

Classes, Constructors, and Inheritance

An Example with Bank Accounts

2/16/2022

Getting Started:

As starting point for this lab an `account.js` file has been provided containing an `Account` class. Read through the code, making sure you understand what everything does.

An `Account` simply encapsulates its number and balance. Getters are provided for both number and balance, but because they should not be modified directly no setters are created.

There are two methods that can change the balance, `deposit(amount)` which add money into the account, and `withdraw(amount)` that removes money from the account.

Lastly there is a `toString()` method that creates a string representation of an `Account`.

Exercises:

- a) Use the [Mocha test file, bankTests.js](#), to verify that that everything in `accounts.js` works as expected.
- b) Extend the `Account` class by creating a class called `SavingsAccount` in a file called `savingsaccount.js`. In addition to the attributes of `Account`, `SavingsAccount` should have an interest variable, which is set in the constructor and has a getter and a setter method. It should also have an `addInterest()` method which deposits the interest amount into the account. The calculation for the amount is $\text{balance} * \text{interest} / 100$. Be sure to also overwrite the `toString()` method, and test with the Mocha tests in `bankTests.js` for the methods in `SavingsAccount`.
- c) Create a `CheckingAccount` class by extending `Account`. In addition to the attributes of an `Account`, it should have an overdraft limit variable. The overdraft amount indicates how much a person is allowed to temporarily withdraw beyond what they have. In other words, it's the amount that an account is allowed to go into the red (negative balance). Be sure to set this value in the constructor and create a getter and a setter for it. Also make sure that you override the `withdraw(amount)` method and the `toString()` method. Test with `bankTests.js`.
- d) Next create a `Bank` class, a `Bank` object should have an array of `Account` objects, and have `addAccount()`, `addSavingsAccount(interest)`, `addCheckingAccount(overdraft)` methods each of which returns the number of the created account. Also add a `closeAccount(number)` method that closes (removes from the array) the account with that number, and a `accountReport()` method that returns a String

with each account on its own line. Use a static `nextNumber` variable on the `Bank` class to know what the number for the next account will be. Test with `bankTests.js`.

- e) Create an `endOfMonth()` method on the `Bank` class, and on `Account`, `SavingsAccount`, and `CheckingAccount`. The method on the `Bank` class should go through the array calling `endOfMonth()` on each of the accounts collecting their output. For normal `Accounts` the `endOfMonth()` method should return an empty string. For `SavingsAccounts` it should call the `addInterest()` method and return a string specifying how much interest was added to this account (see example below), and for `CheckingAccounts` it should check if the balance is below zero, and if so return a string with a warning (see example below). Test with `bankTests.js`.

Interest added SavingsAccount 2: balance: 102.5 interest: 2.5

Warning, low balance CheckingAccount 3: balance: -100 overdraft limit: 500

passes: 32 failures: 0 duration: 0.05s 100%

Account class

- `constructor(number)`
 - ✓ takes a number which becomes the account number
- `getNumber() method`
 - ✓ returns the account number
- `getBalance() method`
 - ✓ returns the current account balance
- `deposit(amount) method`
 - ✓ adds amount to the current balance
 - ✓ throws a `RangeError` if you give a number ≤ 0
- `withdraw(amount) method`
 - ✓ removes amount from the current balance
 - ✓ throws a `RangeError` if you give a number ≤ 0
 - ✓ throws an `Error` if you try to withdraw money you don't have
- `toString() method`
 - ✓ returns a string representation of the account
- `endOfMonth() method`
 - ✓ returns an empty string

SavingsAccount

- `constructor(number, interest)`
 - ✓ takes a number and an interest rate and makes a `SavingsAccount`
- `interest getter / setter`
 - ✓ can get the interest rate for this account
 - ✓ can set the interest rate for this account
- `addInterest() method`
 - ✓ adds the calculated interest to this account
- `toString() method`
 - ✓ returns a string representation of the `SavingsAccount`
- `endOfMonth() method`
 - ✓ returns a string saying that interest was added

CheckingAccount

- `constructor(number, overdraft)`
 - ✓ takes a number and the overdraft limit and makes a checking account
- `overdraft getter / setter`
 - ✓ can get the overdraft limit for this account
 - ✓ can set the overdraft limit for this account
- `withdraw(amount) method`
 - ✓ can withdraw into negative up to the overdraft limit
 - ✓ throws an error if you go beyond the limit