

NOUF ALMEKAWED

Part 1: Preparing a Smart Contract for our dApp

The screenshot displays the Remix Ethereum IDE interface. The top bar shows the browser address: `remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js`. The left sidebar contains the 'DEPLOY & RUN TRANSACTIONS' panel, which includes a 'Deployed Contracts' section showing a contract named 'BANK AT 0x78F...E3CAF (BLOCKCHAIN)'. Below this, there are buttons for 'addaccount', 'transfer', 'Withdraw', 'bank_address', 'bank_name', 'getBalance', and 'number_of_ac...'. The main editor area shows the Solidity code for the 'Bank' contract. The code includes a pragma statement for Solidity version, a contract definition with public variables for bank name and address, a uint variable for the number of accounts, a mapping for balances, a mapping for KYC records, a struct for KYC data, a modifier for registered users, and a public function for adding accounts. The bottom status bar shows a successful transaction: '[block: txIndex:] from: 0xf6E...5731F to: Bank.(constructor) value: 0 wei data: 0x608...70033 logs: 0 hash:'. A 'Debug' button is visible in the bottom right corner.

```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity >=0.7.0 <0.9.0;
4
5 contract Bank {
6     string public bank_name;
7     string public bank_address;
8
9
10    uint public number_of_accounts = 0;
11
12    mapping (address =>uint) balances;
13
14    mapping (address => KYC) record;
15
16    struct KYC {
17        uint customer_ID;
18        string full_name;
19        string profession;
20        string Date_of_Birth;
21        address Wallet;
22    }
23
24    modifier onlyRegistered(address cAddress) {
25        require(record[cAddress].customer_ID > 0, "Not Registered");
26        _;
27    }
28
29    function addaccount(string memory full_name, string memory profession, string memory Date_of_Birth) public {
30        require(record[msg.sender].customer_ID == 0, "Available");
31    }
32}
```