Arv och polymorfism

De klasser/filer som hänvisas till hittar du på kurssidan under Dokument/Övningar/Kodskelett.

Uppgift 1, Arv

Studera klassen Person, som definierar några egenskaper för en person, t ex för ett register. Observera att datamedlemmarna är deklarerade `private`.

Du ska skapa en subklass till Person, Employee, som lagrar info en person med anställning. Klassen Employee ska ha de nya datamedlemmarna position (befattning) och salary.

Klassen ska ha en konstruktor där även basklassdatat initieras. Hur gör man det när dessa datamedlemmar är `private`?

Lagg till en metod, raiseSalary, som höjer lönen för den anställde. Lägg också till accessmetoder samt omdefiniera `toString()` så den ger *all* information om en anställd, även basklassens privata data.

Skriv en main-metod som testar all funktionalitet i den nya klassen. Vilka datamedlemmar har klassen Employee? Vilka metoder?

Uppgift 2, Polymorfism och dynamisk bindning

Student och Employee är subklasser till Person. Alla klasserna har en metod toString(), men metoden har olika definitioner i de olika klasserna.

Skapa objekt av var och en av klasserna som du hanterar via referensvariabler av basklasstyp (Person), t ex i en array av Personreferenser.

Anropa toString() från respektive referens i arrayen och kontrollera att det verkligen är rätt version som anropas (d v s att det sker med dynamisk bindning, utifrån objektets typ).

Anropa också någon metod som endast finns för en subklass. Du måste göra anropet från en subklassreferens. Vill du t ex anropa raiseSalary för en anställd som hanteras med en Personreferens, ska du först kontrollera att objektet som refereras är av rätt typ, Employee, med hjälp av operatorn `instanceof`, och därefter göra ett s.k. downcast till rätt referenstyp, t.ex. som nedan

```
Employee e = (Employee) personArr[i];
```

Först därefter kan du göra ett anrop av metoden som är specifik för Employee.

Uppgift 3, Abstrakt klass

Skapa den arvshierarki som beskrivs i följande deluppgifter. I samtliga tre klasser som beskrivs ska en konstruktor skapas samt en implementation av toString.
a) Skapa en abstrakt basklass Vehicle, som representerar ett fordon, med två attribut, antal hjul (heltal) och färg (t.ex. en sträng). Datamedlemmarna får inte vara åtkomliga för någon annan klass, utan enbart för metoder i klassen.
I klassen skall det finnas metoder som returnerar antal hjul och returnerar färg.
Det ska också finnas en abstrakt metod som returnerar körkortsklass (en text).

b) Skapa klasserna Car och Motorcycle som båda ärver från klassen Vehicle.
Klassen Car ska innehålla datamedlemmen dörrantal (heltal) samt metoder som returnerar antal dörrar. Metoden för körkortsklass ska returnera ”B”.
Klassen Motorcycle ska ha en datamedlem (boolean) som anger om det är en lätt eller tung motorcykel. Metoden för körkortsklass ska returnera ”A1” om det är en lätt motorcykel, annars ”A”.
c) Skriv en testklass som skapar en array av typen Vehicle, samt minst två objekt av klasserna Car och Motorcycle. Du skall testa metoderna i klasserna, speciellt att anropa metoden för att avläsa antalet dörrar i klassen bil.