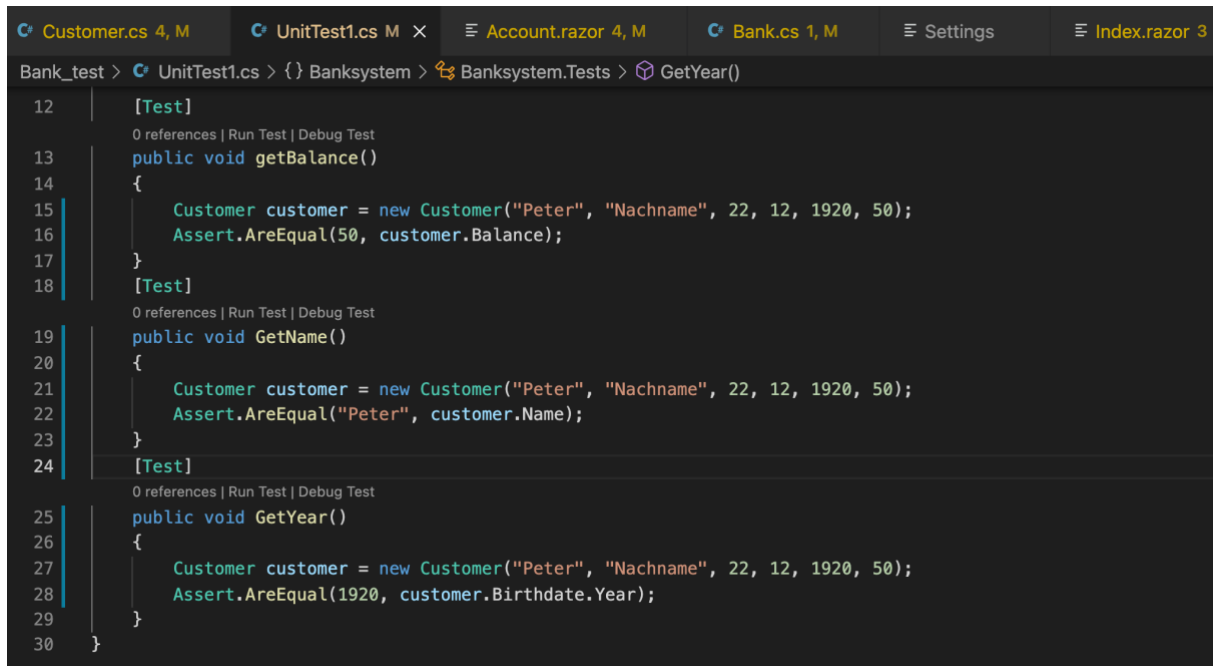


Checkin 2:

Screenshot 1 shows the Unit Tests



```
Bank_test > Customer.cs 4, M | UnitTest1.cs M X | Account.razor 4, M | Bank.cs 1, M | Settings | Index.razor 3
Bank_test > UnitTest1.cs > {} Banksystem > Banksystem.Tests > GetYear()
12 [Test]
13 0 references | Run Test | Debug Test
14 public void getBalance()
15 {
16     Customer customer = new Customer("Peter", "Nachname", 22, 12, 1920, 50);
17     Assert.AreEqual(50, customer.Balance);
18 }
19 [Test]
20 0 references | Run Test | Debug Test
21 public void GetName()
22 {
23     Customer customer = new Customer("Peter", "Nachname", 22, 12, 1920, 50);
24     Assert.AreEqual("Peter", customer.Name);
25 }
26 [Test]
27 0 references | Run Test | Debug Test
28 public void GetYear()
29 {
30     Customer customer = new Customer("Peter", "Nachname", 22, 12, 1920, 50);
31     Assert.AreEqual(1920, customer.Birthdate.Year);
32 }
```

In this Section is the Driver Driven Test. This test allowed them to test their own code.

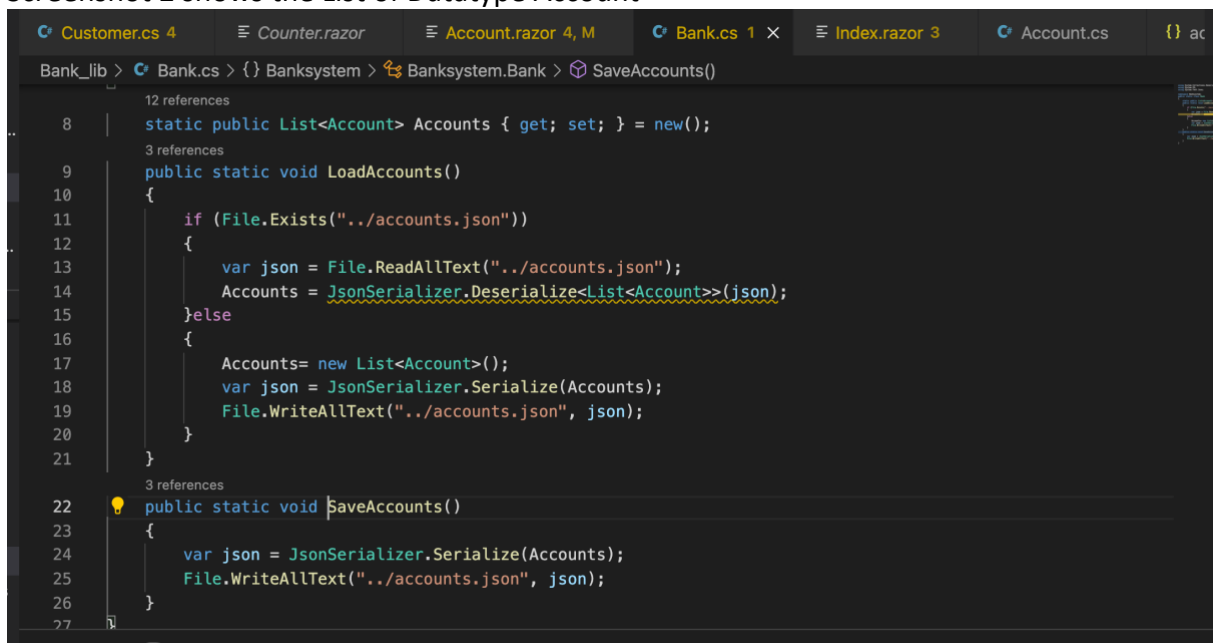
In the first Test, I will get the Balance. In this case, is the balance \$50, and line 16 it's an AreEqual which means the same value except? The test is passed.

In the second public void GetName() I test the pass of value to the variable. I coded that the name is "Peter" in line 22 it gets tested and it's successful.

The third Test is in which year the customer is born. The year of the customer is 1920. In line 28, it will get a test is that true. Yes, the customer is born in 1920.

All three tests passed.

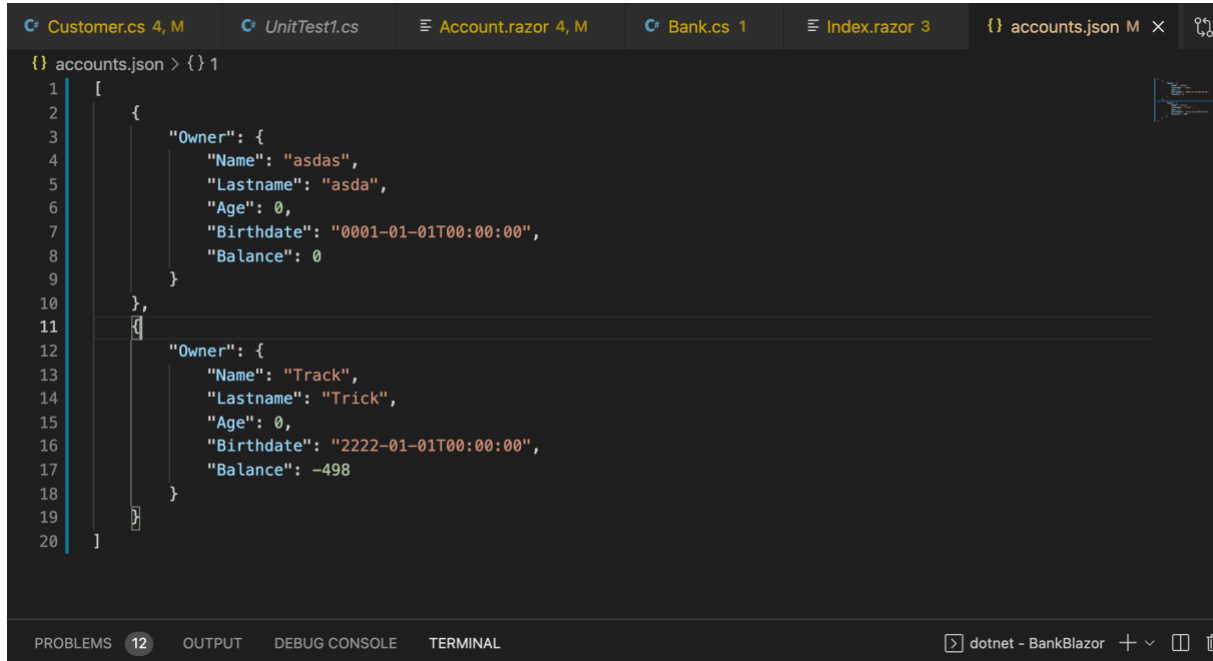
Screenshot 2 shows the List of Datatype Account



```
Bank_lib > Customer.cs 4 | Counter.razor | Account.razor 4, M | Bank.cs 1 X | Index.razor 3 | Account.cs | {} ac
Bank_lib > Bank.cs > {} Banksystem > Banksystem.Bank > SaveAccounts()
12 references
8 static public List<Account> Accounts { get; set; } = new();
3 references
9 public static void LoadAccounts()
10 {
11     if (File.Exists("../accounts.json"))
12     {
13         var json = File.ReadAllText("../accounts.json");
14         Accounts = JsonSerializer.Deserialize<List<Account>>(json);
15     }else
16     {
17         Accounts= new List<Account>();
18         var json = JsonSerializer.Serialize(Accounts);
19         File.WriteAllText("../accounts.json", json);
20     }
21 }
3 references
22 public static void SaveAccounts()
23 {
24     var json = JsonSerializer.Serialize(Accounts);
25     File.WriteAllText("../accounts.json", json);
26 }
27 }
```

In both methods, I use the List. The method load Account is responsible to read the accounts.json and initializing it on the page Index.razor and Accounts.razor. The SaveAccounts() is responsible for saving all value that it gets. The List<Account> Accounts is a list of the data Account. In the list are all entries for the Account. The LoadAccount() reads the file and saves it again in the List<Account> Accounts until the program gets to stop. Without this, wouldn't work the Program, because the memory gets deleted after each stop.

### Screenshot 3



```
{
  "accounts.json": [
    {
      "Owner": {
        "Name": "asdas",
        "Lastname": "asda",
        "Age": 0,
        "Birthdate": "0001-01-01T00:00:00",
        "Balance": 0
      }
    },
    {
      "Owner": {
        "Name": "Track",
        "Lastname": "Trick",
        "Age": 0,
        "Birthdate": "2222-01-01T00:00:00",
        "Balance": -498
      }
    }
  ]
}
```

In this Screenshot, you see 2 Different Accounts. One Account called "Track" and one Account "asdas". The Account has a Balance of 0 Dollars, and the Last name is "asda"

The birthdate is January 1 in the Year 2001.

The second Account is called "Track" with the last name "Trick" it has an account balance of \$-498 the maximum balance is \$-500. The guy is born on January 1<sup>st</sup>, 2022. The function age is not implanted yet. Will be soon. The save path is direct in the root folder of this project.

### Instructions:

On the Index.razor site, you can create the Customer. The customer gets created and save automatically. On this page, there is a list of all Accounts which exists in the file "accounts.json". The accounts are highlighted mean you can click on the name. After clicking you will forward to another page.

After clicking you came to the Account Information page. On this page, you can see all the Account Information about this account, which did you clicked. On this page, you have 2 options to do. The first option is you can Withdraw until \$-500, or you can Pay in Funds, how much money do you have.

I think there is a logical error function in the Withdrawn money because it isn't get showed the right balance.

My next task is to create to transfer the money and log all activities. For withdrawn and pay in funds.