

Ragnar Geir Brynjólfsson

# Forritun í C#



## 2. hluti

Útgáfa 1  
Heimili og menntir ehf  
Júlí 2005

**Forritun í C# 2. hluti.**

1. útgáfa, Heimili og menntir ehf. Júlí 2005.

© Ragnar Geir Brynjólfsson.

Öll réttindi áskilin. Bók þessa má ekki afrita með neinum hætti, að hluta eða í heild án skriflegs leyfis höfundar og útgefanda.

Umbrot, frágangur og prentun:

*Forsíðumynd: Úr forritunartíma í FSu á vorönn 2005.*



*Ragnar Geir Brynjólfsson*

**Um höfundinn:**

Ragnar Geir Brynjólfsson er kerfisstjóri og forritunarkennari við Fjölbrotaskóla Suðurlands á Selfossi þar sem hann hefur starfað síðan 1989. Hann útskrifaðist með BS próf í tölvunarfræði og próf í kennsluréttindum frá Háskóla Íslands árið 1989. Hann hefur áður samið eftirtaldar kennslubækur:

- Windows 3.1 : Word 2.0 ritvinnsla, Access gagnagrunnur : Æfingar fyrir byrjendur. Höfundur gaf út 1994, 104 s. : teikningar, skjámyndir.
- Windows 3.11 : Tölvugrunnur : Æfingar fyrir byrjendur. Höfundur gaf út 1995, 1996 og 1997. 46 s. : skjámyndir.
- Forritun í Java : Heimili og menntir gaf út í einni bók 2003 (315 síður) og í tveim bókum I. hluta og II. hluta 2004.
- Forritun í C# 1. hluti: Heimili og menntir gaf út 2005. 145 síður.

## Efnisyfirlit

<b>Inngangur</b> .....	<b>6</b>
Val forritunarmáls .....	6
Um lestur bókarinnar .....	7
Stuðningur við útgáfuna .....	8
Íslenskun .....	9
<b>T203-1. Grunnhugtök hlutbundinnar forritunar</b>	
Hlutbundna hugmyndafræðin .....	10
Aðferðir—hegðun .....	17
Hjúpun .....	18
Stöðupróf T203SP1 .....	22
Forritunarverkefni .....	24-31
<b>T203-2. Fleiri hugtök hlutbundinnar forritunar</b>	
Fjölbinding aðferða .....	32
Meira um stika .....	34
Staðvær gögn og gildissvið .....	36
Frátekna orðið this .....	39
Erfðir .....	40
Yfirtaka aðferða .....	43
Klasahönnun með UML .....	44
Hugrænir klasar .....	44
Einfaldar erfðir .....	46
Viðmót .....	47
Fjölbreyttni .....	49
Stöðupróf T203SP2 .....	51
Forritunarverkefni .....	54-59
<b>T203-3. Frábrigði, villur straumar og skrár</b>	
Frábrigði .....	60
Frávikum kastað .....	65
Straumar og skrár .....	66
Innlestur frá lyklaborðinu .....	67
Skrár .....	69
Skrár með beinum aðgangi .....	72
Stöðupróf T203SP3 .....	74
Forritunarverkefni .....	77-81
<b>T203-4. Gagnaskipan, safnklasar og pakkar</b>	
Stafli .....	82
Biðröð .....	83
Tengdur listi .....	84
Tré .....	85
Mengi .....	86
C# safnklasar .....	87
Útfærsla stafla og biðraðar .....	88
Útfærsla mengja .....	92
Stöðupróf T203SP4 .....	92
Forritunarverkefni .....	95-99

---

<b>T203-5. Myndræn notendaskil .....</b>	<b>100</b>
Tölvugrafík .....	100
Algeng skráasnið .....	102
Inngangur að litafræði .....	104
Grafísk notendaskil C# .....	105
Eiginleikar forms .....	109
Teiknað á form .....	110
Atburðir .....	114
Textareitur og merkimiði .....	116
Myndir .....	118
Valmyndir .....	120
Stöðupróf T203SP5 .....	122
Forritunarverkefni .....	124-128
<b>T203-6. Röðun—netsamskipti .....</b>	<b>130</b>
Fleiri röðunaralgrím .....	133
Netsamskipti .....	134
TCP/IP og UDP/IP .....	135
Nöfnum og IP tölum flett upp .....	136
Vefsíðukóði sóttur .....	139
TCP tengill opnaður og lesinn .....	143
Stöðupróf T203SP6 .....	145
Forritunarverkefni .....	146-150
<b>Hugtakasafn .....</b>	<b>152-159</b>
<b>Heimildaskrá .....</b>	<b>160-161</b>



## Inngangur

Bók þessi er fyrst og fremst skrifuð sem námsefni fyrir framhaldsskólanema sem leggja stund á nám í forritun en hún getur vonandi nýst öllum sem áhuga hafa á forritun og vilja tileinka sér hana. Við gerð bókarinnar var því tekið mið af TÖL áföngum framhaldsskólans eins og þeir eru settir fram í Aðalnámsskrá framhaldsskóla sem gefin var út 1999. Einnig var tekið tillit til þess að þegar er komið út efni sem að hluta til kemur til móts við þessa áfanga og reynt að láta skörunina verða sem minnsta. Hér er átt við bókina „*Kennslubók í tölvufræði fyrir framhaldsskóla*“ eftir Atla Harðarson. Sú bók reifar sögu tölvufræðinnar og helstu grundvallaratriði hennar og er gert ráð fyrir að hún sé lesin meðfram þessu efni sem fyrst og fremst er ætlað að koma til móts við forritunarþátt áfangamarkmiðanna.

### Val forritunarmáls

Forritunarmálið C# var valið af ýmsum ástæðum. Í fyrsta lagi er mikilvægt að málið geti nýst til innleiðslu á grunnhugtökum svo sem breytum, tögum og flæði aðgerða. Í öðru lagi æskilegt að sem best samfella verði milli TÖL áfanganna 103, 203 og 303 svo ekki tapist tími vegna innleiðingu nýs forritunarmáls. Notkun forritunarmáls í þrem áföngum gerir kröfu um fjölhæfni, að það nýtist í flóknu umhverfi og að það sé hlutbundið. Bóklegt forritunarnám í framhaldsskóla snýst um að byggja upp hugtaka- og færni grunn sem reiknað er með að byggt sé ofan á í framhaldsnámi. Valið stóð því fyrst og fremst milli þriggja forritunarmála sem öll komu til greina. C++, Java og C# (lesist síð eða sí-sharp). Java og C# eru hlutbundin frá grunni og málskipan þeirra og grunnforsendur eru líkar. Þekking á öðru málinu mun því í mörgum tilfellum nýtast í hinu, sérstaklega hvað varðar grunnatriði. C# er orðið til því Java skorti að sumra mati sérhæfingu fyrir Microsoft kerfi. C# er nýrra en Java og nýtur mikilla vinsælda meðal forritara um þessar mundir.

Þessar forsendur hafa orðið til þess að C# málið varð fyrir valinu. Þau ár sem ég hef kennt forritun hef ég stundum heyrt að forritunarmálið sem kennt er sé úrelt, hvergi notað, eða þá að það sé óþarflega flókið eða of torskilið fyrir byrjendakennslu. Ég ráðlegg nemendum sem heyra athugasemdir af þessum toga að taka þeim með fyrirvara. Að fullyrða slíkt við nemendur sem oft á tíðum eiga í baráttu við sjálfa sig við að halda sér að námi er mikill ábyrgðarhluti. Hafa ber í huga að kunnátta og færni yfirferist oftast vel milli forritunarmála og sér í lagi á það við ef málin eru lík.

## Um lestur

Lesið 10 -15 mínútur í einu, takið þá 5 mínútna hlé, gerið eitthvað skemmtilegt og byrjið svo aftur. Hæfileiki heilans til einbeitingar nær ekki yfir lengri tíma, en endurnýjast eftir hvíldina. Þegar þetta vinnulag er viðhaft verður lesturinn skemmtilegri. Lesið fyrst lauslega **og notið áherslupenna** til að merkja við mikilvæg atriði. Lesið svo aftur, ítarlega og vel og reynið að skilja öll atriði. Lesið svo í þriðja skiptið og reynið að ná aðalatriðunum og rifja upp hvað þið lærðuð. Notið spássíurnar til að skrifa strax niður athugasemdir og spurningar sem vakna við lesturinn. Komið svo



Áherslupennar— munið eftir þeim.

spurningunum á framfæri ef þið finnið svörin ekki í bókinni. Nám í forritun á ýmsar hliðstæður við tungumálanám. Verkefnið snýst fyrst og fremst um að læra orð og hugtök og hvernig þau tengjast saman. Áður en hægt er að forrita er nauðsynlegt að hafa numið mörg orð og hugtök. Verið því þolinmóð og gefið ykkur tíma til að læra grunnatriðin vel. Ef þið eruð orðin þreytt leggið þá bókina frá ykkur. Hugtökin verða þarna á morgun líka og þá er hægt að gera aðra atrennu að þeim.

## Sýnidæmi og leyst verkefni

Sýnidæmi bókarinnar og leyst forritunarverkefni má nálgast á vefnum á vefsvæðinu **<http://www.home.is/forrituncsharp/>**. Ég hvet ykkur til að þýða og keyra sýnidæmin og leystu forritunarverkefnin til að ná enn betri skilningi á þeim. Þetta getur þurft að gera aftur og aftur því endurtekning og upprifjun gerir meira fyrir nemandann en nokkur kennari eða kennslubók getur gert.

## Nauðsynlegur hugbúnaður

Til að forrita í C# nægir textaritpór auk .NET Framework frá Microsoft. Eigendur Microsoft kerfa geta sótt þennan hugbúnað inn á <http://windowsupdate.microsoft.com>. Smelltt á „Select optional software updates.“ Eigendur Linux, OSX, Solaris eða BSD kerfa geta sótt Mono vinnuhverfið fyrir C# inn á <http://www.monoproject.com/>.

## Stuðningur við útgáfuna—reikningsnúmer

Gerð kennslubóka á íslensku er ekki ábatasöm atvinnugrein, sér í lagi ef markaður bókanna eru fámennir áfangar framhaldsskólans, áfangar á borð við forritunaráfanga. Forritunaráfangar eru einungis í boði sem kjörsviðsáfangar fyrir fjölmennar brautir á borð við náttúrufræðibraut, viðskiptabraut, málabraut eða félagsfræðibraut. Án dyggilegs stuðnings Menntamálaráðuneytisins væri vart mögulegt að gefa út kennslubækur fyrir þessa og aðra fámenna áfanga. Ráðuneytinu eru færðar þakkir fyrir stuðning sinn við þetta verkefni en jafnframt er leitað eftir frekari stuðningi allra sem nota bókina. Greiða má inn á tékkareikning (höfuðbók 26), reikningsnúmer 38816 í KBbanka — aðalbanka, bankanúmer 301. Kennitala: 310361-4539. Væntanlegum styrktaraðilum eru færðar þakkir fyrir framlag sitt.

## Íslenskun

Ég hef farið þá leiðina við gerð þessa námsefnis að nota fyrst og fremst íslenskuð orð og hugtök enda hægt um vik í mörgum tilfellum að sækja þau í orðaforða Tölvuorðasafnsins, en það byggir á starfi orðanefndar Skýrslutæknifélags Íslands. Heiti hugtakanna eru jafnframt kynnt á ensku innan sviga þegar þau koma fyrst fyrir í textanum og stundum oftar. Meirihluti nemenda hefur verið ánægður með þetta fyrirkomulag.

Selfossi í júní 2005.

Ragnar Geir Brynjólfsson.

## T203-1. Grunnhugtök hlutbundinnar forritunar

### Hlutbundna hugmyndafræðin

Af hverju þurfum við hlutbundna forritun? Hvað hefur hlutbundið forritunarmál fram yfir venjulegt forritunarmál þar sem breytur og aðferðir þurfa ekki að tilheyra tilteknum klasa? Þetta skýrist ef við veltum fyrir okkur vanda óhlutbundinnar forritunar. Þegar forrit í óhlutbundnu forritunarmáli verður stórt verður það flókið og erfitt í viðhaldi. Áður en hlutbundin mál festu sig í sessi var leitað lausna með *föllum* (functions) og *stefjum* (procedures) sem líkja má við föll í stærðfræði og *einingum* (modules), sem þjappa á sama stað endurteknum skipunum í forriti. Þetta var kallað mótuð forritun (structured programming). En þegar forritin stækkuðu ennþá dugðu þessi ráð ekki lengur til að halda utan um flókin forrit. Oft var erfitt að hanna og þróa óhlutbundin forrit því aðal byggingareiningar þess, föllin og *gagnagrindurnar* (data structures) voru ekki heppileg til líkanagerðar. Það eru hlutir og klasar í hlutbundnu forritunarmáli aftur á móti.

Í TÖL103 lærðum við að C# vinnur með upplýsingar á tvennan hátt. Sem *frumgögn* (value types) eða sem *hluti* (objects, reference types). Hlutir eru notaðir til að lýsa samsettum gögnum sem mynda heild ásamt aðferðum sem hægt er að beita á hlutina. T.d. eins og bíl í bílaforriti og því sem hann á að geta gert, t.d. tekið af stað eða stöðvað.

Hlutur tilheyrir alltaf tilteknum *klasa* (class) og segja má að klasinn sé gagnatag hlutarins. Í klasanum eru ennfremur skilgreindar breytur eða aðferðir sem hægt er að kalla á. C# er hlutbundið frá grunni sem þýðir að allar aðferðir í málinu tilheyra einhverjum klasa. Klasanum má því líkja við uppdrátt, fyrirmæli eða skapalón af því hvernig hluturinn eigi að vera. Hægt er að hugsa sér að ef

### Hugtök

#### **Fall:**

*Safn skipana í óhlutbundnu forritunarmáli sem leysa afmarkað verkefni og skila frá sér einni breytu eða vistfangi.*

#### **Stefja:**

*Safn skipana í óhlutbundnu forritunarmáli sem leysa afmarkað verkefni og geta sent frá sér mörg skilagildi gegnum stika.*

#### **Eining:**

*Notað til að lýsa samantekt falla og stefja sem leysa tiltekið verkefni.*

#### **Gagnagrind/**

#### **Gagnaskipan:**

*Byggingareiningar forrits sem notaðar eru til að auðvelda úrvinnslu gagna. Dæmi: frumgögn, strengir, fylki.*

#### **Frumgögn (value type):**

*Gagnagerð sem er innbyggð í málið, t.d. int, float, char, bool.*

#### **Hlutur:**

*Sjálfstæð eining í forriti sem hægt er að vinna með og breyta. Eitt eintak af tilteknum klasa, inniheldur eigin gögn og samsvarandi aðferðir.*

#### **Klasi:**

*Safn skilgreininga sem lýsir því hvernig hlutir eigi að vera.*

## Hugtök

### Tilvik:

Íslenskun enska orðsins 'instance.' Annað orð yfir hlut í hlutbundnu forritunarmáli.

klasi sé smákökuuppskrift þá séu hlutirnir kökurnar sem eru bakaðar samkvæmt uppskriftinni. Við borðum ekki uppskriftina, né leggjum hana á borð, en í uppskriftinni felast samt nákvæmar leiðbeiningar um hvernig útlit kakanna eigi að vera sem og um ýmsa eiginleika þeirra svo sem bragð. Annað orð er notað um hlutina en það er orðið *tilvik* (instance) sem er rökrétt því þá er hægt að tala um ákveðin tilvik klasans. Skoðum fyrst sýnidæmi T203Sd1\_1 til að glöggva okkur á

```

**
*****
Skrá: T203Sd1_1.cs
Höfundur: Ragnar Geir Brynjólfsson.
Dagsetning: 24.05.2005.
Skilgreining klasa og eintökun sýnd
*****
*/
//Klasaskilgreining
class A {}
class T203Sd1_1 {
    static void Main() {
        A tilvik1 = new A();
    }
}

Úttak T203Sd1_1 er ekkert. Forritið þýðist með einni viðvörðun:
[ragnar@forritun tol203]$ mcs T203Sd1_1.cs
T203Sd1_1.cs(13) warning CS0219: The variable 'tilvik1' is
assigned but its value is never used
Compilation succeeded - 1 warning(s)

```

því hvernig klasaskilgreining er sett upp.

Í T203Sd1\_1 er sýnd málfræði klasaskilgreiningar. Klasinn A, sem gerir ekkert og á ekkert er skilgreindur með orðunum `class A { }`. Ekkert er innan bálksins (slaufusvigaparsins) á eftir A og því á klasinn hvorki gögn né aðferðir. Þetta er samt málfræðilega rétt klasaskilgreining og því er hægt að eintaka (búa til) hlut af A með línunni:

```
A tilvik1 = new A();
```

Reyndar eru tvær klasaskilgreiningar í þessari skrá. Fyrst er klasinn A skilgreindur og síðan er klasinn T203Sd1\_1 skilgreindur. T203Sd1\_1 er sérstakur því hann á Main() aðferð, en ekki er

nauðsynlegt að allir klasar eigi slíka aðferð, heldur aðeins þeir klasar sem ætlast er til að Common Language Runtime (CLR) túlkurinn keyri. Þetta forrit er því þýtt og keyrt með skipununum:

```
C:\>mcs T203Sd1_1.cs
C:\>T203Sd1_1.exe
```

í Windows en

```
$mcs T203Sd1_1.cs
$mono T203Sd1_1.exe
```

í mono kerfinu í Linux.  
(Sjá <http://www.mono-project.com/> )

Ekkert úttak birtist því engar úttaksskipanir eru gefnar í T203Sd1\_1.

Ekki er skilyrði að klasar séu í sömu skrá og klasarnir sem eintaka þá eins og í T203Sd1\_1. Hægt hefði verið að búa til sérstaka skrá með klasanum A án þess að hafa klasann T203Sd1\_1 með í sömu skrá. Þá er skráin sem inniheldur A höfð með í þýðingunni. Í T203Sd1\_2 er gengið skrefi lengra í því að gera ekkert en í T203Sd1\_2. Hérna er aðeins ein klasaskilgreining og tóm Main aðferð.

```
**
*****
Skrá: T203Sd1_2.cs
Höfundur: Ragnar Geir Brynjólfsson.
Dagsetning: 24.05.2005.
Skilgreining klasa sýnd
í sinni einföldustu mynd
*****
*/
//Klasaskilgreining
class B {
    static void Main() {
    }
}

Úttak T203Sd1_2 er ekkert:
[ragnar@forritun tol203]$ mcs T203Sd1_2.cs
[ragnar@forritun tol203]$ mono T203Sd1_2.exe
[ragnar@forritun tol203]$
```

## Hugtök

### **Assembly:**

Hið þýdda mál C# gengur undir þessu nafni. Þetta mál líkist vélamáli en það þarfnast CLR túlks til að keyrast.

### **Eiginleikar:**

Annað orð yfir breytur tiltekins klasa.

### **Hegðun:**

Annað orð yfir aðferðir tiltekins klasa.

### **Aðferð:**

Safn skipana sem ætlað er að leysa afmarkað verkefni.

### **Kyrrleg breyta/aðferð:**

Breyta/Aðferð sem þarfnast ekki eintekningar, þ.e. að hlutir séu búnir til að þær séu vaktar.

### **Tilviksbreyta/aðferð:**

Breyta/aðferð sem þarfnast eintekningar, þ.e. að hlutur sé búinn til, til að hægt sé að nota þær. Þær tilheyra þá þessum tiltekna hlut.

### **Klasabreyta/aðferð:**

Sama og Kyrrleg aðferð/breyta.

### **Smiður:**

Aðferð sem heitir sama og klasinn sem hún tilheyrir. Hefur ekkert skilagildi. Ætluð til að eintaka (búa til) hluti.

Höldum nú áfram með klasagerðina og fikrum okkur frá því að gera ekkert til þess að búa til klasa sem lýsir bíl í T203Sd1\_3. Klasinn `Bill` samanstendur af tveim megingerðum upplýsinga: *Eiginleikum* (properties) sem breytur sýna og *hegðun* (behaviour) sem *aðferðirnar* (methods) sýna. Aðferðir má þekkja á því að þær heita eitthvað og á eftir þeim kemur slaufusvigabálkur (`{ }`) sem oftast inniheldur skipanir.

Klasinn `Bill` í T203Sd3\_1 á *kyrrlegu* (static) breytuna `fjoldi_bila`. Hann á einnig *tilviksbreyturnar* (*instance variables*) `Gerð`, `Númer`, `Stada`, `Argerð`, `Hradi` og `Ekinn`. Munurinn á kyrrlegum breytum og tilviksbreytum er sá að tilviksbreytur eru bara sýnilegar og nothæfar í hverju einstöku tilviki (hlut) fyrir sig en kyrrlegar breytur eru sýnilegar í öllum tilvikum (hlutum) tiltekins klasa. Því er einnig algengt að kalla þær *klasabreytur* (class variables). Á eftir upptalningu breytanna kemur svokallaður *smiður* (constructor). Smiðir þekkjast á því að þeir eru aðferðir með engu skilagildi sem heita það sama og klasinn. Í þessu tilfalli er smiðurinn `Bill` í klasanum. Hann gefur tilviksbreytunum gildi auk þess að hækka `fjoldi_bila` um einn. Lítum aðeins nánar á hvað gerist þegar kallað er á `new`:

Kallað er á smið fyrir klasann sem tekur frá minnissvæði og eintakar hlutinn.

Tilviksbreytur hlutarins sem ekki fá sérstök gildi eru núllstilltar, þ.e. tölubreytur fá 0, hlutir fá `null` (þ.e. vísa ekki á minnissvæði), `bool` breytur fá `false` og `char` breytur fá `'\0'`

Tvö atriði greina smiði frá venjulegum aðferðum: Þeir heita sama og klasinn en það geta aðferðir aldrei. Þeir eru ekki af neinu tilteknu tagi, ekki einu sinni `void` og því geta þeir ekki skilað gildi með `return` setningu.

Á eftir smiðunum koma `void` aðferðirnar `keyra()`, `stansa()`, `skrifa_upplys()` og `skrifa_bilafj()`. Það að aðferðirnar eru `void` þýðir að þær skila ekki neinu gildi til annarra að-

```

/**
*****
Skrá: T203Sd1_3.cs
Höfundur: Ragnar Geir Brynjólfsson.
Dagsetning: 24.05.2005.
Klasinn Bill skilgreindur
*****
*/using System;
class Bill {
public static int fjoldi_bila;
public string Tegund;
string Numer,Stada;
int Argerd,Hradi,Ekinn;
public Bill(string n,int a,int e) {
Numer=n;Argerd=a;Ekinn=e;
Console.WriteLine("Bill af gerð "
+ Tegund + " búinn til.");
}
public void keyra(int h) {
Hradi = h;
Stada = "Bíll " + Numer + " er núna á keyrslu"
+ " á hraða " + Hradi + ".";
Console.WriteLine(Stada);
}
public void stansa(int km) {
Ekinn+=km;
Stada = "Bíll " + Numer + " stansar.";
Console.WriteLine(Stada);
}
public void skrifa_uppl () {
Console.WriteLine("\nTegund: " + Tegund
+ "\nNúmer: " + Numer + "\nEr ekinn: "
+ Ekinn + " kílómetra.");
}
}
}

```

***Klasinn er notaður í öðrum forritum.***

ferða heldur gera eitthvað sjálfar, svo sem gefa breytum gildi eða skrifa eitthvað út. Lítum nú á T203Sd1\_4 sem nýttir bílaklasann, býr til eintök af honum og notar aðferðirnar til að lýsa hegðun eintakanna.

Í T203Sd1\_4 er ekki nauðsynlegt að sækja bílaklasann né hafa hann í sömu skrá því hægt er að þýða hann um leið og T203Sd1\_4 er þýtt svona:

```
c:\>csc T203Sd1_4.cs T203Sd1_3.cs
```

í Windows og

```
mcs T203Sd1_4.cs T203Sd1_3.cs
```

## Hugtök

### Stiki:

*Breyta sem send er til aðferðar í gegnum haus hennar, þ.e. hún er staðsett innan sviganna sem koma á eftir heiti aðferðarinnar.*

Í mono kerfinu fyrir Linux. Búinn er til einn bíll **b** með smiðnum. Smiðurinn telur kyrrlegu breytuna `fjoldi_bila` upp um einn. Bíllinn er látinn taka af stað með `keyra()` aðferðinni og tilviksbreytunni `Hradi` gefið gildi. Loks er bíllinn látinn stansa eftir ákveðinn ekinn kílómetrafjöldi og tilviksbreytu, `Ekinn` gefið gildi um leið. Því næst eru upplýsingar skrifaðar út og að lokum er gildi kyrrlegu breytunnar `fjoldi_bila` skrifað. Takið eftir hvernig tilviksgögnin og aðferðirnar eru háð því að kallað sé á þær með nafni hlutarins sem þær tilheyra og punktvirkja og hversu ólíkt það er og þegar gildi kyrrlegu breytunnar `fjoldi_bila` er skrifað út. Í tilfelli kyrrlegra gagna nægir að skrifa

```
/**
*****
Skrá: T203Sd1_4.cs
Höfundur: Ragnar Geir Brynjólfsson.
Dagsetning: 24.05.2005.
Klasinn Bill skilgreindur
*****
*/using System;
class T203Sd1_4 {
    public static void Main() {
        Bill b = new Bill("AD 123",2003,0);
        b.Tegund = "Bjalla";
        Bill.fjoldi_bila++;
        b.keyra(90);
        b.stansa(55);
        b.skrifa_uppl();
        Console.WriteLine("Bílafjöldinn er: " + Bill.fjoldi_bila);
    }
}
```

### Úttak T203Sd1\_4 er:

```
[ragnar@forritun tol203]$ mcs T203Sd1_4.cs T203Sd1_3.cs
[ragnar@forritun tol203]$ mono T203Sd1_4.exe
Bíll af gerð búinn til.
Bíll AD 123 er núna á keyrslu á hraða 90.
Bíll AD 123 stansar.

Tegund: Bjalla
Númer: AD 123
Er ekinn: 55 kílómetra.
Bílafjöldinn er: 1
[ragnar@forritun tol203]$
```

---

heiti klasans ásamt punktvirkja fyrir framan.

Hægt er að hugsa sér forritagerð á fleiri en einn hátt. Ein aðferð er að líta svo á að forrit sér runa skipana sem framkvæmdar eru hver á eftir annarri. Þessi aðferð er venjulega kölluð á ensku „procedural programming“. Áður fyrr stigu menn sín fyrstu forritunarskref í óhlutbundnum málum þar sem þetta verklag var viðhaft. Eitt af því fyrsta sem forritari lærir í þessum málum er hvernig brjóta megi vandamál niður í afmarkaða hluta sem hver um sig inniheldur fáar skipanir.

Hlutbundin forritun horfir á málið frá öðrum sjónarhóli. Lausnin er hönnuð með tilliti til verkefnisins fremur en hvernig best sé að leysa það í tölvu. Hlutbundið forrit er mengi hluta sem leysa tiltekið verkefni. Hver hlutur er sjálfstæð eining sem hefur samskipti við aðra hluti á sérstakan hátt. Þetta gerir að verkum að í tiltölulega stuttum forritum er hægt að setja upp líkön, að vísu frumstæð eins og í tilfelli bílasklasans en líkön samt.

Rennum aðeins yfir þessi atriði aftur:

*Tilviksbreyta* (instance variable) geymir eiginleika ákveðins tiltekins hlutar. Þessir eiginleikar geta verið og eru oftast mismunandi milli einstakra tilvika/hluta, þ.e. einstakra bíla í sýnidæmunum á undan. Hægt er að gefa tilviksbreytum gildi þegar hlutur er búinn til eða *eintakaður* (instantiated) og þessi gildi geta verið þau sömu á meðan hluturinn er til. Það er einnig hægt að breyta þeim meðan á keyrslu forritsins stendur.

Kyrrleg breyta eða *klasabreyta* (static variable - class variable) geymir eiginleika alls klasans og allra tilvika hans (hluta). Aðeins eitt gildi er geymt fyrir hverja klasabreytu án tillits til hversu mörg tilvik (hlutir) eru til af klasanum. Dæmi um klasabreytu í `Bill` klasanum er breytan sem inniheldur upplýsingar um heildarfjölda bíla.

```
public static int fjoldi_bila;
```

## Hugtök

### Aðferð:

Safn skipana sem ætlað er að leysa afmarkað verkefni.

### Eintökun:

Þýðing á 'instantiation'. Notað yfir það þegar hlutur er búinn til.

### Kyrrleg breyta/aðferð:

Breyta/Aðferð sem þarfnast ekki eintekningar, þ.e. að hlutir séu búnir til að þær séu vaktar.

### Tilviksbreyta/aðferð:

Breyta/aðferð sem þarfnast eintekningar, þ.e. að hlutur sé búinn til, til að hægt sé að nota þær. Þær tilheyra þá þessum tiltekna hlut.

### Klasabreyta/aðferð:

Sama og Kyrrleg aðferð/breyta.

Ef tilviksbreyta geymdi bílafjöldann þá býður það upp á vandamál á borð við að hver hlutur hefði sérstakt gildi í þessari breytu. Klasabreytan `fjoldi_bila` gerir að verkum að þetta er ekki vandamál því hún er sýnileg úr öllum hlutunum og allir hlutirnir geta breytt henni. Klasabreytur eru þannig notaðar til samskipta milli hluta í sama klasa eða til að hlutir geti fylgst með upplýsingum um ástand mála.

## Aðferðir - hegðun

Hegðun vísar til þess sem klasi eða hlutur geta gert við sig sjálfa eða aðra hluti. Hegðun getur verið að breyta eiginleikum hlutar, tilviksbreytunum, taka á móti upplýsingum frá öðrum hlutum og senda tilkynningar til hluta og biðja þá að gera eitthvað. Bílarnir hafa t.d. þá hegðun að skrifa út hvað þeir eru að gera. Hegðun klasa er útfærð með *aðferðum* (methods) og þessi fyrrnefnda hegðun bílaklasans er útfærð í aðferðunum `keyra()`, `stansa()` og `skrifa_uppl()`.

Aðferðir eru mengi skyldra setninga í klasa sem leysa afmarkað verkefni. Aðferðirnar eru notaðar á svipaðan hátt og föll og undirforrit í aðgerðamálunum. Hlutir hafa samskipti sín á milli með því að nota aðferðir. Klasi eða hlutur getur kallað á aðferðir í öðrum klasa eða hlut af mörgum ástæðu-um t.d. til að :

Tilkynna breytingu á öðrum hlut.

Til að segja hlutnum að breyta einhverju hjá sér.

Til að biðja annan hlut að framkvæma eitthvað.

Til dæmis væri hægt að útfæra Nanóvélnar frekar þannig að þær nýttu aðferðir til að skiptast á staðsetningum sínum og forðast árekstra.

Alveg á sama hátt og það eru tilviksbreytur og klasabreytur þá eru til tilviksaðferðir og klasaaðferðir. *Tilviksaðferðir* (Instance methods) eru svo al-

gengar að þær eru bara kallaðar aðferðir (methods). Aðferðirnar í klasanum `Bill` eru tilviksaðferðir því orðið `static` stendur *ekki* fyrir framan þær. Tilviksaðferðir eru notaðar þegar verið er að vinna með tilviksbreytur hlutanna. Klasaaðferðir vinna með klasann í heild sinni. Orðið `static` fyrir framan aðferðina gerir hana að klasaaðferð. `Main()` aðferðin er því klasaaðferð. Ólíkt tilviksaðferðum þá þarf ekki að búa til hlut til að hægt sé að kalla á klasaaðferð. Ef frátekna orðið `static` er fyrir framan heiti aðferðarinnar þá er hún klasaaðferð. Klasaaðferðir geta verið aðgengilegar öðrum klösum ef þær eru með `public` aðgangsstýringar en eru það ekki ef þær eru með `private` aðgangsstýringu. Tökum sem dæmi gamlan kunningja úr TÖL103. Við getum sagt:

```
double root = Math.Sqrt(453.0);
```

Hérna er `Sqrt` klasafall í `Math` sem kallað er á með því að skrifa heiti klasans sem það tilheyrir fyrir framan og setja punkt á milli.

## Hjúpun

Hægt er að nálgast klasa og hluti á tvennan hátt. Annars vegar þarf að úthugsa þá í smáatriðum á meðan verið er að hanna þá og forrita en hins vegar þarf að vera hægt að hugsa um þá eins og einingar sem okkur koma lítið við að öðru leyti en því að þær geri það sem þær eiga að gera. Við viljum geta notað tilvik/hluti ítrekað án þess að þurfa að skilja til hlítar innviði þeirra. Við viljum geta kallað á aðferðir sem vinna sitt verk án þess að við þurfum að eyða tíma í að brjóta þær til mergjar. Sá eiginleiki að þurfa ekki að huga að neinu öðru en inntaki, úttaki og skilagildi er einn af ávinningum hlutbundinnar forritunar því þar með opnast sá möguleiki að endurnýta klasa. Þessir endurnýttu klasar geta verið klasar sem forritarinn sjálfur eða aðrir hafa gert. Við þurfum t.d. og sem betur fer ekki að skilja til hlítar hvernig gírkassi í bíl virkar til þess að geta ekið bíl. Það myndi auka fyrirhöfn okkar verulega ef við þyrftum að rifja það upp í hvert skipti sem við ætluðum að nota bílinn. Þannig má segja að gírstöng á bíl sé hjúpun á gírkassanum

Því hún tryggir rétta notkun gírkassans. Það gæti skemmt kassann ef gírstöngin væri ekki til staðar og fólk ætti t.d. að róta með skiptilykli í gírkassanum til að skipta um gír. Hliðstæð rök eru færð fyrir því að að tilviksbreytum eða klasabreytum sé ekki breytt nema með sérstökum aðferðum sem skilgreindar eru í klasanum það má segja að þær séu *hjúpaðar* (encapsulated). Með sérstökum *aðgangsstýringum* (visibility modifiers) er síðan hægt að loka algerlega á aðgang frá utanaðkomandi klösum eða takmarka hann verulega. Þeir klasar sem hingað til hafa verið sýndir hafa ekki haft neinar aðgangsstýringar sem hefur gefið öðrum klösum leyfi til að breyta innihaldi þeirra. Aðgangsstýringar skiptast í *private*, *protected* og *public*.

## Hugtök

**Aðferð:**  
Safn skipana sem ætlað er að leysa afmarkað verkefni.

Tafla T203\_1\_1. Aðgangsstýringar

private	Enginn aðgangur leyfður frá öðrum klösum
protected	Aðgangur leyfður frá erfingjum klasans (sjá erfðahugtakið í næsta kafla).
public	Aðgangur heimill frá öllum klösum
internal	Aðgangur leyfður frá klösum í sama assembly kóða (MSIL kóða).

Ef breyta, aðferð eða klasi er **private** þá er enginn aðgangur leyfður frá köllum sem koma frá aðferðum utan klasans. Ef aðgangsstýring er **protected** þá er aðgangur leyfður frá öllum undirklösum allstaðar en ef aðgangsstýring er **public** þá er aðgangur heimill frá öllum klösum hvar sem er. **internal** opnar á klasa í sömu MSIL skrá (assembly). Ef ekkert er tekið fram þá er aðgangur sjálfkrafa *private*. Það er því ljóst að hægt er að laga verulega bílaklasann hérna að framan. Í stað þess að gefa breytu gildi með setningu á borð við:

```
b.Tegund = "Bjalla";
```

eða

```
Bill.fjoldi_bila++;
```

eins og gert er í T203Sd1\_4 væri betra að láta Tegund fá private aðgangsstýringu og láta smiðinn um að gefa henni gildi. Smiðurinn ætti einnig að telja upp gildið á klasabreytunni fjoldi\_bila. Einnig væri rétt að skrifa sérstaka aðferð til að sækja gildið á fjoldi\_bila til að skrifa út svo hægt sé að þrengja aðganginn að henni niður í private aðgang sem kemur í veg fyrir alla mis-

## Hugtök

### Tilviksaðferð:

*Aðferð sem er ekki kyrrleg. Þ.e. hver hlutur fær sitt eigið eintak af aðferðinni. Til að kalla á aðferðina verður að ná í hlutinn sem hún tilheyrir.*

```
/**
*****
Skrá: T203Sd1_5.cs
Höfundur: Ragnar Geir Brynjólfsson.
Dagsetning: 24.05.2005.
Klasinn Bill skilgreindur
*****
*/using System;
class Bill {
private static int fjoldi_bila;
private string Gerd;
string Numer,Stada;
int Argerd,Hradi,Ekinn;
public Bill(string g, string n,int a,int e) {
fjoldi_bila++;
Gerd=g;Numer=n;Argerd=a;Ekinn=e;
Console.WriteLine("Bíll af gerð "
+ Gerd + " búinn til.");
}
public void keyra(int h) {
Hradi = h;
Stada = "Bíll " + Numer + " er núna á keyrslu"
+ " á hraða " + Hradi + ".";
Console.WriteLine(Stada);
}
public void stansa(int km) {
Ekinn+=km;
Stada = "Bíll " + Numer + " stansar.";
Console.WriteLine(Stada);
}
public void skrifa_uppl () {
Console.WriteLine("\nGerð: " + Gerd
+ "\nNúmer: " + Numer + "\nEr ekinn: "
+ Ekinn + " kílómetra.");
}
public void skrifa_fjolda() {
Console.WriteLine("Bílafjöldinn er: "+ fjoldi_bila);
}
}
```

**Klasinn er notaður í öðrum forritum.**