( = svår)

Klasser, instansvariabler och metoder

Förtydligande: Det finns inget rätt sätt i hur man visar värdena i konsolen. Huvudsaken är att man själv förstår att det man gjort fungerar som man tänker. Du behöver inte heller visa grejer från tidigare uppgifter om du inte vill. Målet är att DU ska känna att du förstår klasser, objekt och kan det.

1. Programmera en klass som representerar ett hus. Huset ska ha en yta, antal våningar, värde, byggnadsår och en adress. Skriv ett program som visar detta i konsolen.

2. Fortsätt i samma program som tidigare uppgift. Programmera en klass som representerar en person som vi gjorde på föreläsningen. Personen ska ha ett namn, en sysselsättning, en ålder, ett kön och en längd. Visa detta i konsolen.

3. Skapa konstruktorer för dina klasser så att de tar variablerna som nämnts som argument och skapar upp objektet.

4. Fortsätt i samma program som tidigare uppgift. Skapa upp två hus och tre personer i ditt program. Ändra på person-klassen så varje person kan ha ett hem (med referens till huset de bor i. Visa i konsolen vilken adress varje person bor på (du får själv välja vilket hus varje person bor i).

5. Fortsätt i samma program som tidigare uppgift. Programmera en klass för en bil. Bilen ska ha en modell, ett värde och en km-räknare. Låt nu varje hushåll ha en bil (huset har en referens till bilen). Visa husen och vilka personer som bor där och vilka bilar som finns där. 2 bilar räcker.

6. Fortsätt i samma program som tidigare uppgift. Programmera följande nya egenskaper för klasserna:

- Bilen - en metod som gör att bilens värde går ner med 20% varje år
- Personen - en metod som gör att personens ålder går upp 1 år
- Huset - en metod som gör att husets värde går upp 5% varje år

7. Gör en metod för varje klass som skriver ut alla värden i en läsbar sträng. Kalla metoden för toString().

8. Fortsätt i samma program som tidigare uppgift. Skriv ut värdena på alla objekt i början av programmet. Låt sedan 20 år gå där vi för varje år ändrar värdena ett steg. Visa vad värdena blir efter 20 år.

Grattis! Du har nu gjort The Sims i textformat (nästan) :-)

Getters och setters

Uppgift 1

Nu ska ni skapa en klass för att representera en person. Personen ska ha ett förnamn, ett efternamn samt en ålder, som ska lagras i privata instansvariabler. Skriv en konstruktor med tre argument, som intierar variablerna på ett lämpligt vis.

Skriv getter- och settermetoder för personens egenskaper. Se också till så att åldern bara får anta ett värde mellan 0 och 100. Om något felaktigt skrivs in, ge personen åldern 20.

Skriv ett main-program som testar er klass genom att skapa ett par olika personer. Ändra personernas egenskaper m.h.a. setter-metoderna, och skriv ut deras instansvariabler m.h.a. getter-metoderna för att se så att objekten uppdaterades ordentligt.

Testa också att försöka ändra en persons ålder till något otillåtet och sedan skriva ut den - vad händer då?

Uppgift 2

Nu ska ni utöka Person-klassen från uppgift 1 med två nya setter-metoder för namnet.
Skapa en metod setName() som tar två argument, förnamn och efternamn.
Skapa sedan en metod setName() som enbart har ett argument, som man kan kalla om man bara vill förändra förnamnet.

Utöka ert testprogram så att det testar båda dessa metoder.

Uppgift 3

Utöka Person-klassen med en statisk variabel (som såklart ska vara private) som ska hålla koll på antalet personer som har skapats (det vill säga antalet instanser som existerar). Gör så att denna variabel inkrementeras varje gång en person skapas.

Skapa också en getter-metod för denna variabel, och utöka ert testprogram så att den skriver ut antalet persons som skapats vid olika tillfällen.

Uppgift 3.1

I ert testprogram:
Försök att hitta ett sätt att skriva ut counterns värde när den är 0, dvs innan ni har skapat några instanser.

Skapa klasser

Uppgift 1. 

Uppgiften är att skapa en klass som kan användas för att representera tid. Den ska spara timmar, minuter och sekunder.

Konstruktorer

Skapa följande konstruktorer, och utnyttja constructor chaining:
Time(int h)
Time(int h, int m)
Time(int h, int m, int s)
Konstruktorerna ska inte tillåta konstiga värden, t.ex. negativa siffror eller minuter över 59. Då kan de kasta ett exception med följande kod:
throw new IllegalArgumentException("Time: Bad value: " + h + ":" + m);

Getters och setters

Skapa getters och setters för timmar, minuter och sekunder.
Även dessa ska säkerställa att tiden är korrekt, på samma sätt som konstruktorerna.

toString

Skapa en metod toString som skriver ut den nuvarande tiden på 24-timmars-format (t.ex. "13:39:12"). Tänk på att denna ska skriva ut t.ex. "13:02:12" om minuterna är 2!

toString12

Skapa en metod toString12 som skriver ut den nuvarande tiden på 12-timmars-format (t.ex. "1:39:12 PM"). Även denna ska klara "1:02:12 PM".
Det finns tre fall när man pratar om AM/PM:

- Innan klockan 12 är AM.
- Emellan 12 och 13 är PM, men skrivs med 12. (12:30 => 12:30 PM)
- 13 och efter är PM med 12 subtraherat från timmarna

isAM

Skriv en metod som kollar om tiden är i AM eller PM just nu.
public boolean isAm()

incSecond/incMinute/incHour

Skriv metoder som inkrementerar sekunderna/timmarna/minuterna. Om de når sitt maxvärde ska de "slå över", samt öka den större enheten (om man inkrementerar 0:00:59, så blir det 0:01:00)

compareTo
Skapa en metod compareTo, som jämför den nuvarande tiden med en annan tid:
public int compareTo(Time other);
Den ska returnera ett negativt tal om this är mindre än other,
Ett positivt tal om this är större än other,
och 0 om this är lika med other.

Uppgift 2. 

Uppgiften här är att implementera en egen datastruktur som fungerar som ArrayList. Den skall vara en egen klass som i sin tur innehåller en array. För att listan skall kunna lagra vilken typ som helst skall arrayen vara en Object-array (Object[]). Eftersom alla klasser ärver klassen Object utnyttjar vi något som heter polymorfism för att kunna lagra vad som helst. Både arv och polymorfism kommer vi att ta upp v4, och du behöver inte förstå det för att göra uppgiften. Det bygger på att alla klasser "är" ett Object, och kan därför användas som ett.

Klassen skall ha en Object-array som lagrar alla element, denna arrayen skall man aldrig komma åt utanför klassen, utan man manipulerar den med hjälp utav metoder.

Följande metoder skall finnas i klassen:

public void add(Object o)
Skall lägga **o** sist i den inre arrayen.

public void add(int i, Object o)
Skall lägga **o** på index **i**. Den skall inte skriva över det som ligger på plats **i** utan flytta det ett steg uppåt.

public Object get(int i)
Skall returnera det som ligger på plats **i**

public void remove(int i)
Skall ta bort det som ligger på plats **i** och flytta de efterföljande elementen ett steg nedåt för att "fylla ut" hålet.

public void set(int i, Object o)
Byter ut det värdet som ligger på plats **i** med objektet **o**.

public void size()
Returnerar hur många element som ligger i arrayen. Alltså inte storleken på arrayen, då den

ofta är större än antalet element i den!

(Finns många fler exempel på metoder du kan lägga till om du redan har gjort dessa. Ta en titt på javadocen för ArrayList om du vill ha inspiration.)

Den inre arrayen skall automatiskt "förstoras" (bytas ut mot en större) när add anropas om den är full. Ett lämpligt sätt att öka storleken är att dubbla den när detta sker, men det är valfritt hur du väljer att göra.

Tips:
För att förstora arrayen måste ni skapa en ny, då arrayer har en bestämd storlek.

Tips:
Det är dessutom lämpligt att ha en variabel som håller koll på hur många element man har lagt till i listan, för att kunna veta "storleken" på listan och vart man skall lägga nästa element.

Utökning:
Ett problem som er list-klass har nu är att get returnerar typen Object. Vill du göra så att man definierar en typ som listan skall innehålla kan du läsa på om generiska typer. Detta är dock rätt komplicerat, så det är ingenting som förväntas att ni ska kunna. Vi kommer att gå igenom generiska typer senare under kursen.