

Specifikation till Fiktiv P-uppgift: BANKKONTON

Beskrivning

- Programmet ska hantera bankkonton.
- Användaren ska kunna öppna ett konto, sätta in och ta ut pengar, byta PIN-kod samt se de senaste tio transaktionerna.
- All kontodata ska sparas på fil mellan körningar.

Algoritm

1. Programmet läser in alla konton från fil
2. Upprepa följande tills användaren vill sluta:
 - a. Visa menyn
 - b. Läs in meny-val från användaren
 - c. Läs in information (t.ex. belopp eller pinkod)
 - d. Anropa vald metod med rätt parametrar
3. När användaren väljer att avsluta ska de ändrade kontoupppgifterna (hela dictionaryn) skrivas ut på filen igen.

Datastrukturer

Varje bankkonto representeras av ett Account-objekt. Alla bankkonton lagras i en dictionary med namnet som nyckel. Mellan programkörningarna lagras alla data på en textfil.

Klasser och metoder

```
class Account:
    """
    Attributes:
    name: The name of the person
    money: The amount of money on the account
    pin_code: The pin code needed to access the account
    transactions: a list of the latest 10 transactions
    """
```

```

def __init__(self, name, money, pin_code):
    """
    Creates a new account
    :param name: The name of the person
    :param money: The amount of money on the account
    :param pin_code: The pin code needed to access the account
    """

def __str__(self):
    """
    Account information for printouts
    :return: A nice string including name, money and transaction
    history of the account
    """

def deposit(self, amount):
    """
    Used to add money to the account, also updates the
    transaction history
    :param amount: the amount of money
    :return: a string confirming the successful deposit
    """

def withdrawal(self, amount, pin):
    """
    Used to withdraw (remove) money from the account, also
    updates the transaction history.
    :param amount: the amount of money
    :param pin: the pin needed to access the account
    :return: a string confirming the successful withdrawal, or
    notifying the user of wrong pin, or lack of money
    """

def ok_PIN(self, pin):
    """
    Used to check if pin is correct
    :param pin: The pin code needed to access the account
    :return: Boolean variable, True if new_pin is correct, False
    otherwise
    """

def change_PIN(self, old_pin, new_pin):
    """
    Used to change pin-code
    :param old_pin: the old pin code
    :param new_pin: the desired new pin code
    :return: a string confirming the successful change, or
    notifying that the pin was wrong
    """

```

Funktioner

```
def read_accounts_from_file(file_name):
    """
    Used to get data from file
    :param file_name: The file with the information
    :return: A dictionary with all accounts on it
    """

def write_accounts_to_file(accounts, file_name):
    """
    Used to write the data to a file
    :param accounts: The dictionary with the accounts
    :param file_name: The file to be created
    :return: (nothing)
    """

def get_int_input(prompt_string):
    """
    Used to get an int from the user, asks again if input is
    not convertible to int
    :param prompt_string: the string explaining what to input
    :return: the int that was asked for
    """

def display_accounts(account_dict):
    """
    Used to display the information on all accounts (debugging
    only, not for bank customers)
    :param account_dict: a dictionary with the accounts
    :return: (nothing)
    """

def menu():
    """
    Used to display the menu:
    What would you like to do?
    1 - Set up a new account
    2 - Deposit
    3 - Withdrawal
    4 - Change pin
    5 - Display earlier transactions
    6 - Exit
    :return: (nothing)
    """
```

```
def menu_choice():  
    """  
    Used to get input on what the user wants to do  
    :return: an int, the chosen menu option  
    """  
  
def execute(choice):  
    """  
    Used to execute the option that the user chose  
    :param choice: an int corresponding the the chosen option  
    :return: (nothing)  
    """
```