Inlämningsuppgifter, EDAF30, 2015

Av

Artur Hellberg

&

Simon Lerpard

Inlämningsuppgift del 1- Bank

Struktur på programmet

Klass main: Hanterar menyn och använder funktionerna i Bank-klassen för att köra systemet.
Klass Bank: Innehåller samling av Konton och alla funktioner för att operera på dessa.
Klass Konto: Innehåller fyra attribut samt funktioner för att hämta och ändra dessa.

Kort information om hur programmet är uppbyggt

Systemet består av tre klasser. Bank, Konto och Main. Main körs som vanligt och är huvudprogrammet. Klassen består av två metoder, main() och printMenu(). Denna klass-funktion är framför allt för att skapa menysystemet med en switch-case sats samt att skapa en så kallad Bank-klass.

Bank-klassen är klassen som hanterar allting som en bank i normala fall hanterar, så hantering samt ändringar av konton sköts via denna klass. När man öppnar (Skapar) ett nytt konto så skapas en instans av klassen Konto. Varje konto sparas undan i en vektor vid namn Konton. Denna vektor används för att samla alla konton i programmet. Vi valde en vektor för att det är ett enkelt och effektivt sätt att hämta och söka efter en samling av objekt, i detta fall en samling av klassen Konto.

Ett Konto består av några privata attribut, ”kontoinnehavare”, ”kontonummer”, ”kontotyp” och ”saldo”. Detta är vad som sparas för varje konto i systemet.

I Bank klassen finns en hel del metoder, både privata och publika. Angående metoduppbyggandet finns det t.ex. en publik metod med namn sattIn() och en privat metod med samma namn fast fler inparametrar, sattIn(int kontonummer, int belopp). Som regel hanterar den publika metoden inmatningen för vad som krävs av den privata metoden, i detta fall kontonummer och belopp.

Ladda & Spara

När det kommer till att spara och ladda all data i systemet så görs detta med två metoder i klassen Bank, void ladda() och void spara(). Metoden ladda() körs enbart när skapandet av Bank-klassen görs, alltså i konstruktorn.
När det gäller metoden spara() så körs denna varje gång ett värde läggs till eller ändras. T.ex. ett namnbyte på kontoinnehavaren för ett konto, eller att ett nytt konto skapas.

Det som sparas i en fil är de attribut som finns i klassen Konto. Detta läses av från alla konton och skrivs ut som ett attribut per rad. När systemet laddar filen läses varje rad av för sig och skapar då nya konton med de attribut som finns sparade i filen.

Textfilen där datan sparas är väldigt känslig. Det finns ingenting som beskriver vad ett värde är för något i textfilen. Det är enbart ordningen den sparades ned på. Skulle man ta bort en rad i filen och sedan ladda programmet kommer fel att uppstå. Detta är något vi har uteslutit att implementera eftersom vi behövde begränsa oss lite och att en användare ska aldrig behöva ändra i en data-fil.

Användning av programmet:

1. Starta programmet
2. En meny visas och välj önskat val, förslagsvis val nr 1. Skapa konto, (Skriv 1 och tryck enter för att välja).
    1. När programmet ber dig skriva in kontoinnehavare skriv in ett namn. T.ex. Bob. (stora och små bokstäver spelar roll)
    2. När programmet ber dig välja kontotyp, skriv in en bokstav (S eller T) för att välja Sparkonto eller Tjänstekonto. Stora och små bokstäver spelar ingen roll här.
    3. Ett nytt konto är nu skapat för att se vad kontot innehåller samt vilket kontonummer det fick kan man enkelt hitta kontot via alternativ 5 eller 6 från huvudmenyn.
3. Fortsätt via huvudmenyn, välj alternativ 0 för att stänga programmet.

Brister och kommentarer

Som vi nämnt innan så borde man möjligtvis göra en tydligare datafil. Programmet består också väldigt mycket av inmatningskontroller. Detta borde gå att göra på ett tydligare sätt med hjälp av några hjälpmetoder för varje typ i stället för varje inmatning. Man borde kanske också ha flyttat ut filhanteringen från Bank klassen. Kanske skapa en ny klass för just filhantering. Filhantering är egentligen ingenting banken borde behöva bry sig om mer än att möjligtvis ropa på metoderna spara() och ladda(). Och därför borde de egentligen ligga i en egen klass eller åtminstone utanför.

Programmet är skrivet på svenska till största del. Det finns några inlånade ord som t.ex. print. Men egentligen borde programmet vara helt på engelska eftersom engelska är världsspråket och flest människor som förstår det. Det hade blivit mer enhetligt rent språkmässigt, men spelar mindre roll i denna uppgift.

Vi missade delen att separera på användargränssnittet och styrprogrammet. Därför implementerade vi uppgiften med krav för t.ex. inmatning på ett visst sätt för att göra det enkelt för användaren. Nu i efterhand, när det inte finns tillräckligt med tid för att slutföra uppgiften på önskat sätt kommer vi inte hinna ändra detta på ett bra sätt. Därför väljer vi att ha kvar det på detta vis, skulle det verkligen behövas ändras på så får vi komplettera den delen. Att göra ett testprogram som testar kan gå att göra, men det kräver väldigt mycket tid och omstrukturering. Vi har alltså inget sådant i nuläget.

Av Artur Hellberg, dat13ahe och Simon Lerpard, dat13sle. 7/1-2015

Inlämningsuppgift del 2 - Kanin

Vår lösning till uppgiften var att dela upp ansvarsområdet mellan kaninen, tävlingsbanan och ”arrangören” när det gällde ansvarsområdena, som inne i klasserna delas mellan olika funktioner. Arrangören (RabbitRace) skapar ett önskat antal kaniner mellan 1 till 999 och tilldelar kaninerna var sin bana (Track) och en identifierare. Banorna är identiska, men åtskilda för att inte störa varandra, likt en löpbana där varje löpare har sin avgränsade yta att springa på.

Arrangören avancerar sedan loppet i x antal sekunder i taget (i medföljande main-metod 1 sekund) och presenterar resultatet av varje sekund i ett kommandofönster. Detta upprepas tills en av kaninerna passerar mållinjen, då tävlingen avbryts och kaninerna presenteras en sista gång innan vinnaren utses samt antalet sekunder loppet pågick.

Klasser:

Track: Denna klass representerar tävlingsbanan och ansvarar för att hålla reda på banans längd, var på banan kaninen befinner sig och om/när kaninen har nått målet (som är slutet på banan). Varje kanin tilldelas en bana.

- Track(int trackLenght): Skapar en ny bana i angiven längd.
- bool reachedGoal(): returnerar True om kaninens position är like med eller längre än vad banan är, annars false.
- int getLenght(): returnerar banans längd.
- int getCurrentPosition(): Returnerar kaninens nuvarande position.
- void movement(int nbrOfSteps): Uppdaterar kaninens position med de angivna siffrorna.

Rabbit: Denna klass representerar kaninen och ansvarar för att hålla reda på kaninens identitet (i detta fall ett nummer), sannolikhet för olika beteenden och intervallet för hur långt kaninen kan hoppa. Kaninen ansvarar även för att rapportera till sin bana (Track) om hur långt den hoppar samt åt vilket håll.

- Rabbit(int nbr, Track track): Tilldelar kaninen ett nummer och en bana.
- void setProbabilities(int stay, int jumpForward): Tilldelar sannolikheterna (i procent) till kaninen. Om funktionen inte anropas så används default-värderna.
- void setMoveDistances(int minimum, int maximum): Tilldelar intervallet som kaninen hoppar. Om funktionen inte anropas så används default-värderna.
- int getInterval(): returnerar ett slumptal inom hopp-intervallet.
- int movement(): Returnerar hur långt kaninen hoppade denna runda (sekund).
- int getProgress(): Returnerar hur långt kaninen har hoppat totalt.
- int getRabbitNbr(): Returnerar kaninens identifierare.
- bool hasWon(): Returnerar true om kaninen har vunnit, annars false.

RabbitRace: Denna klass ansvarar för att skapa ett x antal kaniner, tilldela dem en identifierare och bana, se till att tävlingen avancerar med en sekund i taget och presentera vinnaren av tävlingen. Detta är klassen som används av main-metoden.

- RabbitRace(int nbrOfRabbits): Skapar ett antal kaniner efter givet tal.
- bool advance(int nbrOfRounds): Avancerar loppet med angivet antal sekunder. Returnerar true om en vinnare har utsetts, annars false.
- void printStandings(): Skriver ut de tävlandes positioner på banan.
- int getWinner(): Returnerar identifieraren till vinnaren. Om ingen vinnare har utsetts returneras -1.

Main: Denna klass skapar en arrangör (RabbitRace) och ansvarar för att förmedla mellan användaren och arrangören, samt att tolka och agera efter inmatningar av arrangör och/eller användaren. T.ex. vid felaktig inmatning av användaren i val av antalet kaniner så ges arrangören ett default-värde (i detta fall 1) för att undvika konflikter.

Användarmanual:

1. Starta main-klassen. Ett kommandofönster ska öppnas.
2. Följ första instruktionen genom att mata in ett tal mellan 1 och 999. Tryck ENTER.
3. Programmet kommer att skriva ut ställningarna efter varje sekund. I slutet av tävlingen kommer vinnaren att presenteras eller, om tävlingen drog ut på tiden, så kommer ett meddelande ut om att tävlingen avbröts.
4. Avsluta programmet genom valfritt knapptryck. Om ni önskar att köra om loppet, gå tillbaka till steg 1.

Kommentarer:

Något som borde ha förbättrats i klassen Rabbit är slumpgeneratorn, som ger nästan lika mycket variation som ett konstant tal. Detta för att de mycket större variation i programmet när det gäller kaninernas beteenden och längden på deras hopp.

En annan notering är att även om testklassen ger mycket utskrifter så beror detta på att det enorma antalet kaniner som tävlar och inte på att programmet hamnat i en så kallad ”loop”. Kom ihåg detta när testprogrammet körs.