

JS Advanced - Exam: 08.04.2020

Exam problems for the ["JavaScript Advanced" course @ SoftUni](#).

Problem 3. Bank

```
class Bank {  
    // TODO: implement this class...  
}
```

Your Task

Write a Class Bank, Which Implements the Following Functionality:

Functionality

constructor (bankName)

newCustomer (customer)

depositMoney (personalId, amount)

Both the **personalId** and the **amount** are numbers.

- Check if the given **personalId** corresponds to a customer in the **customers** array, if not **throw a new error**:

"We have no customer with this ID!"

- Otherwise **add the amount** to the corresponding customer in a property named **totalMoney** and **store the transaction information** to this customer (for more clarity see the example below and the hints), then **return the total money** of the corresponding customer and a dollar sign:

"{totalMoney}\$"

withdrawMoney (personalId, amount)

Both the **personalId** and the **amount** are numbers.

- Check if the given **personalId** corresponds to a customer in the **customers** array, if not **throw a new error**:

"We have no customer with this ID!"

- If there is a customer with the given **personalId**, check if the customer **has enough money** in his account, to withdraw the given amount. If the money is not enough **throw a new error**:

"{firstName} {lastName} does not have enough money to withdraw that amount!"

- Otherwise subtract the **amount** from the **totalMoney** of the customer and store the **transaction information** to this customer, then **return the total money** of the corresponding customer and a dollar sign:

"{totalMoney}\$"

customerInfo (personalId)

The **personalId** is of type **number**.

- Check if the given **personalId** corresponds to a customer in the **customers** array, if not **throw a new error**:

“We have no customer with this ID!”

- Otherwise **return the whole information** for the customer in the following format:

“Bank name: {bankName}

Customer name: {firstName} {lastName}

Customer ID: {personalId}

Total Money: {totalMoney}\$

Transactions:

n. {firstName} {lastName} made deposit of {amount}\$!

...

2. {firstName} {lastName} withdrew {amount}\$!

1. {firstName} {lastName} made deposit of {amount}\$!”

Transaction information contains information about:

- **number** of the transaction in descending order;
- **names** (firstName, lastName);
- if the transaction is **deposit/withdraw**;
- **amount** of the transaction.

Examples

This is an example how the code is **intended to be used**:

Sample code usage

```
let bank = new Bank('SoftUni Bank');

console.log(bank.newCustomer({firstName: 'Svetlin', lastName: 'Nakov', personalId: 6233267}));
console.log(bank.newCustomer({firstName: 'Mihaela', lastName: 'Mileva', personalId: 4151596}));

bank.depositMoney(6233267, 250);
console.log(bank.depositMoney(6233267, 250));
bank.depositMoney(4151596, 555);

console.log(bank.withdrawMoney(6233267, 125));

console.log(bank.customerInfo(6233267));
```

Corresponding output

```
{ firstName: 'Svetlin', lastName: 'Nakov', personalId: 6233267 }  
{ firstName: 'Mihaela', lastName: 'Mileva', personalId: 4151596 }  
500$  
375$  
Bank name: SoftUni Bank  
Customer name: Svetlin Nakov  
Customer ID: 6233267  
Total Money: 375$  
Transactions:  
3. Svetlin Nakov withdrew 125$!  
2. Svetlin Nakov made deposit of 250$!  
1. Svetlin Nakov made deposit of 250$!
```