";

//}

}

else //Sets lunch column to empty

{

dTRow[col] = "";

}

}

dT.Rows.Add(dTRow);

}

return dT;

}

## Item (Object) Storage

By overriding the ToString() method for an object, I was able to store objects as an item in combo boxes, which allows me to easily retrieve various data, instead of one piece of data. Specifically, the ToString() method helps popularise the combo box with meaningful text.

public override string ToString()

{

if(customerId == -1)

{

return "(Not Specified)";

}

else

{

return forename + " " + surname + " (ID: " + customerId + ")";

}

}

For example, you can use it when populating the listbox for viewing customers.

public void populateListBox(List<Customer> customers)

{

listBoxCustomers.Items.Clear();

listBoxCustomers.Items.AddRange(customers.ToArray());

if (listBoxCustomers.Items.Count > 0)

{

listBoxCustomers.SelectedIndex = 0;

}

}

As you can see, the following code is efficient, unlike using a data table for example.

## Colour Formatting Data Grid View

To colour coordinate a specific column, I used the following code to colour specific cells.

private void dGridViewMonthlyBookings\_CellFormatting(object sender, DataGridViewCellFormattingEventArgs e)

{

for (int i = 0; i <= dGridViewMonthlyBookings.Rows.Count - 1; i++)

{

if (dGridViewMonthlyBookings.Rows[i].Cells[7].Value.ToString() == "Upcoming")

{

dGridViewMonthlyBookings.Rows[i].Cells[7].Style.BackColor = Color.FromArgb(255, 128, 0);

}

else if (dGridViewMonthlyBookings.Rows[i].Cells[7].Value.ToString() == "Checked In")

{

dGridViewMonthlyBookings.Rows[i].Cells[7].Style.BackColor = Color.LimeGreen;

}

else if (dGridViewMonthlyBookings.Rows[i].Cells[7].Value.ToString() == "Missed")

{

dGridViewMonthlyBookings.Rows[i].Cells[7].Style.BackColor = Color.Red;

}

else if (dGridViewMonthlyBookings.Rows[i].Cells[7].Value.ToString() == "Waiting")

{

dGridViewMonthlyBookings.Rows[i].Cells[7].Style.BackColor = Color.LightBlue ;

}

if(dGridViewMonthlyBookings.Rows[i].Cells[8].Value.ToString() == "Urgent")

{

dGridViewMonthlyBookings.Rows[i].Cells[3].Style.BackColor = Color.Yellow;

}

}

}

## User-friendly Validation

By using the TextChanged event, I was able to deliver an error message as the mistake was made. For example, the method below also removes letters as you press, therefore, removing the invalid text.

private void txtbTeleNo\_TextChanged(object sender, EventArgs e)

{

if (!txtbTeleNo.Text.All(Char.IsDigit))

{

txtbTeleNo.Text = txtbTeleNo.Text.Remove(txtbTeleNo.Text.Length - 1);

txtbTeleNo.SelectionStart = txtbTeleNo.Text.Length;

lblTeleError.Show();

teleIsValid = true;

}

else if (txtbTeleNo.Text == string.Empty || txtbTeleNo.TextLength < 11)

{

lblTeleError.Hide();

lblV6.Text = "Fill";

lblV6.ForeColor = Color.Orange;

}

else

{

lblTeleError.Hide();

lblV6.Text = "Valid";

lblV6.ForeColor = Color.Lime;

teleIsValid = true;

return;

}

teleIsValid = false;

}

## Search functionality for customers

private void txtbListSearch\_TextChanged(object sender, EventArgs e)

{

if (txtbListSearch.Text == "Search...")

{

populateListBox(customers);

}

else if (txtbListSearch.Text != string.Empty)

{

List<Customer> searchResults = new List<Customer>();

foreach (Customer cust in customers)

{

if ((cust.Forename + cust.Surname).ToLower().Replace(" ", string.Empty).Contains(txtbListSearch.Text.ToLower().Replace(" ", string.Empty)))

{

searchResults.Add(cust);

}

}

if (searchResults.Count == 0)

{

panelCustView.Enabled = false;

panelCustName.Enabled = false;

dGridViewUpcoming.Enabled = false;

}

populateListBox(searchResults);

}

else

{

populateListBox(customers);

}

}