NBI Bank Application

NBI Bank Application (Console)

RESTORY...

Inledning

I den här uppgiften ska följande levereras:

- Github Repo
- Fungerande Applikation (mer info i nästa slide)
- Mockdata

Syftet med projektet är att tillämpa kunskaper från objektorienterad programmering och Python programmeringsspråket

RESTORY...

Fungerande Applikation:

Vad ska ni göra

- Projektuppgiften går ut på att ta fram ett enkelt system för en fiktiv bank. Banken har ett antal kunder och varje kund kan ha ett eller flera olika bankkonton

Krav och Specifikationer:

- Banken erbjuder för tillfället endast konton av typen debitkonto (**Account**)
- Börja med att skapa en klass som definierar konto (**Account**), en klass som definierar kund (**Customer**) samt en bank klass (**Bank**).
- Alla klasser ska fungera som moduler som importeras in i main ***script***:et

RESTORY...

Specifikation

I Banken ska man kunna:

- *Printa en lista med bankens kunder (personnummer, för och efternamn)*
- *Lägga till en ny kund med ett unikt personnummer.*
- *Ändra en kunds namn (personnummer ska inte kunna ändras).*
- *Ta bort en befintlig kund, befintliga konton måste också avslutas.*
- *Skapa konto (Account) till en befintlig kund, ett unikt kontonummer genereras (**VG**)(första kontot får nummer 1001, nästa 1002 osv.).*
- *Konton ska även kunna avslutas via kontonummer attributet, saldo skrivs ut och kontot tas bort.*
- (**VG**) *Se alla transaktioner en kund har gjort med ett specifikt konto.*

RESTORY...

Specifikation

För varje kund ska man kunna utföra följande:

- *Se information om vald kund inklusive alla konton (kontonummer, saldo, kontotyp).*
- *Sätta in pengar på ett konto.*
- *Ta ut pengar från kontot (men bara om saldot täcker uttagsbeloppet)*.

RESTORY...

Klassdesign - Account

Börja med att implementera klassen **Account** som ska hantera följande information:

- Saldo
- Kontotyp (<class ‘str’>)
- Kontonummer (det kan inte finnas flera konton med samma kontonummer).

Man ska kunna utföra *transaktioner* (insättning/uttag), hämta *kontonummer*, samt *presentera* kontot (visa kontonummer, saldo, kontotyp).
Implementera metoder som säkerställer ovanstående krav i klassen Bank.
(Bank ***klassdesign*** inkluderar metoderna som ska användas. Komplettera dessa med en implementation eller fler metoder om det behövs.)

RESTORY...

Klassdesign - Customer

Klassen Customer ska hantera följande information:

- Id
- Kundens namn
- Personnummer
- Kundens alla konton
- (**VG**) Transaktioner

Man ska till exempel kunna ändra kundens namn samt hämta information om kunden (personnummer, för- och efternamn samt hämta information om kundens konton (kontonummer, saldo, kontotyp)).

Dessutom ska man kunna hantera kundens konto(n). Implementera metoder som säkerställer ovanstående krav i klassen Bank nedan. (Bank inkluderar förslag på metoder. Komplettera dessa med fler metoder om det behövs).

RESTORY...

Klassdesign - Bank

Bank
Klassen Bank ska innehålla en lista med alla kunder.
Klassen ska innehålla ett antal metoder som hanterar kunder och dess konton.

OBS! För **G** använd följande metod för att ladda all info från textfilen
(*inkluderas. bild finns längre ner*).

def _load():
- Läser in text filen och befolkar listan som ska innehålla kunderna.

def get_customers():
- Returnerar bankens alla kunder (***personnummer*** och ***namn***)

def add_customer(name, pnr):
- Skapar en ny kund med namn och personnummer. Kunden skapas endast om det inte finns någon kund med personnumret som angetts. Returnerar **True** om kunden skapades annars returneras **False**.

Klassdesign - Bank

def get_customer(pnr):

- Returnerar information om kunden inklusive dennes konton. Första platsen i listan är förslagsvis reserverad för kundens namn och personnummer sedan följer informationen om kundens konton.

def change_customer_name(name, pnr)

- Byter namn på kund, returnerar **True** om namnet ändrades annars returnerar det **False** (om kunden inte fanns).

def remove_customer(pnr)

- Tar bort kund med personnumret som angetts ur banken, alla kundens eventuella konton tas också bort och resultatet returneras. Listan som returneras ska innehålla information om alla konton som togs bort, saldot som kunden får tillbaka.

def add_account(pnr)

- Skapar ett konto till kunden med personnumret som angetts, returnerar kontonumret som det skapade kontot fick alternativt returneras –1 om inget konto skapades.

RESTORY...

Klassdesign - Bank

def get_account(pnr, account_id)

- Returnerar Textuell presentation av kontot med kontonummer som tillhör kunden (kontonummer, saldo, kontotyp).

def deposit(pnr, account_id, amount)

- Gör en insättning på kontot, returnerar **True** om det gick bra annars **False**.

def withdraw(pnr, account_id, amount)

- Gör ett uttag på kontot, returnerar **True** om det gick bra annars **False.**

def close_account(pnr, account_id)

- Avslutar ett konto. Textuell presentation av kontots saldo ska genereras och returneras.

(VG) def get_all_transactions_by_pnr_acc_nr(pnr, acc_nr):

- Returnerar alla transaktioner som en kund har gjort med ett specifikt konto eller -1 om kontot inte existerar.

RESTORY...

(VG) Klassdesign - DataSource

DataSource (base class)
Klassen DataSource ska innehålla metoder som hanterar var datan kommer från, t.ex. *Textfil, Json fil, Databas, API.*
DataSource klassen kräver konkreta implementationer. Ett krav är att implementationen ska använda en ***textfil*** som datasource.

def datasource_conn():

- Denna metod implementerar kopplingen till en generisk datasource. Returnerar en <class ‘tuple’> med en <class ‘bool’> och en <class ‘str’> t.ex., (True, “*Connection successful”* [, *datasource namn*])

def get_all():

- Returnerar alla kunder i banken.

RESTORY...

(VG) Klassdesign - DataSource

def update_by _id(id):

- Uppdaterar en kund baserad på id:n som angetts som parameter. Returnerar info om kunden som uppdaterats, eller -1 om kunden inte finns.

def find_by_id(id):

- Returnerar en kund baserad på id:n som angetts eller -1 om kunden in finns.

def remove_by_id(id):

- Raderar en kund baserad på id:n som angetts som parameter. Returnerar info om kunden som tagits bort eller -1 om kunden inte finns.

RESTORY...

(VG) Klassdesign - Transaktion

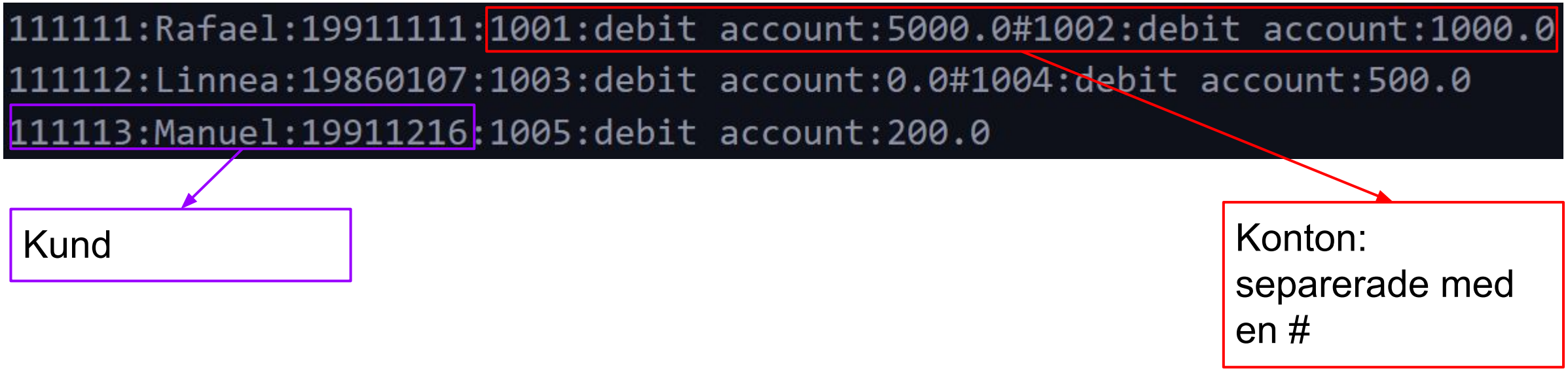
Transaction
Klassen Transaction kommer att hantera följande information:

- Id
- Kundens Id
- Konto Id
- Datum
- Belopp

Belopp attributet kan ha negativa eller positiva nummer, t.ex., om kunden har tagit ut 2000 kr visar *belopp* attributet -2000. Att sätta in 300 kr ger attributet ett värde av +300 eller 300.

RESTORY...

En bild av textfilen (mockdata)



Dokumentation

1. Skapa en grundläggande struktur för dokumentet som ska lämnas in (individuellt)
2. **Inledningen** ska ha en beskrivning av arbetet som ska göras:
   a. Inkluderar en beskrivning av projektet och målet.
   b. User Stories *(user stories är en beskrivning av vad användaren ska kunna göra i applikationen. Titta i "specs" slide)*
   c. Teknologierna som ska användas. (Python [, -***moduler***-])
3. Inkludera **klassdiagram** av följande klasser;
   a. Bank, Customer, Account
4. **Milestones** och **Tasks** ska dokumenteras, inkluderar:
   a. Beskrivning av *milestones*.
   b. Tasks för varje *milestone*.

RESTORY...

Redovisning (27/1 & 1/2) & Inlämning (27/1)

1. **Redovisning**:
    - Studenten ska redovisa och diskutera sin kod.

2. Inlämning: Producera ett dokument med följande information:
    - Länk till GitHub-repository
    - Dokumentation (*google.docs*)

Bedömning

1. Denna uppgift kan ge betyg G & VG
    - För **G** måste samtliga krav i “specifikationerna” slide ha uppnåtts.
    - För **VG** måste studenten diskutera och visa god förståelse för koden i applikationen, lämna in dokumentation och ha tillämpat alla **VG** markerade *features.*

RESTORY...