Grit Bank Application

Grit 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 Java 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**)
- Man ska börja med att skapa en klass som definierar konto (**Account**), en klass som definierar kund (**Customer**) samt en klass (**Bank**).

RESTORY...

I banken ska man kunna:

Specs:

I Banken ska man kunna:

- *(**VG**)Skriva ut en lista med bankens kunder (personnummer, för och efternamn) till en textfil*
- *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, saldo skrivs ut och kontot tas bort*

I banken ska man kunna:

Specs:

För en viss 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)*

Klassdesign

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

- Saldo
- Kontotyp (**konto (Account)**)
- Kontonummer (det kan inte finnas flera konton med samma kontonummer).

Man ska kunna utföra transaktioner (insättning/uttag), hämta kontonummer, samt presentera kontot (kontonummer, saldo, kontotyp). Implementera metoder som säkerställer ovanstående krav i klassen Bank nedan (Bank inkluderar förslag på metoder. Komplettera dessa med fler metoder om det behövs).

Klassdesign

Customer
Klassen Customer ska hantera följande information:

- Kundens namn
- Personnummer
- En lista med kundens alla konton

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 konton.
Implementera metoder som säkerställer ovanstående krav i klassen Bank nedan (Bank inkluderar förslag på metoder. Komplettera dessa med fler metoder om det behövs).

Klassdesign

Bank

Klassen Bank ska innehålla en lista med alla inlagda kunder. Klassen ska innehålla ett antal publika metoder som hanterar kunder och dess konton (se ovan). Du kommer troligtvis att skapa ett antal hjälpmetoder, privata metoder, men de publika metoderna som ska finnas i Bank är följande:

public List<String> GetCustomers()

- Returnerar en List<String> som innehåller en presentation av bankens alla kunder (personnummer och namn)

public boolean AddCustomer(String name, long pNr)

- Skapar upp en ny kund med namnet name samt personnummer pNr, kunden skapas endast om det inte finns någon kund med personnummer pNr. Returnerar true om kund skapades annars returneras false.

public List<String> GetCustomer(long pNr)

- Returnerar en List<String> som innehåller informationen om kunden inklusive dennes konton. Första platsen i listan är förslagsvis reserverad för kundens namn och personnummer sedan följer informationen om kundens konton.

Klassdesign

public boolean ChangeCustomerName(String name, long pNr)

- Byter namn på kund med personnummer pNr till name, returnerar true om namnet ändrades annars returnerar false (om kunden inte fanns).

public List<String> RemoveCustomer(long pNr)

- Tar bort kund med personnummer pNr 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 samt vad räntan blev.

public int AddAccount(long pNr)

- Skapar ett konto till kund med personnummer pNr, returnerar kontonumret som det skapade kontot fick alternativt returneras –1 om inget konto skapades.

public String GetAccount(long pNr, int accountId)

- Returnerar en String som innehåller presentation av kontot med kontonummer accountId som tillhör kunden pNr (kontonummer, saldo, kontotyp, räntesats).

public boolean Deposit(long pNr, int accountId, int amount)

- Gör en insättning på konto med kontonummer accountId som tillhör kunden pNr, returnerar true om det gick bra annars false.

public boolean Withdraw(long pNr, int accountId, int amount)

- Gör ett uttag på konto med kontonummer accountId som tillhör kunden pNr, returnerar true om det gick bra annars false

Klassdesign

public String CloseAccount(long pNr, int accountId)

- Stänger ett konto med kontonummer accountId som tillhör kunden pNr, presentation av kontots saldo ska genereras.

Dokumentation.

1. Skapa en grundläggande struktur för dokumentet som ska lämnas in individuellt.
2. **Inledning** 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. (Java)
3. **Milestones** och **Tasks** ska dokumenteras, inkluderar:
   a. Beskrivning av *milestones*.
   b. Tasks för varje *milestone*.

RESTORY...

Redovisning (26/11) & Inlämning (28/11)

1. **Redovisning**:
    - Studenten ska redovisa och diskutera det som gjorts.

2. Inlämning: Producera ett dokument med följande information:

- Länk till GitHub-repository
- Länk till **SCRUM**-board (*Om ni ens vet vad detta är*)
- Dokumentation

Bedömning

1. Denna uppgift kan ge betyg G
    - 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*