6
Eiffel
Programmeringsomgivelsen.

Programafvikling.

Klasser, klynger og univers.

Systembeskrivelsesfilen.

Bibliotekter.

Oversættelse og udførelse.

Den interaktive omgivelse.

Dokumentationsværktøj.

PS1 -- Eiffel programmeringsomgivelsen

Noter

En programmeringsomgivelse er et sæt af (mere eller mindre) integreret programudviklingsværktøj.

Eiffel omgivelsen er beskrevet og dokumenteret i manualen "Eiffel the Environment". Kapitel 2 i denne manual er en let-at-følge eksempel session, hvor et på forhånd lavet program oversættes og udføres.

Eiffel3 omgivelsen er ændret radikalt i forhold til Eiffel2 omgivelsen. Derfor giver det ikke mening af læse om omgivelsen i OOSC.

I forelæsningen som dækkes af dette kapitel vil vi indskrænke os til at behandle de mere principielle emner i Eiffel omgivelsen, samt at introducere vigtige begreber. Det er således ikke formålet med materialet i dette kapitel at “give et kørekort” til selve omgivelsen.

I Eiffel-kataloget (som er det sted, hvor alle filer i vores Eiffelinstallation befinder sig) findes der et katalog som hedder 'examples'. I dette katalog er der en række eksempelprogrammer, som man kan have nytte af at se på og lære af.

Man kan placere sig selv i Eiffel kataloget ved at skrive:

```
cd $EIFFEL3
```

Programafvikling.

To forskellige programafviklings-former:

- **Total** programafvikling:
  - Programudførelsen starter med udførelse af et hovedprogram.
  - Programudførelsen terminerer, når og hvis hovedprogrammet terminerer.
- **Inkrementel** programafvikling:
  - Programudførelse består i at programmøren skaber objekter og kalder operationer på disse i en interaktiv dialog.

Mulige fortolkninger af “hovedprogram” i et objekt-orienteret sprog:

- Et hovedprogram a la Pascal
- En funktion ekstern til klasserne, der udnævnes til hovedprogram.
- En operation i en på forhånd udpeget klasse, som instantieres af programmeringsomgivelsen.

PS1 -- Eiffel programmeringsomgivelsen

Noter

I de programmeringssprog, vi er optagede af på programmeringssystemer 1, benyttes total programafvikling.

I Eiffel skal man via programmeringsomgivelsen udnævne en klasse, som er rod i systemet. Denne klasse instantieres af Eiffel systemet, og en nærmere angive creation-procedure i denne klasse bliver dernæst kaldt. Med dette udgangspunkt starter programafviklingen i et Eiffel program. Man kan sige, at den angivne creation procedure i rod klassen spiller rollen af et hovedprogram.

Klasser, klynger og univers i Eiffel.

Én klasse én fil.

- Hver klasse skal placeres på en fil, som har samme navn som klassen, med fil-extension '.e' .

Klynger.

- En klynge (cluster) er en samling af klasser, som er lagret på filer i det samme katalog i filsystemet (i operativsystemet).

- Klynger er en program-organisatorisk enhed, der ikke understøttes af sproget.

Univers.

- Et univers er mængden af alle klasser, der kan komme i betragtning i en oversættelse af en given klasse.

- Et univers defineres som en ordnet mængde af klynger.

PS1 -- Eiffel programmeringsomgivelsen

Noter

Filnavnet på en klasse er med små bogstaver.

Rod-klassen er den klasse, der instantieres, når programmet starter. Efter instantiering af denne klasse kaldes en nærmere angiven creation procedure, som typisk hedder make eller create. Denne procedure er ansvarlig for den yderligere fremdrift i programmet.

Systembeskrivelsesfilen: ACE.

Hovedformålet med systembeskrivelsesfilen er at angive

- rod-klassen og creation-procedure
- sekvensen af klynger i universet.
- 'options' til compileren.

Systembeskrivelsesfilen hedder **Ace**

Systembeskrivelsesfilen lagres i samme katalog som rod-klassen.

Simpel systembeskrivelse:

```
system name root
   root-class (creation-procedure)
default
   compiler-options
    precompiled("file")
cluster
   cluster-listning
end
```

PS1 -- Eiffel programmeringsomgivelsen

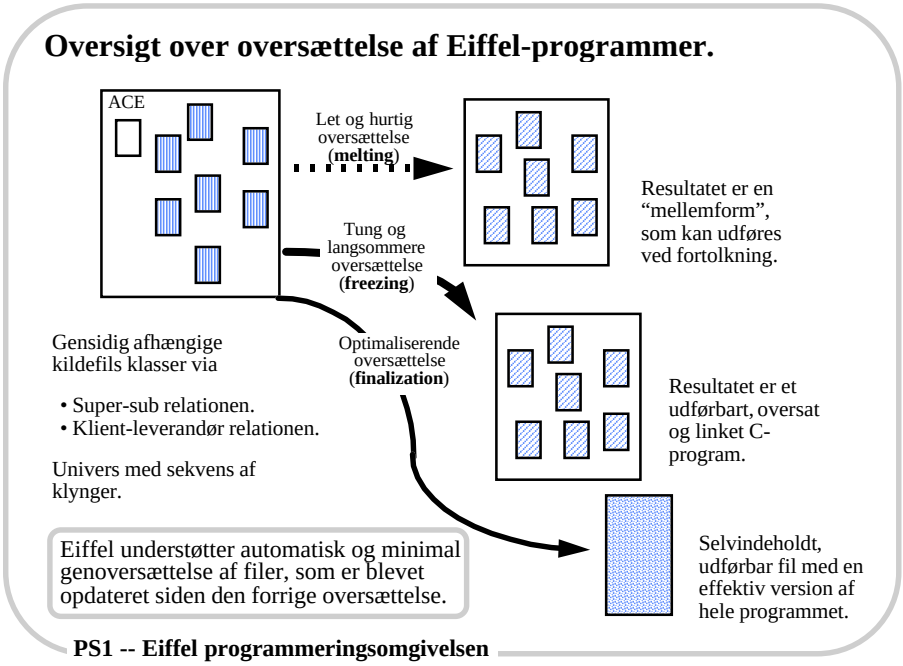
Noter

Der findes en skabelon for Ace filer, som det anbefales at man benytter på dette kursus. Skabelonen finder man på filen:

$EIFFEL3/library/local/Ace.tpl

Appendix D i "Eiffel the Language" beskriver i stor detalje hvordan ACE filer skal skrues sammen. Hvis man har et specielt behov i forbindelse med sammensætning af klasserne til et udførbart Eiffel program, bør man konsultere dette appendix.

En bemærkning om prækompilering, og **precompiled** under **default** ovenfor: Prekompilering handler om forud-oversættelse af dele af Eiffelbiblioteket, én gang for alle. Dermed skal der ikke bruges tid på at oversætte klasser fra Eiffelbibliotekter, når disse bruges. Der kan kun angives én fil, som refererer til ét sæt af prækompilerede klasser. Undervejs i kurset vil det blive oplyst, hvilke muligheder til tilbydes. Se ovenstående Ace skabelon fil for et godt bud på et sæt af prækompilerede klasser. Det er ikke nødvendigt at angiver clusters for klasser, der er omfattet af prækompilering. Men det skader ikke at gøre det. Hvis man senere skal "finalize" er det nødvendigt at gøre det, idet finalization ikke kan være afhængig af en prækompilering.Der henvises til afsnit 6.5.2 i "Eiffel the Environment" angående anvendelse af prækompilerede biblioteker.



Noter

Denne slide indeholder en oversigt over compilering i Eiffel omgivelsen.

Vores Eiffel oversætter benytter C som et højniveau assembler sprog, i den forstand at et Eiffel program først oversættes til C, dernæst oversættes C programmet til maskinkode, og dernæst linkes de forskellige dele til et udførbart program. Det siger sig selv at en sådan oversættelsesproces er væsentlig langsommere end de fleste andre teknikker (direkte oversættelse til maskinkode). Fordelenen ved teknikken er naturligvis, at C kan anvendes som en maskin-uafhængig mellemforms sprog; det er således ikke nødvendigt at lave forskellige Eiffel (bagende-)oversættere til forskellige maskintyper mv.

Eiffel-folkene benytter et billedsprog for at forklare de forskellige former for oversættelse: smeltning og frysning.

Smeltning er er en let oversættelse, som ikke involverer C-oversættelse og linkning. Dette der medicinen mod den langsommelige oversættelse af Eiffel til C, og videre mod makskinkode og linkning, som har plaget Eiffel2 programmører. Et Eiffel program, der én gang er oversæt til C opfattes som frosset. Et sådant program kan tøes op, i takt med at man ændrer i dele af programmet. De optøede dele genoversættes ikke gennem C til maskinkode, men fortolkes derimod på et langt højere niveau. *Det er altså muligt i Eiffel3 at opnå programudførelsen ved en kombination af fortolkning af maskinkode, og fortolkning af en mellemform, der er langt billigere at producere.*

Frysning består i oversættelse af Eiffelprogrammet til C, C kompilering, og linkning. Dette tager typisk fra 1/2 minut og opefter, afhængig af programstørrelse, maskintype og belastning.

Den automatiske genoversættelse af Eiffel klasser er af stor praktisk værdi. I andre sprog, f.eks. C++, er det programmørens ansvar at oversætte de enkelte klasser. Hvis der er mange klasser (på mange filer) er dette en meget krævende opgave. Derfor findes der værktøj i omgivelsen, som understøtter denne opgave. I Unix hedder dette værktøj **make**.

Options til Eiffel oversætteren.

Options til en oversætter beskriver forskellige variationer i oversættelsen, som vil afspejle sig i forskellige resultater af transformationen fra kildeprogram til udførbart program.

default
assertion(*assertion-niveau*); **trace**(*trace-niveau*)
debug(*debug- niveau*)

uddrag af ACE fil

Assertion niveau	*Betydning*
no	ingen assertions check
require	kun prebetingelser checkes (default)
ensure	kun postbetingelser og prebetingelser checkes
invariant	klasse invarianter samt pre- og postbetingelser checkes
loop	løkkeinvarianer, -varianter, klasseinv. samt pre- og postbet. checkes
check	check instruktioner samt alt under "loop" checkes
all	alle assertions checkes (identisk med check)

Noter

Som det ses, er assertion niveauerne struktureret således at angivelse af ét niveau inkluderer alle de tidligere niveauer. Under programudvikling anbefales det kraftigt at benytte **all**.

Debug niveauerne er **yes**, **no** eller en særlige nøgle (en tekststreng, se sliden "Specielle kommandoer" i kapitlet om kommandoer og udtryk i Eiffel).

Trace niveauerne er ligeledes **yes** eller **no**. Tracing gør det muligt at følge med i hvilke operationer der bliver kaldt i en programudførelse.

Det er også muligt at styre hvorvide 'garbage collectore' er slået til eller ej. Vi vil imidlertid altid anbefale, at der benyttes garbage collection.

Biblioteker.

Bibliotekerne er grundlaget for genbrug.

- Kerne biblioteket: ($EIFFEL3/library/base/kernel)
  Arrays, strings, files, (aritmetiske) basisklasser, mv.
- Datastrukturbiblioteker: ($EIFFEL3/library/base/structures)
  Lister, mængder, træer, hashtabeller, stakke, køer.
- Iterations bibliotek: ($EIFFEL3/library/base/structures/iteration)
  Klasser med faciliteter til iteration over forskellige datastrukturer.
- Biblioteker til leksikalsk- og syntaks-analyse.
  ($EIFFEL3/library/lex og $EIFFEL3/library/parse).
- Grafik bibliotek. ($EIFFEL3/library/vision).
  X-baserede vinduer, menuer og andre bruger-grænseflade elementer.

Noter

Eiffel systemet indeholder et forholdsvis veludbygget bibliotek, som er under stadig udvikling.

Eiffel bibliotekerne er beskrevet i manualen "Eiffel the Libraries". Dette er en næsten telefonbog-tyk rapport med dokumentation af klasse interfaces. Det er muligt at få dokumentationen af klasse-interfaces ud på en skærm, f.eks. ved brug af den interaktive Eiffel omgivelse (læs senere i dette kapitel om dokumentationsværktøj).

Tag et kig på struktur og indhold i bibliotekerne! Start med

```
cd $EIFFEL3/library
```

og se dig lidt omkring. I del-katalogerne af dette katalog findes selve kilde-filerne til klasserne i Eiffel bibliotekerne.

Den interaktive Eiffel omgivelse.

En vinduesbaseret programmeringomgivelse (med en speciel interaktionsteknik) der tillader editering, oversættelse samt statisk- og dynamisk undersøgelse af et Eiffel program.

- **Projektværktøj**: Hovedværktøj der tillader programoversættelse og - udførelse samt initiering af andet værktøj.
- **Systemværktøj**: Editering af systembeskrivelser (Ace's).
- **Klasseværktøj**: Editering og browsing af klasser:
  - Generering af alternative syn på klasser og deres kontekst.
- **Featureværktøj**: Editering og browsing af features (attributter og routiner).
  - Krydsreferering: Finde alle steder en feature anvendes (kaldes).
  - Mulighed for at angive stop-punkter.
- **Objektværktøj**: Præsentation af et objekt, og rekursivt de objekter det refererer.

PS1 -- Eiffel programmeringsomgivelsen

Noter

Store dele af "Eiffel the Environment" handler om den interaktive programmeringsomgivelse for Eiffel. Det er ikke formålet med dette kapitel at gå i nogen detalje om dette.

Eiffel omgivelsen´s værktøj kan tilgås på andre måder end gennem den interaktive omgivelse. Det er muligt at oversætte, og generene dokumentation, via en kommando der hedder es3. Dennekommando gives direkte til en UNIX shell, og den tager forskellige parametre alt efter hvad man ønsker. Detaljer om denne kommando findes beskrevet i kapitel 10 i "Eiffel the Environment".

Endvidere forberedes til tilgang til Eiffel3 værktøjet (primært compileren) fra Emacs, som er den foretrukne teksteditor for mange brugere af et Unix-baseret system. I skrivende stund er det for tidligt at sige noget konkret om dette.

Dokumentationsværktøj.

En *short* præsentation af en klasse:

Giver en abstrakt præsentation af klassens eksternt interessante dele.

- features, uden kroppe af routiner.
- kommentar
- Specifikation (logiske udtryk).

En *flat* præsentation af en klasse.

Sammentrækker direkte og indirekte superklasser af en klassse til én klasse (eliminerer nedarvning).

En *flat-short* præsentation af en klasse.

Sammentrukket abstrakt præsentation.

Noter

Mange sprog tillader, at man udtrykker korte og klare interface beskrivelser af datatyper. Det er tilfældet i C++ (traditionelt i .h filer), i Modula definitionsmoduler, og ligeledes i ADA.

I Eiffel er dette ikke gjort let. Man kan skrive abstrakte klasser, som tjener udelukkende som interfaces, men det grænser til misbrug af abstrakte klasser, hvis dette gøres konsekvent for alle klasser.

I Eiffel er filosofien derimod, at klasseinterfacet kan trækkes ud af den totalt beskrevne klasse ved anvendelse af et stykke værktøj: **short**.

Short og flat-short præsentationer kan genereres via klasse værktøjet i den interaktive omgivelser. Flat og flat-short præsentationer kan genereres via es3 kommandoen, som blevt nævnt i noterne til forrige slides.