**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).

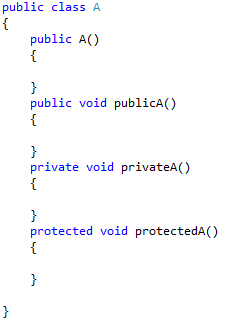
Illustrerat såhär:

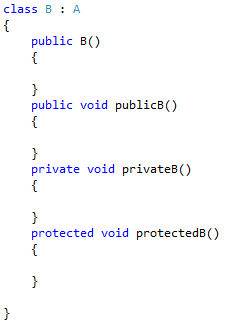
Där klass A är den svarta cirkeln och klass B den gröna. Vi ser att klass B har klass A i sig med även att den har mer än bara klass A.

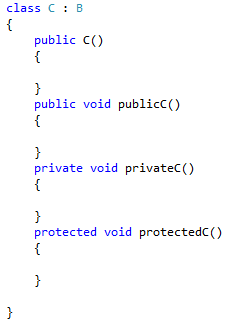
Skulle vi skapa en till klass, klass C som ärver av klass B (Förlänger B)

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. Man kommer alltid åt public. Man kommer bara åt private i sin klass. Man kommer åt protected i sin klass och klasser som ärver från den. Det kommer inte upp någon text eftersom man inte har skrivet ut något i klassen Program.
2. Gör detta i klassen Program samt i de olika klasserna A, B och C. Vad syns och vad funkar? De som man kommer åt i uppgift 1 syns om man skriver ut de i klassen Program.
3. Vilka funkar resp funkar inte?
   1. A a = new A(); Funkar
   2. A a = new B(); Funkar
   3. A a = new C(); Funkar
   4. B b = new A(); Funkar inte
   5. B b = new B(); Funkar
   6. B b = new C(); Funkar
   7. C c = new A(); Funkar inte
   8. C c = new B(); Funkar inte
   9. C c = new C(); Funkar
4. Vad ser du i de olika objekten som faktiskt funkar? Gör en tabell.

|  |  |  |  |
| --- | --- | --- | --- |
| A a = new A(); | Public A |  |  |
| A a = new B(); | Public A |  |  |
| A a = new C(); | Public A |  |  |
| B b = new B(); | Public A | Public B |  |
| B b = new C(); | Public A | Public B |  |
| C c = new C(); | Public A | Public B | Public C |

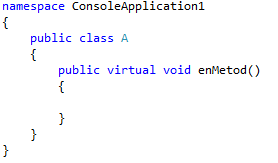
Virtual och override

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.

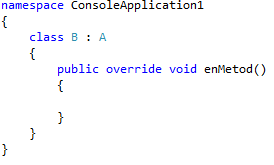
C#:

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?