4 Projekt – anvisningar

- Projektet ska lösas individuellt eller i par (observera: inte större grupper än två och två) och ska redovisas på för ändamålet avsedd laborationstid.
- Var förberedd inför presentationen. Du ska vara beredd att svara på frågor kring ditt program och hur du har tänkt. Vid arbete i par måste båda kunna svara på alla frågor. Dela därför inte upp arbetet utan samarbeta kring samma program.
- Det är tillåtet att diskutera uppgifterna och tolkningen av dessa med utomstående på ett allmänt plan men *inte* att få hjälp med de konkreta lösningarna.
- Det är inte tillåtet att kopiera annans lösningar helt eller delvis. Det är inte heller tillåtet att kopiera från litteratur eller internet.
- Väsentlig hjälp, av annat än lärare på kursen, för att genomföra en uppgift skall redovisas tydligt. Detsamma gäller om man använt någon annan form av hjälpmedel som läraren inte kan förutsättas känna till.
- Kontakta ansvarig lärare om du är osäker på någon av punkterna ovan.

Projektuppgift – bankapplikation

Uppgiften är att skriva ett program som håller reda på bankkonton och kunder i en bank. Programmet är helt textbaserat: en meny skrivs ut till konsollen och menyvalen görs via tangentbordet. Du ska skriva programmet helt själv, och det ska följa de objektorienterade principer du lärt dig i kursen.

Krav

Kraven för programmet beskrivs nedan. Sist i häftet återfinns också ett antal exempelkörningar, som illustrerar hur ditt program ska fungera.

För att ditt program ska bli **godkänt** ska det uppfylla följande krav:

- Programmet ska ha följande funktioner:
  1. Hitta konton för en viss kontoinnehavare
  2. Sök kontoinnehavare på (del av) namn
  3. Sätta in pengar
  4. Ta ut pengar
  5. Överföring mellan konton
  6. Skapa nytt konto
  7. Ta bort konto
  8. Skriv ut bankens alla konton
  9. Avsluta
- Programdesignen ska följa de specifikationer som finns under avsnittet Design nedan.
- Du ska använda en lista (t.ex. `ArrayList`) för att hålla reda på alla bankkonton.
- Inga utskrifter eller inläsningar (`System.out`, `Scanner` och liknande) får lov att finnas i klasserna `Customer`, `BankAccount` eller `Bank`. Allt som gäller användargränssnitt skall skötas från klassen `BankApplication`. I `BankApplication` får du lov att använda valfritt antal hjälpmetoder eller andra hjälpklasser för att sköta användargränssnittet. (Det är vanligt att man begränsar användargränssnittet till vissa klasser på detta sätt. På så sätt kan klasserna `Customer`, `BankAccount` och `Bank` återanvändas i ett annat program med ett annat användargränssnitt.)
- Hela projektet får bara innehålla ett `Scanner`-objekt för att läsa från tangentbordet (dvs. man får bara göra `new Scanner(System.in)` en gång).
- Klasserna `Customer`, `BankAccount` och `Bank` ska exakt följa specifikationerna som anges under avsnittet Design nedan. Det är tillåtet att lägga till privata hjälpmetoder. (Det är inte ovanligt att en klass, för att fungera ihop med andra klasser, måste se ut på ett visst sätt – följa en viss specifikation – trots att man implementerar funktionaliteten själv.)
- Alla metoder och attribut ska ha lämpliga åtkomsträttigheter (private/public).
- De indata som ges i exemplet nedan ska ge väsentligen samma utskrift som exemplen i bilagan. Du får gärna byta ordval, men ditt program ska i allt väsentligt fungera likadant.
- Listor ska skrivas ut genom att iterera över listans objekt och anropa respektive `toString`-metod. Det är inte tillräckligt att använda listans `toString`-metod direkt, eftersom utskriften då inte får rätt format.

- Koden ska vara snyggt formaterad med korrekt indrag och klamrar placerade så att koden blir lättläst. I möjligaste mån ska kodkonventioner gällande namn på variabler och klasser följas (framför allt: tydliga namn, liten begynnelsebokstav för variabler och metoder samt stor begynnelsebokstav för klasser).
- Du behöver inte hantera exceptions men övrig rimlig felhantering är önskvärd. Exempelvis ska programmet inte krascha om användaren matar in kontonummer eller menyval som inte finns. Det är inte nödvändigt att hantera indata av fel typ (exempelvis att användaren matar in bokstäver då programmet förväntar sig ett heltal).

Tips och övriga anvisningar

- I vissa lösningar kan Scannern tyckas bete sig konstigt vid inläsning av omväxlande tal och strängar. Detta kan exempelvis visa sig i att inlästa strängar blir tomma.

  Mer information om detta problem hittar ni på Moodle.
- Tänk på val av attribut i klasserna. En variabel som bara används i en metod behöver inte vara åtkomlig i hela objektet och ska således ej vara attribut. Fråga en handledare om du är osäker.
- Observera att man inte ska kunna ta ut eller överföra pengar på ett sådant sätt att kontots saldo blir negativt. Av specifikationerna framgår att klassen `BankAccount` inte ska hantera övertrasseringar, men däremot ska `BankApplication` inte tillåta någon tar ut för mycket (vilket också framgår av exemplen i bilagan).
- Observera att `Customer` ska ha både ett idNr och ett kundnummer, dessa båda ska alltså inte sättas till samma värde. Kundnummer motsvarar ett löpnummer som är internt för banken och idNr kan tänkas motsvara verklighetens personnummer.
- Programmet ska testas noggrant. Tänk på att den som rättar kan mycket väl testa på helt andra fall än de som förekommer i exemplen. Till er hjälp finns testfall och rättningsprotokoll sist i denna uppgiftsbeskrivning. Läs igenom detta innan ni börjar programmera.

Frivillig utökning av uppgiften

Gör först klart inlämningsuppgiftens obligatoriska delar. Därefter kan du, om du vill, utöka ditt program enligt följande.

- Då man söker kunder utifrån en del av deras namn (menyval 2) kan samma kund förekomma flera gånger i resultatet. (Att det är samma kund ser man på att inte bara namn och personnummer är detsamma, utan även kundnumret.) Detta är naturligtvis inte helt önskvärt. Utöka gärna ditt program så att varje kund bara förekommer högst en gång i utskriften från menyval 2.
- En annan utökning, som även kan underlätta testningen av programmet, är att implementera läsning från och skrivning till fil och på så vis ge möjlighet att spara kund- och kontoinformationen mellan körningarna.

Design

Följande klass ska användas för att lagra grundläggande information om en kontoinnehavare:

`Customer`

```
/**
 * Skapar en kund (kontoinnehavare) med namnet 'name' och id-nummer 'idNr'.
 * Kunden tilldelas också ett unikt kundnummer.
 */
Customer(String name, long idNr);

/** Tar reda på kundens namn. */
String getName();

/** Tar reda på kundens personnummer. */
long getIdNr();

/** Tar reda på kundens kundnummer. */
int getCustomerNr();

/** Returnerar en strängbeskrivning av kunden. */
String toString();
```

Följande klass ska användas för att lagra information om ett bankkonto:

`BankAccount`

```
/**
 * Skapar ett nytt bankkonto åt en innehavare med namn 'holderName' och
 * id 'holderId'. Kontot tilldelas ett unikt kontonummer och innehåller
 * inledningsvis 0 kr.
 */
BankAccount(String holderName, long holderId);

/**
 * Skapar ett nytt bankkonto med innehavare 'holder'. Kontot tilldelas
 * ett unikt kontonummer och innehåller inledningsvis 0 kr.
 */
BankAccount(Customer holder);

/** Tar reda på kontots innehavare. */
Customer getHolder();

/** Tar reda på det kontonummer som identifierar detta konto. */
int getAccountNumber();

/** Tar reda på hur mycket pengar som finns på kontot. */
double getAmount();

/** Sätter in beloppet 'amount' på kontot. */
void deposit(double amount);

/**
 * Tar ut beloppet 'amount' från kontot. Om kontot saknar täckning
 * blir saldot negativt.
 */
void withdraw(double amount);

/** Returnerar en strängrepresentation av bankkontot. */
String toString();
```

Följande klass ska användas som register för att administrera bankens samtliga konton:

`Bank`

```
/** Skapar en ny bank utan konton. */
Bank();

/**
 * Öppna ett nytt konto i banken. Om det redan finns en kontoinnehavare
 * med de givna uppgifterna ska inte en ny Customer skapas, utan istället
 * den befintliga användas. Det nya kontonumret returneras.
 */
int addAccount(String holderName, long idNr);

/**
 * Returnerar den kontoinnehavaren som har det givna id-numret,
 * eller null om ingen sådan finns.
 */
Customer findHolder(long idNr);

/**
 * Tar bort konto med nummer 'number' från banken. Returnerar true om
 * kontot fanns (och kunde tas bort), annars false.
 */
boolean removeAccount(int number);

/**
 * Returnerar en lista innehållande samtliga bankkonton i banken.
 * Listan är sorterad på kontoinnehavarnas namn.
 */
ArrayList<BankAccount> getAllAccounts();

/**
 * Söker upp och returnerar bankkontot med kontonummer 'accountNumber'.
 * Returnerar null om inget sådant konto finns.
 */
BankAccount findByNumber(int accountNumber);

/**
 * Söker upp alla bankkonton som innehas av kunden med id-nummer 'idNr'.
 * Kontona returneras i en lista. Kunderna antas ha unika id-nummer.
 */
ArrayList<BankAccount> findAccountsForHolder(long idNr);

/**
 * Söker upp kunder utifrån en sökning på namn eller del av namn. Alla
 * personer vars namn innehåller strängen 'namePart' inkluderas i
 * resultatet, som returneras som en lista. Samma person kan förekomma
 * flera gånger i resultatet. Sökningen är "case insensitive", det vill
 * säga gör ingen skillnad på stora och små bokstäver.
 */
ArrayList<Customer> findByPartofName(String namePart);
```

Implementation

Implementationen görs lämpligen stegvis, genom att du implementerar några metoder i taget och därefter provkör resultatet. Därefter implementerar och provkör du fler metoder. Förslagsvis följer du dessa steg i ditt arbete:

1. Skapa projektet i Eclipse.

2. Implementera klassen `Customer`.

3. Implementera klassen `BankAccount`.

4. Skapa klassen `BankApplication`, som ska innehålla programmets `main`-metod.

   Det är lämpligt att införa ytterligare metoder i `BankApplication`: exempelvis en metod för att driva hela programmet (t. ex. `runApplication`) som i sin tur anropar andra metoder beroende på vad som ska göras. Ha gärna en separat metod för att skriva ut menyn.

   Implementera tillräckligt för att kunna köra igång programmet, skriva ut menyn och låta användaren välja i den.

5. Implementera menyval 6 och 8, som motsvarar metoderna `addAccount` och `getAllAccounts` i klassen `Bank`. Återigen kan du införa metoder i `BankApplication`, exempelvis för att läsa in data och anropa `Bank` med dessa.

   Observera din sortering i `getAllAccounts` inte ska använda någon av Javas standardklasser för sortering, som t. ex. `Collections.sort` eller `Arrays.sort`.

   Observera också återigen att det inte är tillåtet att använda `Scanner` och `System.out` i klassen `Bank`.

6. Testa menyval 6 och 8 noga.

   Tips: under testningen kan det bli enformigt att mata in uppgifter om konton om och om igen. Du kan lägga in satser i ditt program som anropar `addAccount` ett antal gånger, så att ditt program startar med ett antal färdiga testkonton. Låt dessa satser skapa kontona i icke-alfabetisk ordning (med avseende på kontoinnehavarnas namn), så att du ser att sorteringen i `getAllAccounts` fungerar.

7. Implementera menyvalen 1 och 2. De motsvaras av metoderna `findAccountsForHolder` och `findByPartOfName` i klassen `Bank`.

8. Lägg till några konton (om du inte införde testkonton enligt punkt 6 ovan) och testa menyvalen 1 och 2.

9. Implementera resten av menyvalen en åt gången. Testa efter varje.

10. Testa ditt program som helhet. Om du lade in satser för att automatiskt skapa testkonton (under punkt 6 ovan), ta bort dem.

    Kör först igång programmet utan att skapa några konton, och kontrollera att menyns alternativ fungerar även utan konton. Skapa därefter ett antal konton och kontrollera att programmets delar fungerar som förväntat.

    Glöm inte testa med orimliga indata och kontrollera att programmet hanterar dessa på ett rimligt sätt. Gå tillbaka och åtgärda de fel du upptäcker.

11. Innan redovisningen ska du gå igenom bedömningsprotokollet och testfallen som finns sist i detta avsnitt. Helst ska alla testfall fungera perfekt, men om något inte stämmer skriv upp det och diskutera det med din handledare.

Redovisning

När du gått igenom alla moment ovan, och testat noga, är det dags att redovisa. Förbered dig inför redovisningen. Om du hunnit glömma vad du gjort så repetera innan redovisningen – du förväntas kunna förklara ditt program!

Exempel på körning av programmet

Här visas en möjlig exempelkörning av programmet. Data som matas in av användaren visas i fetstil. Ditt program behöver inte efterlikna dessa utskrifter i exakta detaljer, men den övergripande strukturen ska vara densamma. De data som matas in av användaren har markerats i fetstil.

Ett antal konton skapas, och pengar sätts in på ett par av dem. Ett stort belopp överförs mellan två konton. Slutligen tas de flesta kontona bort, och endast ett finns kvar då programmet avslutas.

Tänk på att detta bara är ett exempel och inte en komplett beskrivning av hur programmet ska fungera. Programmet ska ju också fungera med de mer uttömmande testfallen som finns bilagda sist i denna uppgift.

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 6
namn: Doris Kruth
id: 4111194444
konto skapat: 1001
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 6
namn: Charles Ingvar Jönsson
id: 3705209999
konto skapat: 1002
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 6
namn: Jakob Morgan Rockefeller Wall-Enberg, Jr.
id: 2306207777
konto skapat: 1003
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 6
namn: Jakob Morgan Rockefeller Wall-Enberg, Jr.
id: 2306207777
konto skapat: 1004
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
```

```
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 2
namn: ROCK
Jakob Morgan Rockefeller Wall-Enberg, Jr., id 2306207777, kundnr 103
Jakob Morgan Rockefeller Wall-Enberg, Jr., id 2306207777, kundnr 103
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 2
namn: or
Doris Kruth, id 4111194444, kundnr 101
Jakob Morgan Rockefeller Wall-Enberg, Jr., id 2306207777, kundnr 103
Jakob Morgan Rockefeller Wall-Enberg, Jr., id 2306207777, kundnr 103
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 3
konto: 1003
belopp: 1000000.0
konto 1003 (Jakob Morgan Rockefeller Wall-Enberg, Jr., id 2306207777, kundnr 103): 1000000.0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 3
konto: 1001
belopp: 50.0
konto 1001 (Doris Kruth, id 4111194444, kundnr 101): 50.0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 4
från konto: 1001
belopp: 900.0
uttaget misslyckades, endast 50.0 på kontot!
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
```

```
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 5
från konto: 1003
till konto: 1001
belopp: 500000.0
konto 1003 (Jakob Morgan Rockefeller Wall-Enberg, Jr., id 2306207777, kundnr 103): 500000.0
konto 1001 (Doris Kruth, id 4111194444, kundnr 101): 500050.0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 4
från konto: 1001
belopp: 900.0
konto 1001 (Doris Kruth, id 4111194444, kundnr 101): 499150.0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 1
id: 2306207777
konto 1003 (Jakob Morgan Rockefeller Wall-Enberg, Jr., id 2306207777, kundnr 103): 500000.0
konto 1004 (Jakob Morgan Rockefeller Wall-Enberg, Jr., id 2306207777, kundnr 103): 0.0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 8
konto 1002 (Charles Ingvar Jönsson, id 3705209999, kundnr 102): 0.0
konto 1001 (Doris Kruth, id 4111194444, kundnr 101): 499150.0
konto 1003 (Jakob Morgan Rockefeller Wall-Enberg, Jr., id 2306207777, kundnr 103): 500000.0
konto 1004 (Jakob Morgan Rockefeller Wall-Enberg, Jr., id 2306207777, kundnr 103): 0.0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 7
konto: 1002
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
```

```
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 7
konto: 1003
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 7
konto: 1004
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 7
konto: 87654
felaktigt kontonummer!
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 8
konto 1001 (Doris Kruth, id 4111194444, kundnr 101): 499150.0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1. Hitta konto utifrån innehavare
2. Sök kontoinnehavare utifrån (del av) namn
3. Sätt in
4. Ta ut
5. Överföring
6. Skapa konto
7. Ta bort konto
8. Skriv ut konton
9. Avsluta
val: 9
```

Bedömningsprotokoll - bankuppgiften

Namn (programmerare):

1	*Programmets funktion och användarvänlighet*	Stämmer	Stämmer delvis	Stämmer ej	Kommentar
1.1	Programmets meny är snygg och tydlig				
1.2	Programmet är smidigt att använda.				
1.3	Programmet reagerar rimligt då man gör ett menyval som inte finns (d.v.s. matar in annan siffra än 1-9)				
1.4	Användaren får information i en en saklig och vänlig ton.				
1.5	Programmet löser den givna uppgiften enligt anvisningarna och testfallen som finns sist i dokumentet. Om något testfall misslyckas ange vilket, och på vilken position det misslyckas.				
1.7	Listningar skrivs ut i ett trevligt och begripligt format utan onödiga tecken. (exempelvis objekt som ligger i listor skrivs inte ut med listans toString-metod).				
1.6	Övrigt				

2	*Mjukvarudesign*	Stämmer	Stämmer delvis	Stämmer ej	Kommentar
2.1	Programmet är uppdelat i (minst) de klasser som anges i anvisningarna				
2.2	Användargränssnittet är avgränsat från modellen, det vill säga: Customer, BankAccount och Bank innehåller ingen Scanner och inga System.out				
3	*Implementering*	Stämmer	Stämmer delvis	Stämmer ej	Kommentar
3.1	Specifikationerna för klasserna givna i uppgiften följs (privata hjälpmetoder är tillåtna)				
3.2	Attributen är deklarerade private				
3.3	Inga onödiga attribut (d.v.s. inga variabler som borde varit lokala är deklarerade som attribut)				
3.4	BankAccount har två konstruktorer och båda används				
3.6	Övrigt				
4	*Programmets läsbarhet*	Stämmer	Stämmer delvis	Stämmer ej	Kommentar
4.1	Programkoden är lätt att följa och förstå.				
4.2	Alla namn är väl valda och återspeglar klassens, metodens, variabelns,… innebörd.				
4.3	Programmet har en vettig indentering				

Testning och testfall för sluttest - bankuppgiften

Läs igenom testfallen i ett tidigt stadium. Testfallen är gjorda för att kunna följas från början till slut. Men: Glöm inte att under utvecklingen hela tiden testa ditt program fritt efter eget huvud och med egenpåhittad data. Kör därefter testfallen efter att hela ditt program fungerar hyfsat, det är nämligen inte säkert att du bäst implementerar funktionerna i den ordning som testfallen har. Testfallen specificerar vilka situationer ditt program ska klara av, det hindrar inte din labhandledare från att testa situationerna på annat sätt, t.ex. i en annan ordning eller med annan data.

Testfall 1: Lista alla konto när inget konto finns i banken

- Notera att detta testfall ska köras på en bank utan konton

1. Välj menyalternativ 8 (skriv ut bankens konto)
2. Kontrollera att programmet inte krashcar
3. Kontrollera att det går att göra ett nytt menyval (t.ex. genom att påbörja testfall 2 direkt)

Testfall 2: Lägg till ett konto

1. Välj menyalternativ 6 (skapa nytt konto)
2. Ange kontoinformation enligt följande [namn: Anna B, personnummer: 1234] när programmet efterfrågar det
3. Kontrollera att du kommer tillbaks till menyn efter att kontot skapats
4. Välj menyalternativ 8 (skriv ut bankens konto)
5. Kontrollera att kontot du nyss skapade finns med
6. Kontollera att informationen om kontot är korrekt
7. Kontrollera att all information visas samt är tydlig. Det ska synas tydligt vilket värde som är saldo, vilket som är kontonummer och vilket som är kundnummer etc.
8. Kontrollera att du kommer tillbaks till menyn och kan göra ett nytt menyval

Testfall 3: Lägg till en kund med samma namn men annat personnummer

1. Välj menyalternativ 6 (skapa nytt konto)
2. Ange kontoinformation enligt följande [namn: Anna B, personnummer: 5678] när programmet efterfrågar det
3. Kontrollera att du kommer tillbaks till menyn efter att kontot skapats
4. Välj menyalternativ 8 (skriv ut bankens konto)
5. Kontrollera att både kontot du nyss skapade och kontot från testfall 2 finns med
6. Kontollera att utskriften ger dig två olika kontoinnehavare, dv.s. att kundnummer är olika för de båda kontona.
7. Kontrollera att all information visas samt är tydlig. Det ska synas tydligt vilket värde som är saldo, vilket som är kontonummer och vilket som är kundnummer etc.
8. Kontrollera att du kommer tillbaks till menyn och kan göra ett nytt menyval

Testfall 4: Skapa ytterligare konton för samma person

1. Välj menyalternativ 6 (skapa nytt konto)
2. Ange kontoinformation enligt följande [namn: Anna B, personnummer: 5678] när programmet efterfrågar det
3. Kontrollera att du kommer tillbaks till menyn efter att kontot skapats
4. Välj alternativ 1 (mata in personnummer 5678 när du blir ombedd)
5. Kontrollera att både kontot du nyss skapade och kontot från testfall 3 finns med
6. Kontrollera att kundnummerna är samma för de båda kontona i listan *(viktigt)*
7. Kontrollera att inga andra konton än ovan nämnda finns med
8. Kontrollera att du kommer tillbaks till menyn och kan göra ett nytt menyval

(notera, för godkännande kräver vi inte att programmet detekterar inmatningsfel av typen: samma personnummer men annat namn. Därför finns inget testfall för detta)

Testfall 5: Sorterad lista

1. Lägg till följande konton: [namn: Jesper Z, personnummer: 1313] [namn: Anders A, personnummer: 1212],
2. Välj menyalternativ 8 (skriv ut bankens konto)
3. Kontrollera att namnen i kontoutskriften kommer i bokstavsordning (Anders A, Anna B, Jesper Z). Den inbördes ordningen mellan de olika Anna B är valfri

Testfall 6: Sök på del av namn (case-insensitive, en träff)

1. Välj menyalternativ 2 (Sök på kontoinnehavares (del av) namn)
2. När programmet frågar efter söksträngen: mata in jesper
3. Kontrollera att utskriften ger dig information om kontot skapat för Jesper Z i testfall 5
4. Kontollera att inga andra konton är med i utskriften

Testfall 7: Sök på del av namn (många träffar)

1. Välj menyalternativ 2 (Sök på kontoinnehavares (del av) namn)
2. När programmet frågar efter söksträngen: mata in An
3. Kontrollera att båda Anna B samt Anders A dyker upp i listningen
   (för uppgiften är det ok om den Anna B som har två konton dyker upp två gånger, det är en frivillig uppgift att sortera bort dessa dubletter)

Testfall 8: Sök på del av namn (ej början av strängen)

1. Välj menyalternativ 2 (Sök på kontoinnehavares (del av) namn)
2. När programmet frågar efter söksträngen: mata in ders
3. Kontrollera att Anders A dyker upp i listningen
4. Kontrollera att inga andra konton är med i uppgiften

Testfall 9: Sätta in pengar, normalfallet

1. Välj menyalternativ 8 (skriv ut bankens konto)
2. Notera kontonummret som Jesper Zs konto har
3. Välj menyalternativ 3 (Sätt in pengar)
4. När programmet frågar efter kontonummer: ange numret som du hittade i steg 2
5. När programmet frågar efter belopp: ange 500
6. Välj menyalternativ 8 (skriv ut bankens konto)
7. Kontrollera kontot för Jesper Z, saldot ska vara 500

Testfall 10: Sätta in pengar, negativt belopp

1. Välj menyalternativ 3 (Sätt in pengar)
2. När programmet frågar efter kontonummer: ange det kontonummer du använde i testfall 9
3. När programmet frågar efter belopp: ange -1000
4. Kontrollera att programmet meddlear att negativa belopp ej är tillåtna
5. Välj menyalternativ 8 (skriv ut bankens konto)
6. Kontrollera kontot för Jesper Z, saldot ska vara 500

Testfall 11: Sätta in pengar, kontrollera att det blir rätt konto

1. Välj menyalternativ 8 (skriv ut bankens konto)
2. Notera de kontonummer som Anna Bs båda konton har
3. Välj menyalternativ 3 (Sätt in pengar)
4. När programmet frågar efter kontonummer: ange ett av de nummer som du hittade i steg 2
5. När programmet frågar efter belopp: ange 5000
6. Välj menyalternativ 8 (skriv ut bankens konto)
7. Kontrollera kontona för Anna B, kontrollera att saldot är 5000 på det konto du valde i steg 4.
8. Kontrollera att saldot är noll på Anna Bs andra konto

Testfall 12: Sätta in pengar, kontot finns inte

1. Välj menyalternativ 3 (Sätt in pengar)
2. När programmet frågar efter kontonummer: ange ett nummer som du vet inte finns i banken
3. Om programmet frågar efter belopp: ange 500
4. Kontrollera att du får meddelande om att kontot ej existerar
5. Kontrollera att programmet inte krashar
6. Kontrollera att du kommer tillbaks till menyn och kan göra ett nytt menyval
7. Välj menyalternativ 8 (skriv ut bankens konto)
8. Kontrollera så att samtliga konton har oförändrat saldo

(notera: när kontot inte finns är det valfritt om programmet frågar efter belopp eller ej)

Testfall 13: Ta ut pengar, normalfallet

1. Välj menyalternativ 8 (skriv ut bankens konto)
2. Notera kontonummret som Jesper Zs konto har
3. Välj menyalternativ 4 (Ta ut pengar)
4. När programmet frågar efter kontonummer: ange numret som du hittade i steg 2
5. När programmet frågar efter belopp: ange 100
6. Välj menyalternativ 8 (skriv ut bankens konto)
7. Kontrollera kontot för Jesper Z, saldot ska vara 400

Testfall 14: Ta ut pengar, kontot finns inte

1. Välj menyalternativ 4 (Ta ut pengar)
2. När programmet frågar efter kontonummer: ange ett nummer som du vet inte finns i banken
3. Om programmet frågar efter belopp: ange 500
4. Kontrollera att du får meddelande om att kontot ej existerar
5. Kontrollera att programmet inte krashar
6. Kontrollera att du kommer tillbaks till menyn och kan göra ett nytt menyval
7. Välj menyalternativ 8 (skriv ut bankens konto)
8. Kontrollera så att samtliga konton har oförändrat saldo

Testfall 15: Ta ut pengar, täckning saknas

1. Välj menyalternativ 8 (skriv ut bankens konto)
2. Notera kontonummret som Anders As konto har
3. Välj menyalternativ 4 (Ta ut pengar)
4. När programmet frågar efter kontonummer: ange numret som du hittade i steg 2
5. Om programmet frågar efter belopp: ange 100
6. Kontrollera att du får meddelande om att det inte finns tillräckligt med pengar
7. Välj menyalternativ 8 (skriv ut bankens konto)
8. Kontrollera kontot för Anders A, saldot ska vara noll

Testfall 16: Överföring av pengar, normalfallet

1. Välj menyalternativ 5 (Överföring mellan konton)
2. Se till att ha noterat kontonummerna för Anders A och Jesper Z (noterades i tidigare testfall)
3. När programmet frågar vilket konto pengarna ska överföras från: mata in kontonummret för Jepser Zs konto
4. När programmet frågar vilket konto pengarna ska överföras till: mata in kontonummret för Anders As konto
5. När programmet frågar efter belopp: ange 200
6. Välj menyalternativ 8 (skriv ut bankens konto)
7. Kontrollera att Anders A och Jesper Z båda har saldot 200

Testfall 17: Överföring av pengar, täckning saknas

1. Välj menyalternativ 5 (Överföring mellan konton)
2. Se till att ha noterat kontonummerna för Anders A och Jesper Z (noterades i tidigare testfall)
3. När programmet frågar vilket konto pengarna ska överföras från: mata in kontonummret för Jepser Zs konto
4. När programmet frågar vilket konto pengarna ska överföras till: mata in kontonummret för Anders As konto
5. När programmet frågar efter belopp: ange 6000
6. Välj menyalternativ 8 (skriv ut bankens konto)
7. Kontrollera att Anders A och Jesper Z båda har saldot 200

Testfall 18: Ta bort konto (normalfallet)

1. Välj menyalternativ 8 (skriv ut bankens konto)
2. Notera kontonummret som Jesper Zs konto har
3. Välj menyalternativ 7 (Ta bort konto)
4. När programmet frågar efter kontonummer: ange numret som du hittade i steg 2
5. Välj menyalternativ 8 (skriv ut bankens konto)
6. Kontrollera att kontot för Jesper Z inte finns i listan
7. Välj menyalternativ 2 (Sök på kontoinnehavares (del av) namn)
8. När programmet frågar efter söksträngen: mata in jesper
9. Kontrollera att programmet meddlear att inget sådant konto finns
10. Kontrollera att programmet inte krashar utan återgår till menyn som vanligt

(notera: man behöver inte ta hänsyn till att det finns pengar på det konto som ska tas bort. Om man valt att hantera det på nåt vis så är det också ok)

Testfall 19: Ta bort konto (flera konton finns)

1. Välj menyalternativ 8 (skriv ut bankens konto)
2. Notera de kontonummer som Anna Bs båda konton har
3. Välj menyalternativ 7 (Ta bort konto)
4. När programmet frågar efter kontonummer: ange ett av de nummer som du hittade i steg 2
5. Välj menyalternativ 8 (skriv ut bankens konto)
6. Kontrollera att kontot du nyss tog bort inte finns i listan
7. Kontrollera att Anna Bs andra konto finns kvar
8. Välj menyalternativ 1
9. Mata in personnummer 5678 när du blir ombedd
10. Kontrollera att Anna B bara dyker upp med ett konto

Testfall 20: Avsluta

1. Välj menyalternativ 9 (avsluta)
2. Kontrollera att programmet slutar köra (om du är i Eclipse ser du det genom att den röda stoppknappen inte längre lyser)

Testfallen är egentligen inte kompletta. För att verkligen hårdtesta hela systemet kan man anse att det finns vissa luckor. Fundera på vilka testfall ni tycker saknas.