Inlämningsuppgift 1 – Små Stålars AB (del 1 av 3)

Denna inlämningsuppgift handlar om den fiktiva banken Små Stålars AB, som i dagens läge enbart erbjudit sina kunder en mer personlig kontakt. Dessa tider kräver dock nästan att allt mer blir datoriserat och framförallt användarvänligt och funktionellt. För att Små Stålars ska kunna ta nästa steg framåt behöver de hjälp med att skapa en applikation där kunder kan komma åt sina konton, göra insättningar, uttag, kolla saldon på olika konton med mera.

Det är där du kommer in i bilden. Du ska hjälpa denna bank att skapa en applikation för bankens kunder. Till hjälp har de anlitat ett annat företag som har tittat över systemet som ska skapas och kommit fram till vilka delar som behövs implementeras. Ditt jobb blir att tillämpa dessa delar och framställa en fungerande och användarvänlig applikation (ej grafiskt användargränssnitt utan utskrifter och inmatningar sker i kommandofönstret).

En kund i banken har följande information i systemet:

- Namn (för- och efternamn)
- Personnummer
- Adress
- Postnummer
- Postort
- E-postdress
- Kundtyp (privat eller företag)

En kund kan ha ett eller flera av följande bankkonto eller kombinationer av dessa:

- Konto
  Detta är inte ett konto kunden kan ha, utan är ett "stödkonto" som måste implementeras i din lösning. Antingen som ett interface eller en abstrakt basklass. Kontot utgör grunden för resterande konton. Här kan instansvariabler och metoder som är gemensamma för alla typer av konton samlas. Alla konton ska minst ha ett unikt id, ett saldo och en sparränta. Id ska inte kunna förändras efter att ett konto är skapat och det får inte finnas två konton, oavsett kund, som har samma id. Det ska minst vara möjligt att se aktuellt saldo, att se och ändra sparräntan samt givetvis göra uttag och insättningar på kontot. Negativa belopp ska inte kunna sättas in eller tas ut.

- Sparkonto
  Detta är ett konto, även kallat lönekonto o.s.v., som minst ska ha en sparränta. Det ska vara möjligt att se och ändra sparräntan. Vid uttag från ett sparkonto får saldot aldrig bli negativt.

- Kreditkonto
  Detta är ett konto som innehåller en kreditdel (kunden kan "låna" på kontot upp till den kreditgräns som finns). Ett kreditkonto innehåller därför en kreditgräns och en kreditränta. Kreditgränsen anger hur mycket negativt saldot får blir innan ett uttag från kontot förhindras. Utöver detta ska det minst vara möjligt att se kreditgräns, aktuell kreditränta samt ändra krediträntan.

  Banken ska kunna hantera t.ex. skapandet av kunder och dess konton samt administrera dessa. I själva banksystemet kommer det finnas en mängd olika konton och även en mängd olika kunder. Dessa kan du lagra som en array med max 10 kunder i banken och max 10

konton för varje kund. I verkligheten skulle antalet kunder och konton vara mycket större och man skulle använda listor. Listor tas upp i lektion 2 men har du redan erfarenhet av listor är det ok att använda det redan nu.

Ett råd för att göra koden mer objektorienterad är att dela upp din lösning i flera olika klasser som sköter var sin begränsad del av systemet (Bank, Customer, Account m.fl). Använd arv och polymorfism om möjligt. Kan en uppräkningstyp (enum) vara en lämplig lösning för kundtyp? Funder en extra gång hur du ska ”länka” konton till kunden.

Skapa till sist en ”testklass” som på lämpligt sätt demonstrera alla övriga klasser och deras metoder. Minst två olika kunder med minst två konton vardera ska skapas. Det vill säga när applikationen starar ska minst två kunder skapas och läggas in i banken.

Prova att sätta in och ta ut pengar med både giltiga och ogiltiga belopp. Testklassen ska hela tiden meddela användaren vad som sker (om uttaget lyckades, hur mycket saldot är efter insättning m.m.).

Viktigt! Inga in- eller utmatningar från/till kommandofönstret får ske i någon annan klass än ”testklassen”. Använd i stället lämpliga returtyper i dina övriga klassers metoder.

Ett kort exempel på innehåll i testklassen kan vara:

```
System.out.println("Skapar bank");
Bank bank = new Bank("Små Stålars AB");

System.out.println("Skapar en kund med ett sparkonto och lägger in kunden i
banken");
Customer c1 = new Customer("112233-4455", "Pelle", "Nilsson", "Gatan 1", "80011",
"Orten", "pelle@nilsson.se", CustomerType.PRIVATE);
SavingsAccount a1 = new SavingsAccount(1, 200.5, 0.5);
c1.addAccount(a1);
bank.addCustomer(c1);

System.out.println("Tar ut 100 kr från kundens sparkonto");
boolean ok = bank.withdraw(1, 1, 100); // ta ut 100 kr från första kundens
första konto
if (!ok) {
  System.out.println("Uttag medgavs ej");
}
else {
  System.out.println("Nytt saldo är: " + bank.getBalance(1, 1));
}
// osv
```

Observera att metodanropen jag gör till klassen Bank endast är förslag på metoder som kan finnas. Ni får givetvis använda ett annat upplägg på lösningen och/eller andra metoder i Bank.