Exercises for Inheritance

Inheritance-01

Läs filerna i

Inheritance/Inheritance01/Exercises/src

`Person.java`

`Student.java`

`Employee.java`

`Register.java`

Person.java är vad man kallar en superklass (eller basklass). Där ligger det som är gemensamt för personer som skall lagras i registret.

Subklasser (ärvande klasser) är Student och Employee. Bägge ärver klassen Person. Man kan säga att "en Student är en Person" respektive "en Employee är en Person". Detta är ett tecken på att arv går att använda.

Titta på filerna Student.java och Employee.java

Vilket är det Java-keyword (ord som hör till Java-språket) som visar att dessa klasser ärver klassen Person?

Inheritance-02

Titta på filen Employee.java

Klassen Employee har en konstruktor:

```
public Employee(String pNumber, String address, String name, long salary){
    super(pNumber, address, name);
    this.salary = salary;
}
```

Konstruktorn tar fyra parametrar men konstruktorn i Person (superklassen/basklassen) tar bara tre. Det första som händer i konstruktorns kropp är:

```
    super(pNumber, address, name);
```

Detta är ett anrop till superklassens konstruktor och den anropas med de tre argument som är relevanta för Person (personnummer, adress och namn). Detta är ett vanligt sätt att initiera ett objekt i en subklass, att man anropar superklassens konstuktor (med det Java-keyword som heter `super`) och sedan utför övrig initiering av sådant som är unikt för den ärvande klassen (subklassen, i detta fall Employee och det som är unikt är i detta fall `salary`).

En regel i Java är att ett sådant anrop till konstruktorn i superklassen från en konstruktor i en ärvande klass måste vara det första som sker i konstruktorns kropp (ske på den första raden i konstruktorblocket).

En annan regel i Java är att om man inte skriver något anrop till `super()` (dvs inget anrop till någon konstruktor i superklassen), så lägger kompilatorn till ett anrop till `super()` precis så, utan parameter, först i konstruktorns kropp. Om det inte finns någon konstruktor i superklassen som inte tar några parametrar så blir det ett kompileringsfel.

Uppgift 2a:

Kommentera bort anropet till `super(pNumber, address, name);` i Employee.java och försök kompilera. Vilket blir kompileringsfelet?

Uppgift 2b:

Byt plats på raderna så att `this.salary = salary;` är första raden och `super(pNumber, address, name);` istället är andra raden. Detta är alltså att bryta mot regeln. Försök kompilera och ange vad kompilatorns felmeddelande är. Det är nu två fel, det vill säga, vi har brutit mot två regler. Vilka är reglerna?

Inheritance-03

Låt säga att Chalmers vill lägga till en sorts personer i sitt register, GuestLecturer. Man vill fortfarande kunna använda mail-metoden för att skicka vykort, utan att skriva om den. Lösningen på detta var ju att låta Student och Employee ärva Person och låta mail-metoden anropas med en referens till någon sorts Person som argument.

a.

Skapa klassen GuestLecturer i samma katalog som övriga filer. Klassen skall anges ärva Person. Skriv minst en konstruktor som tar som argument personalNumber, address, name. För att kunna kompilera klassen måste du (se uppgift 2a,b) anropa superklassens konstruktor med tre argument.

b.

Skriv om Register.java så att main-metoden skapar också en GuestLecturer med värden på personalNumber, address och name som du hittar på själv. Anropa mail med referensen till denna GuestLecturer som argument och någon vykortstext.

Inheritance-04

Skapa en klass GreatApe.

Inheritance-05

Skapa en klass Chimpanzee som ärver GreatApe.

Extra: Gör GreatApe till en klass som man inte kan ärva ifrån (final class). Testa att kompilera koden. Vi kommer att gå igenom detta i kommande föreläsningar.

```
public final class GreatApe { .....
```

Inheritance-06

Skriv egna konstruktorer till de båda klasserna ovan.

Inheritance-07

Skapa ett Chimpanzee-objekt i en separat klass, t ex Apan.

Ta reda på hur konstruktorerna anropas. T ex kan du göra detta genom att lägga till en utskrift (System.out.println) i konstruktorerna.

Inheritance-08

Skapa ett Chimpanzee-objekt i en separat klass, t ex Apan.

Spara detta objekt som ett GreatApe-objekt. Kan du göra detta? Varför?

Inheritance-09

Lägg till följande i GreatApe:

Klassvariabel:

```
String Genus = "Pan";
```

Instansvariabel:

```
String name;
```

OBS: För att inte kunna ändra värde på en varibel kan man använda ordet final. T ex så här:

```
public static final String Genus = "Pan";
```

Testa om du kan ändra på klassvaribeln Pan efter att du satt den till final.

Inheritance-10

Skapa en konstruktor för GreatApe som tar ett namn (t ex “Herr Nilsson”) som parameter.

Går det att kompilera GreatApe? Går det att kompilera Chimpanzee?

Vad säger kompilatorn? Vad betyder det? Läs kompilatorns utskrift ordentligt?

OBS: I Inheritance-12 löser vi problemet

Inheritance-11

Skapa en konstruktor för Chimpanzee som tar ett namn (t ex “Herr Nilsson”) som parameter.

Går det att kompilera GreatApe? Går det att kompilera Chimpanzee?

Vad säger kompilatorn? Vad betyder det? Läs kompilatorns utskrift ordentligt?

OBS: I Inheritance-12 löser vi problemet

Inheritance-12

I konstruktor för Chimpanzee, som ju tar en String som parameter, skall vi nu anropa vår superklass konstruktor.

Hur gör vi detta?

Betrakta konstruktorn nedan:

```
public Chimpanzee(String inName) {
    // your code goes here
}
```

Testa att lägga till följande möjliga sätt att anropa superklassens konstruktor:

super.GreatApe(inName); GreatApe(inName); super(inName);

Vilken fungerade, d v s vilken är korrekt?

OBS: Anropet till superklassens konstruktor måste ligga först i konstruktorn.

Inheritance-13

Lägg till följande klassvariabel i Chimpanzee-objekt:

- String Species = “P. troglodytes”;

Inheritance-14

Skapa ett Chimpanzee-objekt i en separat klass, t ex Apan. Skapa ett GreatApe-objekt som pekar på (refererar tlll) Chimpanzee-objektet.

Ungefär så här: Chimpanzee c = new Chimpanzee(“Herr Nilsson”); GreatApe ga = c;

Skriv ut Genus- och Species-variablerna från båda objeketen, typ liksom så här alltså:

```
System.out.println("c.Genus:    " + c.Genus);
System.out.println("c.Species:  " + c.Species);

System.out.println("ga.Genus:   " + ga.Genus);
System.out.println("ga.Species: " + ga.Species);
```

Varför fungerar det inte? Rätta till koden, genom att ta bort en av raderna ... men läs först felmeddelandet och begrunda*.

Varför behövde du ta bort just den raden?

*) “Sjung, läs och glöm, tänk, begråt och begrunda!”

`http://sv.wikisource.org/wiki/Fredmans_epistel_n:o_30`

Inheritance-15

Lägg till en String-variabel (klass-variabel) som heter myClassName i GreatApe och Chimpanzee. Varibeln skall tilldelas namnet på respektve klass.

Skapa ett Chimpanzee-objekt i en separat klass, t ex Apan. Skapa ett GreatApe-objekt som pekar på (refererar tlll) Chimpanzee-objektet.

Ungefär så här: Chimpanzee c = new Chimpanzee(“Herr Nilsson”); GreatApe ga = c;

Skriv ut variablerna mittKlassNamn från båda objeketen, typ liksom så här alltså:

```
System.out.println("c:    " + c.myClassName);
System.out.println("gac:  " + ga.myClassName);
```

Innan du kör programmet (Apan):

- Vad kommer att skrivas ut?
- Betänk att "Referensvariablens (mv eller m) typ avgör vilken variabel som menas". Vad kommer att skrivas ut?

Kör programmet.

Inheritance-16

Lägg till en klass-metod, getClassName, som returnerar en Strings. Metoden skall returnera klassvariabeln myClassName. Metoden skulle kunna se ut så här i de båda klasserna:

```
public String getClassName() {
return myClassName;
}
```

Skapa ett Chimpanzee-objekt i en separat klass, t ex Apan. Skapa ett GreatApe-objekt som pekar på (refererar tlll) Chimpanzee-objektet som ovan.

Skriv ut variablerna mittKlassNamn från båda objeketen, som ovan. Skriv också ut strängen du får tillbaka efter att ha anropat metoden getClassName på c respektive ga.

```
System.out.println("c (variabel):    " + c.myClassName);
System.out.println("c (metod):       " + c.getClassName());
System.out.println("gac (variabel):  " + ga.myClassName);
System.out.println("gac (metod):     " + ga.getClassName());
```

Innan du kör programmet (Apan):

- Vad kommer att skrivas ut?
- Betänk att "Referensvariablens typ avgör vilken variabel som menas". Vad kommer att skrivas ut?
- Betänk också att "Objekets typ avgör vilken metod som menas". Vad kommer att skrivas ut?

Kör programmet.

Vad drar vi för slutsats? Jo, följande:

- "Referensvariablens typ avgör vilken variabel som menas".
- "Objekets typ avgör vilken metod som menas".

Inheritance-17

Klassen Object (ja, den heter så) har en metod som heter getClass(). Denna metod returnerar Class<?>.

Använd denna metod på Chimpanzee-objektet du skapade i någon av övningarna innan.

Det finns en metod att få ut namnet på den klass objektet har/är. Skriv ut klass-namnet på c respektive ga med hjälp av denna metod.

Object: http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html

Class: http://docs.oracle.com/javase/7/docs/api/java/lang/Class.html