The Bank project

Your project will count as 70 % of the note I have to give you, with 20% on your mini project, and 10% on the class participation.

Your project should be sent to me on teams before November 7 midnight. Every project send to me after this date will lose 25/70 of its final grade.

As described in the class, this project will make use of all the things you have learned till now and more.

The objective of this project is to make a software that can be used to manage clients in a bank first, and then for the bank clients to manage their accounts.

Your software user will have one of 2 types: Admin or Client

An admin have the following functionalities:

- Create a client
- Manage a client (unblock, block, change pin, reset tries, delete client….)
- Verify user transactions(optional)
- View a list of all users

A client have the following functionalities:

- View GUID and credentials (all except pin!)
- View total amount in preferred currency
- Retrieve money from currency
- Add money to currency
- Change pin
- Exchange between currencies
- Transfer between client (optional)
- Leave message for admin(optional)

An admin will only have a hard coded username and password.

You will create a class for admin and in the class, you'll have 2 parameters with values that will represent the admin username and admin password.

As for the client, he will have the following:

- A GUID (use guid library in c#), GUID is a global unique id
- A first name and last name, not unique
- A pin
- List of currencies and amount in each currency (use list)
- Main currency
- Any other important fields for the smooth run of the software

You will use **json** to store all the client you have, as well as a database, this database will be **SQLite**

When your application start, it will search for a data source, on SQLite, a database and will read and create a list of the clients you already have. Json will be used in 2 cases, either if a connection to sqlite is impossible (mandatory), or to verify is all users are synced between sqlite and json(optional).

For the interaction of your code with SQLite you can use entity framework (optional) this will make your project a lot easier and you wont write any sql query.

Finally, for your last technologies to use, you must get the currency exchange prices for a public API, so if the user wants to exchange amounts form a currency to another, he will get the latest prices available in the market. To get those prices use one of those public currency converter API or any other api of your choice:

- [Free Currency Conversion and Forex Exchange Rate API (currencyfreaks.com)](currencyfreaks.com) (free for 1000 calls but have a great documentation)
- [Free Currency Converter API | Free Currency Converter API](#) (free and good documentation)
- [C# Exchange Rate API | C# Currency Converter (exchangerate-api.com)](exchangerate-api.com) (free for 1.5k calls and great complex documentation for c#)

For your software, you will choose a software architecture and work base on it.

You will use Object Oriented Programming, as well as creating a lot of classes.

You must use nterfaces to make this easier on yourself.

As an example:

You will create an interface called IClientDataAccess that will have all the empty functions for the data access, and you will then 2 classes will inherit in, ClientDBAccess(responsible of the db access), and ClientJsonAccess(responsible for the json access).

Those 2 classes will only write the interface function in their own way.

```
┌─────────────────────────────┐
│      IClientDataAccess      │
│                             │
│          GetAll()           │
│                             │
│         CreateUser()        │
│                             │
│      GetClient (int guid)   │
│                             │
│     UpdateClient(Client c)  │
│                             │
│      DeleteClient(int guid) │
└─────────────────────────────┘
       ╱                    ╲
┌──────────────────────┐   ┌──────────────────────┐
│    ClientDBAccess    │   │   ClientJsonAccess   │
│                      │   │                      │
│   GetAll(){//body}   │   │   GetAll(){//body }  │
│                      │   │                      │
│ CreateUser(){//body} │   │ CreateUser(){//body} │
│                      │   │                      │
│ GetClient (int guid){}│  │ GetClient (int guid){}│
│                      │   │                      │
│UpdateClient(Client c){}│ │UpdateClient(Client c){}│
│                      │   │                      │
│DeleteClient(int guid){}│ │DeleteClient(int guid){}│
└──────────────────────┘   └──────────────────────┘
```
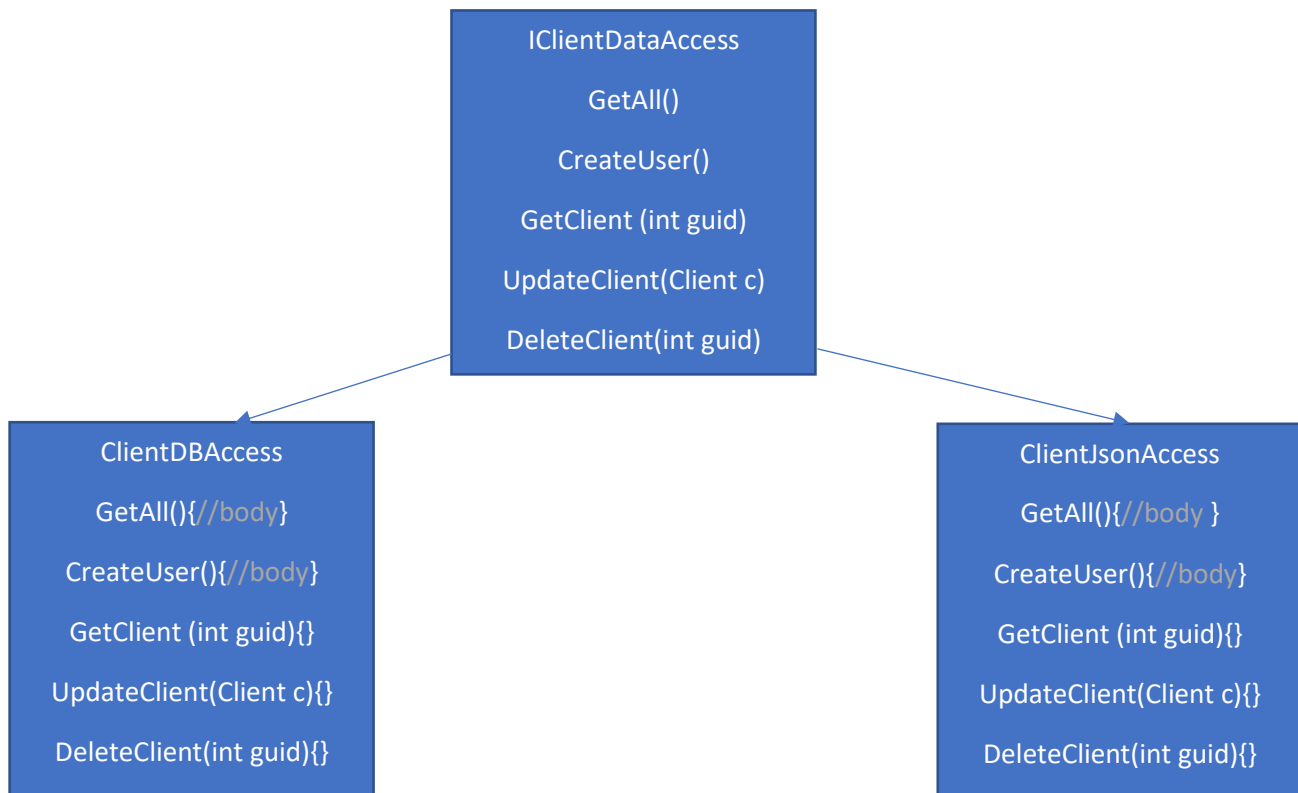
In the previous drawing, you can see that in the interface the functions are empty, but we redefined them in the child class with bodies.

In addition, all the functionalities we need to do to the user can be done using 4 -5 functions, named CRUD operations, Create, Read, Update, Delete.
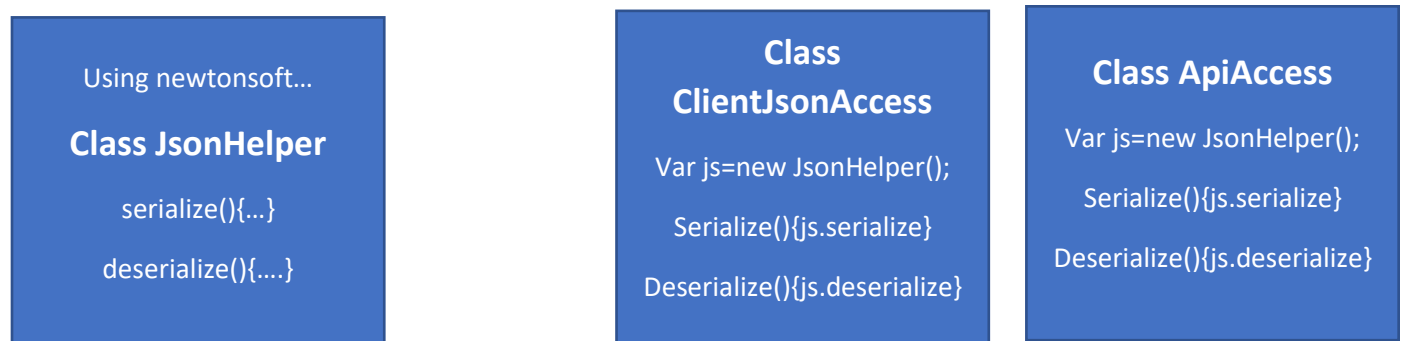
The updateClient function can be used to any changes made to the client or to any field of the client being changed.

Moreover, use a helper class. What's a helper class?

It's a class where we put functions that are common to a problem and not an object.

This means, a JsonHelper, is a class where we define json function that don't depend on database access of api calls or any other object.

Exemple:

```
┌──────────────────────┐   ┌──────────────────────┐   ┌──────────────────────┐
│ Using newtonsoft...  │   │       Class          │   │   Class ApiAccess    │
│                      │   │   ClientJsonAccess   │   │                      │
│  Class JsonHelper    │   │                      │   │ Var js=new JsonHelper();│
│                      │   │Var js=new JsonHelper();│  │                      │
│   serialize(){...}   │   │                      │   │Serialize(){js.serialize}│
│                      │   │Serialize(){js.serialize}│ │                      │
│  deserialize(){....} │   │                      │   │Deserialize(){js.deserialize}│
│                      │   │Deserialize(){js.deserialize}│ │                  │
└──────────────────────┘   └──────────────────────┘   └──────────────────────┘
```

3

This work should be done by up to 3 persons.

<mark>When you finish your project, you will have to send me your code, with a pdf that explain the most important things about it. The pdf must have the name of the persons that did the project.</mark>

More on the PDF you should send It should contain:

- Your names and a small intro of your work.
- A list of all the technologies used.
- Your software architecture.
- Parts of your code with explanation
- Everything that important in your code, add it with explanation.
- Everything you feel proud of being able to code it, add it to your pdf.
- If you did the optional question, add them to your pdf, with a small explanation of how to achieve them.
- You'll also specify each person belongs to which group.

Finally, and again all your project should be sent to me on teams before November 7 midnight. Every project send to me after this date will lose 25/70 of its final grade.

Remember all the optional things in the projects, if attempted and done correctly, will give extra 10 points.

Goodluck everyone.

Mikael Chaaya.