40TBITS BANK PROJECT

What our program do?

Our software is used to control the staff and customer's details in the system.

For Customers :

- **Add new data**
- **Check existing data**
- **Control their bank account**

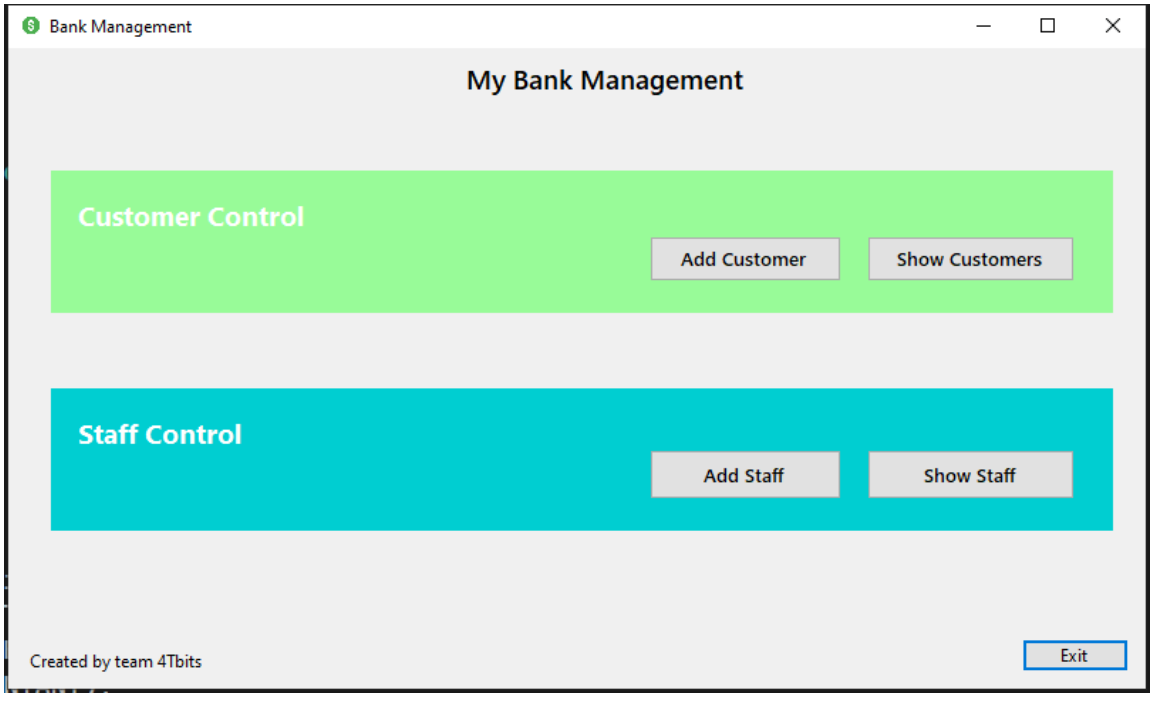
For Staff:

- **Check their data**
- **Add new data**
- **Control their salary**

First of all , our program consists of 3 Main classes: Person , Staff, Customer.

Staff and Customer classes are inherited to the Person class.

Here's our main menu



As it is clear you can either Add or Check existing data.

Add customer

- `private Boolean isAdd;` – Checks whether if we are adding a customer or not
- `private void clearInputs()` – Clears all input data.
- `Boolean isValid()` – Checks if the input data is correct.
- `private void validate(object sender, EventArgs e)` – Makes the contact textbox text color to red if it is wrong.
- `Database.saveCustomer(customer)` – Saves the input data to the new customer object and sends data to our TXT file (database).
- `Utils.displayMenu();` – displays the main menu

```
if (isValid())
{
    Customer customer = new Customer(customerId.Text,
        name.Text,
        lastName.Text,
        contact.Text,
        email.Text,
        address.Text,
        profilePicture.ImageLocation.ToString(),
        balance.Value,
        accountNumber.Text,
        (Plan) plan.SelectedIndex,
        Decimal.Parse(modifySavings.Text)
        );
    if (isAdd)
    {
        Database.saveCustomer(customer);
        Utils.displayMenu();
    }
```

It its not adding the data. We use same Form for editing data in the future.

- `Database.updateCustomer(customer)` – updates the data of existing customer

```
    else
    {
        Database.updateCustomer(customer);
        Utils.display(new ShowCustomer());
    }
```

Show customer

`Database.getCustomer(index) – Gets the customer by INDEX.`

```
Customer customer = Database.getCustomer(index);
name.Text = customer.name;
lastName.Text = customer.lastname;
contact.Text = customer.contact;
email.Text = customer.email;
address.Text = customer.adress;
balance.Text = customer.balance.ToString();
profilePicture.ImageLocation = customer.profilePic;
current.Text = "" + (Database.customerIndex + 1);
customerId.Text = customer.customerId;
accountNumbeer.Text = customer.accountNumber;
savings.Text = customer.savings.ToString();
outOf.Text =" / "+ Database.maxCustomers.ToString();
plan.Text = customer.plan.ToString();
```

Editing the customer data :

`AddCustomer editCustomer = new AddCustomer(false); – Opens the Add Customer form but with the loading the data and modifying it.`
`Utils.display(editCustomer); – Shows the Add customer form with changing label into Edit customer`

```
ссылка: 1
private void editCustomer(object sender, EventArgs e)
{
    AddCustomer editCustomer = new AddCustomer(false);
    Utils.display(editCustomer);
}
Ссылок: 3
```

Add Staff

`if (isValid()) – if the filled data correct`

`Database.saveStaff(staff); – saves the data to the staff object and saves it into TXT (database)`

```
        */
        Staff staff = new Staff(
                staffId.Text,
                name.Text,
                lastName.Text,
                contact.Text,
                email.Text,
                address.Text,
                profilePicture.ImageLocation.ToString(),
                (Role)role.SelectedItem,
                0,
                0,
                balance.Value);
        if (isValid())
        {
            Database.saveStaff(staff);
            clearInputs();
            Utils.display(new BankManagement());
        }
        else MessageBox.Show("Please fill in all the staff details");
    }
    Ссылок: 2
```

Show staff

`Staff staff = Database.getStaff(index)` – loads the data of staff by their INDEX

```
        */
        Staff staff = Database.getStaff(index);

        name.Text = staff.name;
        lastName.Text = staff.lastname;
        email.Text = staff.email;
        address.Text = staff.adress;
        balance.Text = staff.balance.ToString();
        role.Text = staff.role.ToString();
        hours.Text = staff.hours.ToString();
        extraHours.Text = staff.extra_hours.ToString();
        salary.Text = ""+(int)staff.role;
        contact.Text = staff.contact;
        profilePicture.ImageLocation = staff.profilePic;
        page.Text = (Database.staffIndex + 1)+" / "+ Database.maxStaffs;
```

Account Control

`Customer customer = Database.getCustomer(Index.CURENT);` – Gets the customer by its INDEX

`Database.updateCustomer(customer);` – Updates the customer data with the new data

WITHDRAW BUTTON :

```
ссылка: 1
private void button1_Click(object sender, EventArgs e)
{
    //TODO this is where you withdraw
    /*
     * variables
     * ---------
     * withdraw.Valvue
     */
    Customer customer = Database.getCustomer(Index.CURENT);
    decimal result = customer.balance - decimal.Parse(withdraw.Text);
    if (withdraw.Value > customer.balance)
    {
        MessageBox.Show("Withdraw amount is more than balance");
        return;
    }
    else
    customer.balance = result;
    balance.Text = customer.balance.ToString();
    MessageBox.Show($"Withdrawed {withdraw.Text} from the balance");
    Database.updateCustomer(customer);
    clearInputs();
}
```

Staff Control

Staff staff = Database.getStaff(Index.CURENT); - Gets the Staff data by its INDEX

```
ссылка: 1
private void initialize()
{
    /*
     * WRITE CODE BELLOW THE COMMENTS AND DO NOT DELETE THE COMMENTS
     *
     * TODO bind all your data to these variable to display
     * Variables
     * -----------------
     *balanceInfo.Text
     * extraHoursInfo.Text
     * unpaidHoursInfo.Text
     */
    Staff staff = Database.getStaff(Index.CURENT);
    balanceInfo.Text = staff.balance.ToString();
    customerId.Text = staff.staff_id;
    unpaidHoursInfo.Text = staff.hours.ToString();
    extraHoursInfo.Text = staff.extra_hours.ToString();
}
```

Make a payment button :

```
ылка: 1
rivate void button3_Click(object sender, EventArgs e)

   //TODO Pay all the staff balance
   //NOTE Remember to reflesh the user info data by putting new values
   //variable
   /*-----------------
    * balanceInfo.Text
    * extraHoursInfo.Text
    * unpaidHoursInfo.Text
    */
   Staff staff = Database.getStaff(Index.CURENT);
   decimal payment = (staff.balance + (int)staff.role * (staff.hours + 1.4m * staff.extra_hours))
   staff.balance = 0;
   balanceInfo.Text = staff.balance.ToString();
   //Paying calculation ^^^^^^^
   staff.hours = 0;
   staff.extra_hours = 0;
   unpaidHoursInfo.Text = staff.hours.ToString();
   extraHoursInfo.Text= staff.extra_hours.ToString();
   MessageBox.Show(staff.balance.ToString());
   Database.updateStaff(staff);
```

It calculates the hours and extra hours of staff based by their roles. And adds the extra + average hour rates.

`(int)staff.role – cast that links the staff by its role`

Roles

Each role has its own per hour salary

```
namespace FourtBitsBank_0
{
    Ссылок: 9
    public enum Role
    {
        CASHIER = 30,
        ACCOUNTANT = 80,
        MANAGER = 160,
        DIRECTOR = 200
    }
}
```

Public Methods

`public static string nextCustomerId() – creates the Random customer ID`

`public static string nextStaffId() – creates the Random staff ID`

`public static string generateAccountNumber() – creates a new Account Number`