**Arv**

Skapa klasser det kan du. Men kan du förlänga en redan existerande klass?

Arv handlar om just detta. Att förlänga en redan existerande klass. Alla klasser ärver från Object. Det är därför de har ett visst antal fördefinierade metoder och annat som du själv inte skrivit till din klass.

Tänk dig klass A med ett antal metoder och variabler. Klass B ärver från klass A och förlänger därmed klass A med några till metoder, kanske ändrar någon metod för ett mer specifikt syfte etc (mer om det senare i Polymorphism).

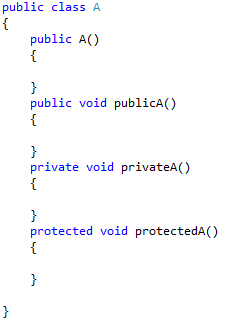
Illustrerat såhär:

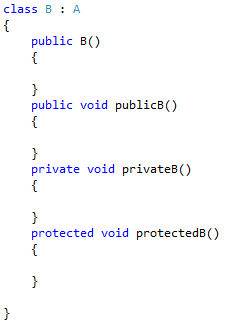
Där klass A är den svarta cirkeln och klass B den gröna. Vi ser att klass B har klass A i sig med även att den har mer än bara klass A.

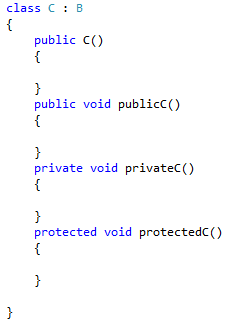
Skulle vi skapa en till klass, klass C som ärver av klass B (Förlänger B)

Klass C är den lila cirkeln. Som illustrerat innehåller då klass C, klass A som förlängts med klass B. alltså allt som klass A och B har i sig (dock är B överlappande A).

*Några exempel på arv:*







Skriv/skapa nu en klass i samma paket/namespace.

Instansiera objekt av klass A, klass B och Klass C och undersök vad du kan se från de olika objekten. Vad du kommer åt.

**Uppgifter:**

1. **Skriv klasserna ovan men ändra dem gärna så de skriver ut text någon stans. Ex: ”protected för A” i protected void protectedA(). Testa sedan vilka metoder som finns och vilka texter som faktiskt kommer upp när main metoden körs.**  
   I objektet A kommer man åt endast Public A.  
   I objektet B kommer man åt public A samt B för att B refererar till A.  
   I objektet C kommer man åt Public A, B, C för att klassen C refererar till klassen B vilket refererar till klassen A.
2. **Gör detta i klassen Program samt i de olika klasserna A, B och C. Vad syns och vad funkar?**  
   1. Det beror på om man skapar ett objekt av en klass eller inte  
   2. När man skapar objektet, så beror det på vilken av klasserna man skapar objektet utav.  
   Om vi nu säger att vi skapar ett objekt av korresponderande klass så kommer det se ut såhär:  
   Man kommer åt alla funktioner från sin egen klass, man kommer även åt subklassernas protected samt public.
3. **Vilka funkar resp funkar inte?**
   1. **A a = new A();**  
      Funkar, man skapar ett nytt objekt av klassen.
   2. **A a = new B();**  
      Funkar, Det funkar för att A innehåller färre attribut än vad b innehåller och därmed så blir det inge överflöd. Attributen från b sparas även.
   3. **A a = new C();**Funkar.
   4. **B b = new A();**  
      Funkar ej. B innehåller bade a fast mer. Om man skulle göra såhär så skulle alla atribut som B har i överflåde skulle resulteras till null.
   5. **B b = new B();**  
      Funkar.
   6. **B b = new C();**  
      Funkar.
   7. **C c = new A();**  
      Funkar ej.
   8. **C c = new B();**  
      Funkar ej.
   9. **C c = new C();**  
      Funkar.
4. **Vad ser du i de olika objekten som faktiskt funkar? Gör en tabell.  
   Objekten blir detsamma. C c = new**

Man kommer alltid kunna få ut public metoden från det objektet som faktiskt tillkommer i slutändan. MEn man man nu skulle skapa ett A a = new C(); Så skulle man bara komma åt public A. Men man kan parsa objektet ”a” för att komma åt metoder från både classen C och B. Classen B kommer man åt för att C ärver från klassen B.

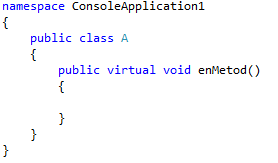
Virtual och override

När du förlänger en klass kanske du vill ändra på en av metoderna som följer med eller skapa en metod med samma namn bara det att du ändrar den lite för just det objektet.

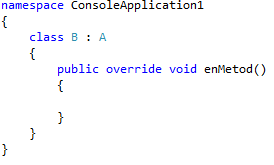
C#:

Kodmässigt gör du detta genom att deklarera de metoder som kan ändras till virtual och de metoder som skall/kan ändras till override.

I klass A:



I klass B:



Notis: Du kan även göra metoden i B virtual för att sedan kunna skapa overrides i klasser som ärver från den.

Java:

Du döper metoden du vi ändra till samma som den i superklassen. Om du inte vill att någon skall kunna ärva och ändra en metod i en klass deklarerar du den med final.

Public final void aMethod()

# Uppgifter:

Gör övningarna för förra delen men skriv innan till en metod för alla klasser som heter samma sak. I C# får du lägga till lite extra som nämnts ovan. Frågan är: Vilken metod körs? För att göra det mer komplicerat, komplicera det själv. Används olika variabler i klasserna med mera och använd dem i metoderna.

Skriv en kortare sammanfattning om vad du kommer fram till efter alla dessa uppgifter och övningar. Skriv strukturerat och noggrant. Gärna med exempel. Försök även svara på frågan; vad betyder allt och när används det och varför?  
  
**Svar:**  
När man skapar objektet som A ab = new B(); Så är det Override metoden som B som körs. Objektet skapas som ett objekt utav A men får spcifikationer från B. Annars om man skapar ett objekt som  
C c = new C(); så kallar du fortfarande på samma metod men resultatet blir annorlunda.   
  
Man skulle kunna använde detta i ett ”inventory” system. Låt säga att att vi har en grunklass som heter ”item” som har en metod som heter use som är virtual, sjävla metoden Use i klassen Item kommer inte göra något.  
Sedan skapar vi 2 subklasser till item som heter te.x Potion, Equipment.  
I varje klass skapar vi en ovveride till Use metoden. Där potion te.x ökar spelarens hälsa. Medans use på Equipment skulle göra att spelaren sätter på sig föremålet.  
Detta var bara ett exempel på hur man skulle kunna använda virtual samt override.  
Att använda override kan göra att programering koden ser finare ut samt att den blir lättare att läsa och det går snababre att skriva det.