NACKADEMIN

Uppgift 3: Bygg en bank!

Den tredje inlämningsuppgiften ska lösas individuellt.

Du ska bygga en del av ett system för en bank som uppfyller kraven på följande sidor.

Fusk

Uppgiften ska lösas individuellt. Vid redovisningen får du frågor och måste beskriva hur ditt program fungerar så du måste kunna förklara hur all din egen kod fungerar.

Det är inte tillåtet att plagiera (skriva av) någon annans kod. Även om du byter namn på klasser, metoder och variabler är det enkelt att se om två olika applikationer har samma struktur.

Fusk rapporteras till Nackademins ledningsgrupp, som beslutar om eventuellt påföljd, som normalt är avstängning från deltagande i undervisning och examination under en bestämd tid.

Betyg

Det går att få G, VG eller IG på uppgiften. Den är obligatorisk tillsammans med inlämningsuppgift 1 och 2 för att kunna få G på hela kursen.

För att få G på inlämningsuppgiften ska alla krav märkta med G vara uppfyllda.

För att få VG på inlämningsuppgiften ska alla krav märkta med G och VG vara uppfyllda.

För att kunna få VG på hela kursen måste du skriva VG på tentan sam ha VG på inlämningsuppgift 3.

Redovisning

Redovisning sker genom att du visar läraren den fungerande uppgiften och kan besvara frågor om koden. Du kommer att få en tid antingen tisdag 9 oktober eller onsdag 10 oktober då du ska redovisa. Skicka INTE in något via mail!

NACKADEMIN

Krav för Godkänt

Bygg ett realtidssystem för en bank som ska hålla objekt för alla kunder och konton i minnet och hålla reda på hur mycket pengar en kund har just nu.

Alla kunder och konton lagras i en textfil. När programmet startas läses en textfil in med data och alla objekt skapas.

Det finns en fil med data på studentportalen som du ska kunna läsa in och filen du skapar ska ha samma format. Tips! Läs på mer om *ToString* och *Parse* som kan ta fler parametrar som styr hur tal och datum formateras och tolkas. Kolla in *InvariantCulture* som är en variant av *CultureInfo* som ofta används för import och export.

När programmet avslutas ska en ny uppdaterad textfil skapas. Filnamnet ska vara aktuellt datum och tid på formatet ”*åååmmdd-ttmm.txt*”, t ex *20171231-2359.txt* där 20171231är dagens datum och 2359 är klockslaget. Tips! Använd *DateTime.Now* och läs mer om *Custom Date and Time Format Strings*.

Skriv ut statistik när systemet läser in eller sparar så det är lätt att se att det fungerar (se exempel).

Det ska gå att ta fram en kundbild genom att ange antingen ett kundnummer eller kontonummer. Kundbilden ska visa all information om kunden och alla kundens konton. Kundbilden ska också visa det totala saldot för kunden genom att summera saldot på kundens konton.

Det ska gå att söka efter kund på namn eller postort. En lista ska visas med kundnummer och namn.

Det ska gå att skapa en ny kund. När en kund skapas ska de få ett nytt unikt kundnummer.
Det ska också automatiskt skapas ett transaktionskonto med kunden som ägare.

En kund måste ha ett kundnummer, namn, organisationsnummer, adress, postnummer och postort samt ett eller flera konton.

Det ska gå att skapa ett nytt konto för en kund. När ett konto skapas ska de få ett nytt unikt kontonummer. Nya konton har alltid noll som saldo.

Ett konto har alltid en kund som ägare och måsta ha ett kontonummer och saldo.

Det ska även gå att ta bort konton och kunder. Konton får bara tas bort om saldo är noll. Kunder får bara tas bort om de inte har några konton. Det vill säga, det får inte gå att ta bort en kund som fortfarande har pengar på något av sina konton.

Systemet ska också hantera insättningar, uttag och överföringar mellan konton. Det får inte bli några avrundningsfel så använd rätt typ för saldo och belopp som är pengar.

Det ska framgå tydligt om någon försöker ta ut eller överföra mer pengar än vad som finns på kontot!

Exempel på hur programmets gränssnitt

```
C:\Bank> BankApp.exe bankdata.txt

*****************************
* VÄLKOMMEN TILL BANKAPP 1.0 *
*****************************

Läser in bankdata.txt...
Antal kunder: 123
Antal konton: 1234
Totalt saldo: 123456789.00

HUVUDMENY
0) Avsluta och spara
1) Sök kund
2) Visa kundbild
3) Skapa kund
4) Ta bort kund
5) Skapa konto
6) Ta bort konto
8) Insättning
8) Uttag
9) Överföring

> 1
* Sök kund *
Namn eller postort? eri
1234: Telefonaktiebolaget Ericsson
4567: Eriks Bageri AB

> 2
* Visa kundbild *
Kundnummer? 1234

Kundnummer: 1234
Organisationsnummer: 556016-0680
Namn: Telefonaktiebolaget Ericsson
Adress: TORSHAMNSGATAN 21, 164 40 Kista

Konton
12345: 123 456 789,00 kr
12346: 123 456 789,00 kr
12347: 123 456 789,00 kr

> 9
* Överföring *
Från? 12345
Till? 12346
Belopp? 1000

1 000,00 kr överfört från konto 12345 till 12346.

> 0
* Avsluta och spara *
Sparar till 20171231-2359.txt...
Antal kunder: 123
Antal konton: 1234
Totalt saldo: 123456789.00
```

NACKADEMIN

Krav för Väl Godkänt

Lösningen måste vara gjord med objektorienterad programmering. Separera olika funktioner så som filhantering till separata objekt så inte all funktionalitet hamnar i en stor fet Bank-klass. Läs mer om Single Responsibility Principle.

Lägg till ett transaktionsobjekt så det går att se historiken för dagens insättningar, uttag och överföringar på respektive konto. Det ska inte gå att ändra saldo på ett konto utan att det skapas en transaktion.

Lägg till en kontobild där man kan se transaktioner för ett individuellt konto.

När bankappen startar behöver du inte ladda in gamla transaktioner. Däremot ska varje transaktion sparas ned som sist i en textfil (t ex transaktionslogg.txt) med datum, inblandade konton, belopp och kontots saldo efter transaktion. Filen ska stängas efter varje transaktion.

Skapa ett **Unit Test**-projekt där du skriver tester som testar att det 1) inte går att ta ut eller 2) överföra mer pengar än det finns på kontot. Det ska inte heller 3) gå att sätta in eller 4) ta ut negativa belopp.

Lägg till stöd för sparränta på konton. Sparräntan är en årsränta och ska kunna anges och ändras per konto. Räntan måste vara positiv. Gör ett menyalternativ som räknar ut och lägger på dagens ränta på alla konton. En transaktion ska skapas på varje konto för insättningen av ränta.

Lägg till stöd checkkredit på konto genom att lägga till egenskaper för kreditgräns och skuldränta på varje konton. Kreditgräns och skuldränta ska kunna ändras per konto. Om ett konto har en kreditgräns på 10000 kr så kan pengar tas ut och överföras till saldot på konto är som lägst -10000 kronor. Komplettera rutinen för ränteberäkning med att lägga på skuldränta på konton där saldot är under noll. När skuldränta debiteras kan saldot gå under kreditgränsen.

Skapa Unit Tester som verifierar att uttag och överföringar fungerar med checkkredit och räntan beräknas korrekt.