

This project is programmed by Christopher Hardamek.

How to use:

- To start the Blazorapp go to the terminal and navigate to blazorbank. After this run this command "dotnet watch"
- It opens a window of an Internet browser.
- There is some information.
- Please fill out the information.
- After you are done press "Create Account" and then "Save Account"
- The Accounts will be saved in the main Directory in "accounts.json"

Structure:

- Bank_lib: There is all code inside what is necessary to create an account
- Bank.test: There is all test inside
- Blazorbank: There is all stuff inside what the program needs to run on Blazor
- Screenshots: In this folder are all screenshots what I took

Here some Screenshots

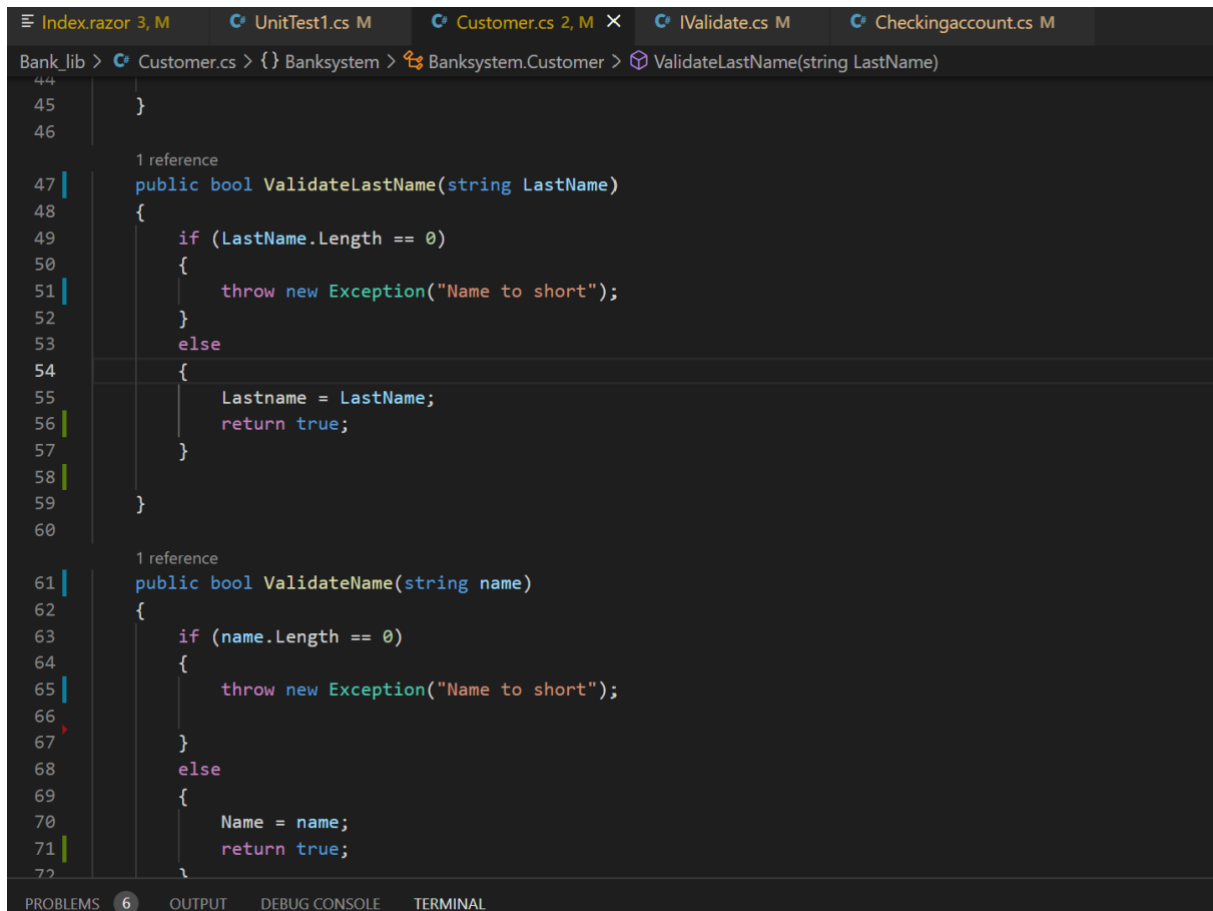
#1 Exceptions

```
}

1 reference
public bool ValidateLastName(string LastName)
{
    if (LastName.Length == 0)
    {
        throw new Exception("Name to short");
    }
    else
    {
        Lastname = LastName;
        return true;
    }
}

1 reference
public bool ValidateName(string name)
{
    if (name.Length == 0)
    {
        throw new Exception("Name to short");
    }
    else
    {
        Name = name;
        return true;
    }
}
```

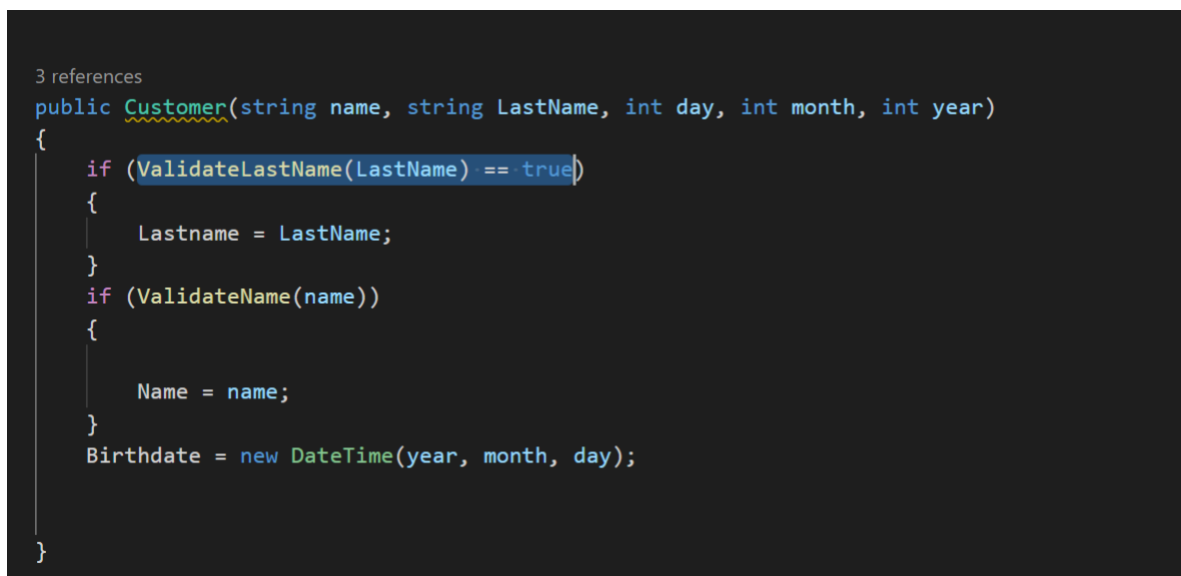
#IValidate



The screenshot shows the Visual Studio IDE with the 'Customer.cs' file open. The file is part of the 'Banksystem' project, under the 'Banksystem.Customer' namespace. The code implements the 'IValidate' interface, which has two methods: 'ValidateLastName(string LastName)' and 'ValidateName(string name)'. Both methods check if the input string is empty. If it is, they throw an 'Exception' with the message 'Name to short'. If it is not empty, they assign the input to a class property ('Lastname' or 'Name') and return 'true'. The code is as follows:

```
44  
45 }  
46  
47 1 reference  
48 public bool ValidateLastName(string LastName)  
49 {  
50     if (LastName.Length == 0)  
51     {  
52         throw new Exception("Name to short");  
53     }  
54     else  
55     {  
56         Lastname = LastName;  
57         return true;  
58     }  
59 }  
60  
61 1 reference  
62 public bool ValidateName(string name)  
63 {  
64     if (name.Length == 0)  
65     {  
66         throw new Exception("Name to short");  
67     }  
68     else  
69     {  
70         Name = name;  
71         return true;  
72 }
```

#3 Polymorphism



The screenshot shows the 'Customer' class constructor in the 'Customer.cs' file. The constructor takes five parameters: 'name', 'LastName', 'day', 'month', and 'year'. It uses the 'ValidateLastName' and 'ValidateName' methods to validate the input strings before assigning them to the class properties. The code is as follows:

```
3 references  
public Customer(string name, string LastName, int day, int month, int year)  
{  
    if (ValidateLastName(LastName) == true)  
    {  
        Lastname = LastName;  
    }  
    if (ValidateName(name))  
    {  
        Name = name;  
    }  
    Birthdate = new DateTime(year, month, day);  
}
```

Here is the commit→

https://github.com/christopherhardamek/Banksystem_final/commit/12cabfea3772b6d47598bf16bf21c75e58faa47c