

Withdrawal 5 into account 1

The screenshot displays the Remix IDE interface, which is used for developing and interacting with Ethereum smart contracts. The interface is divided into several panels:

- Left Panel (Deploy & Run Transactions):** This panel contains a sidebar with icons for environment, injected web3, and deployed contracts. The main area shows the 'Deployed Contracts' section with a list of contracts. The 'JointSavings' contract is selected, and its 'deposit' and 'withdraw' functions are visible. The 'withdraw' function is currently selected, and its parameters are set to 'amount: 5' and 'recipient: 0x00'. The 'contractBalance' and 'lastWithdraw' functions are also visible.
- Right Panel (Contract Definition):** This panel shows the Solidity code for the 'JointSavings' contract. The code is as follows:

```
contract JointSavings {  
    /*  
    Inside the new contract define the following variables:  
    - Two variables of type 'address payable' named 'accountOne' and 'accountTwo'  
    - A variable of type 'address public' named 'lastWithdraw'  
    - Two variables of type 'uint public' named 'lastWithdrawAmount' and 'contractBalance'.  
    */  
    address payable accountOne;  
    address payable accountTwo;  
    address public lastWithdraw;  
    uint public lastWithdrawAmount;  
    uint public contractBalance;  
    /*  
    Define a function named 'withdraw' that will accept two arguments:  
    - A 'uint' variable named 'amount'  
    - A 'payable address' named 'recipient'  
    */  
    function withdraw(uint amount, address payable recipient) public {  
        /*  
        Define a 'require' statement that checks if the 'recipient' is equal to either 'accountOne' or 'accountTwo'. The  
        */  
        require(recipient == accountOne || recipient == accountTwo, "You do not own this account");  
        /*  
        Define a 'require' statement that checks if the 'balance' is sufficient to accomplish the withdraw operation. If  
        */  
        require(contractBalance >= amount, "Insufficient funds");  
        /*  
        Add an 'if' statement to check if the 'lastWithdraw' is not equal to 0. If not, set 'recipient' to 'lastWithdraw'.  
        */  
        if(lastWithdraw != 0){  
            lastWithdraw = recipient;  
        }  
    }  
}
```
- Bottom Panel (Transaction Log):** This panel shows the transaction log for the 'JointSavings' contract. It displays a successful transaction for the 'withdraw' function, with the following details: 'From: 0x00', 'To: 0x00', 'Value: 5 wei', 'Data: 0x00', 'Gas: 42000', 'Status: Success'.

Withdraw 10 into account 2

The screenshot displays the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is active, showing the 'Deployed Contracts' section with a contract named 'JmintSavings' at address 0x001...39138-64. Below this, the 'setAccounts' function is visible, and the 'withdraw' function is being executed. The 'amount' field is set to 10, and the 'recipient' field is set to 0x7A17747a74a1984860C06...Wam00008129. The 'contractBalance' and 'lastWithdraw' fields are also visible. The 'Low level interactions' section at the bottom shows the 'Call DATA' field.

The main editor displays the Solidity code for the 'JmintSavings' contract:

```
contract JmintSavings {  
    /*  
    Inside the new contract define the following variables:  
    - two variables of type 'address payable' named 'accountOne' and 'accountTwo'  
    - a variable of type 'address public' named 'lastWithdraw'  
    - two variables of type 'uint public' named 'lastWithdrawAmount' and 'contractBalance'.  
    */  
    address payable accountOne;  
    address payable accountTwo;  
    address public lastWithdraw;  
    uint public lastWithdrawAmount;  
    uint public contractBalance;  
    /*  
    Define a function named "withdraw" that will accept two arguments:  
    - a 'uint' variable named 'amount'  
    - a 'payable address' named 'recipient'  
    */  
    function withdraw(uint amount, address payable recipient) public {  
        /*  
        Define a 'require' statement that checks if the 'recipient' is equal to either 'accountOne' or 'accountTwo'. The 'require' statement will throw an error if the condition is not met.  
        */  
        require(recipient == accountOne || recipient == accountTwo, "You do not use this account!");  
        /*  
        Define a 'require' statement that checks if the 'balance' is sufficient to accomplish the withdraw operation. If not, it will throw an error.  
        */  
        require(contractBalance >= amount, "Insufficient funds!");  
        /*  
        Add an 'if' statement to check if the 'lastWithdraw' is not equal to ('/') to 'recipient'. If 'lastWithdraw' is not equal to 'recipient', it will update 'lastWithdraw' to 'recipient' and 'lastWithdrawAmount' to 'amount'.  
        */  
        if(lastWithdraw != recipient){  
            lastWithdraw = recipient;  
            lastWithdrawAmount = amount;  
        }  
        contractBalance -= amount;  
        recipient.transfer(amount);  
    }  
}
```

The bottom status bar shows the transaction details: [tx] from: 0x001...39138-64 to: JmintSavings.withdraw(uint256,address) 0x001...39138-64 value: 0 wei data: 0x00...0120 logs: 0 hash: 0x011...13077. The transaction is confirmed as 'call to mintSavings.contractBalance'.