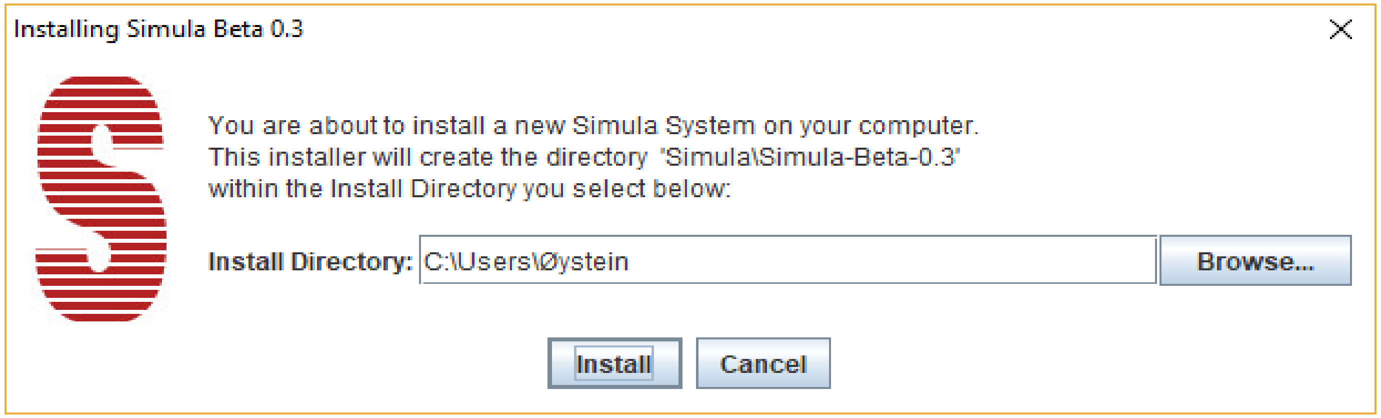
Agenda

- Installing Simula Systems
- Demo: Simula Editor App
- Compiler Overview
- Switch Statement
- Signering av 'setup.jar' ?
- UTF-8 ?!
- Java 11 ?
- Status Project Loom

Installing Simula



Installing Simula Beta 0.3

You are about to install a new Simula System on your computer.
This installer will create the directory 'Simula\Simula-Beta-0.3'
within the Install Directory you select below:

Install Directory: C:\Users\Øystein

I dette tilfelle får vi følgende directory struktur:

C:\Users\Øystein

\Simula (This Directory is called 'Simula Home')

\Simula-Beta-03

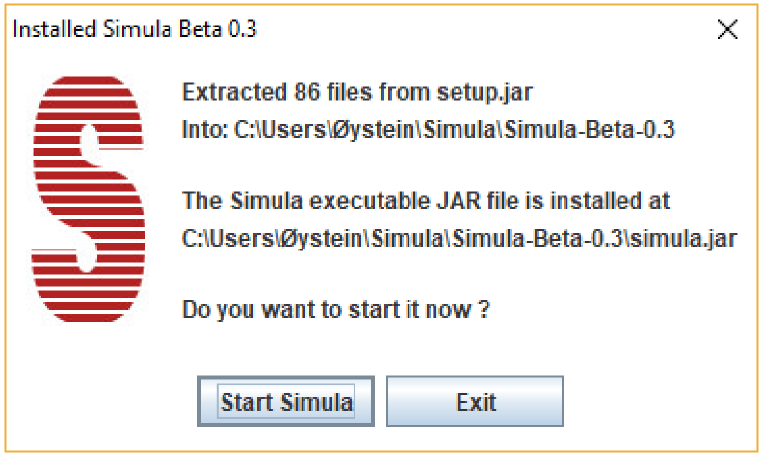
\rts ... (Simula Runtime System)

\tst ... (Simula Sample Programs)

simula.jar (Executable .jar)

ReleaseNotes.txt

Installation Completed


Installed Simula Beta 0.3

Extracted 86 files from setup.jar
Into: C:\Users\Øystein\Simula\Simula-Beta-0.3

The Simula executable JAR file is installed at
C:\Users\Øystein\Simula\Simula-Beta-0.3\simula.jar

Do you want to start it now ?

The actual installation directory and executable simula .jar is reported.

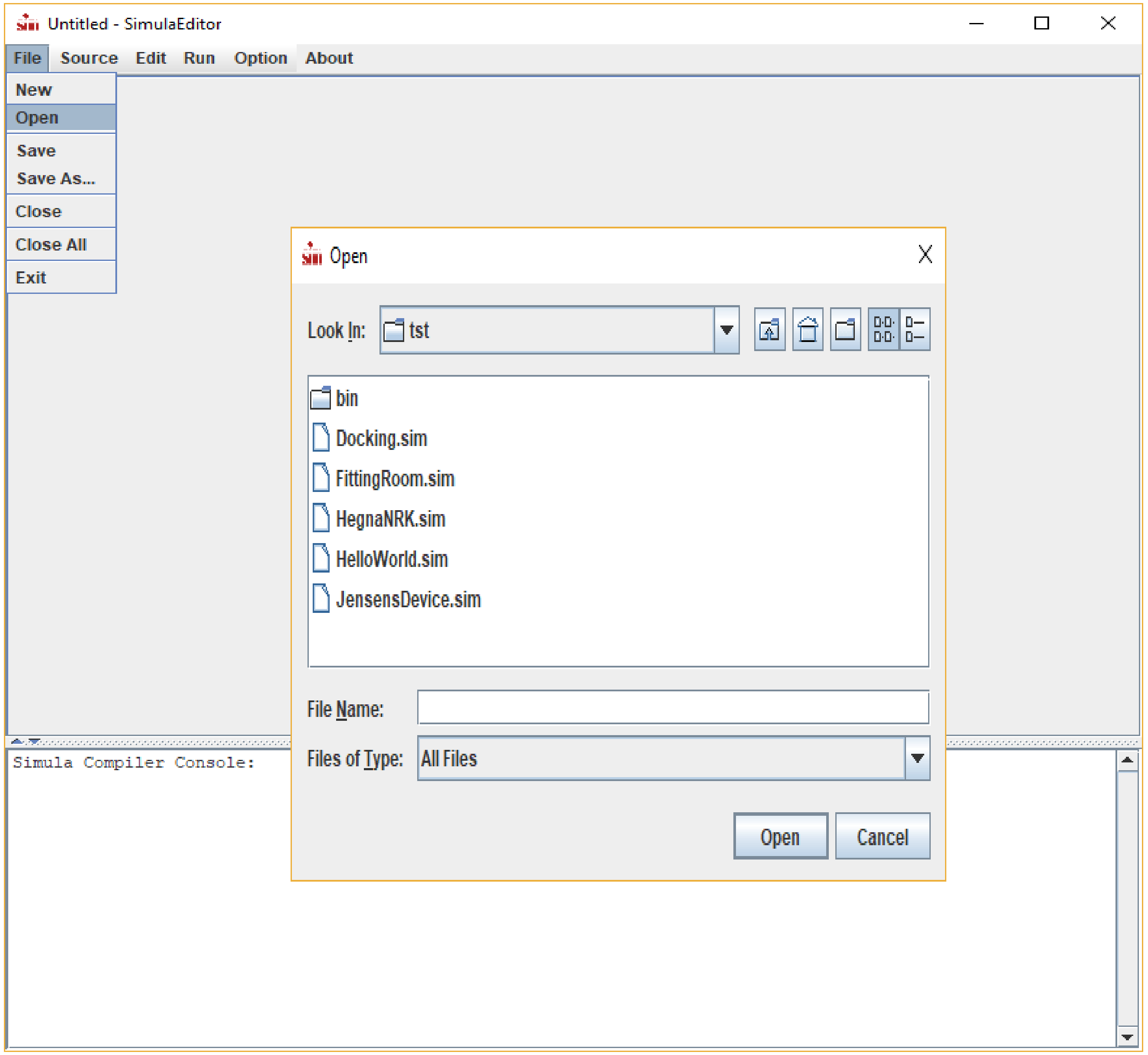
Press [Exit] and the installation is finished

Press [Start Simula] and the executable simula.jar is started.

In this case without any arguments which will result in the Simula Editor being started.

See next page for details ....

Simula Editor

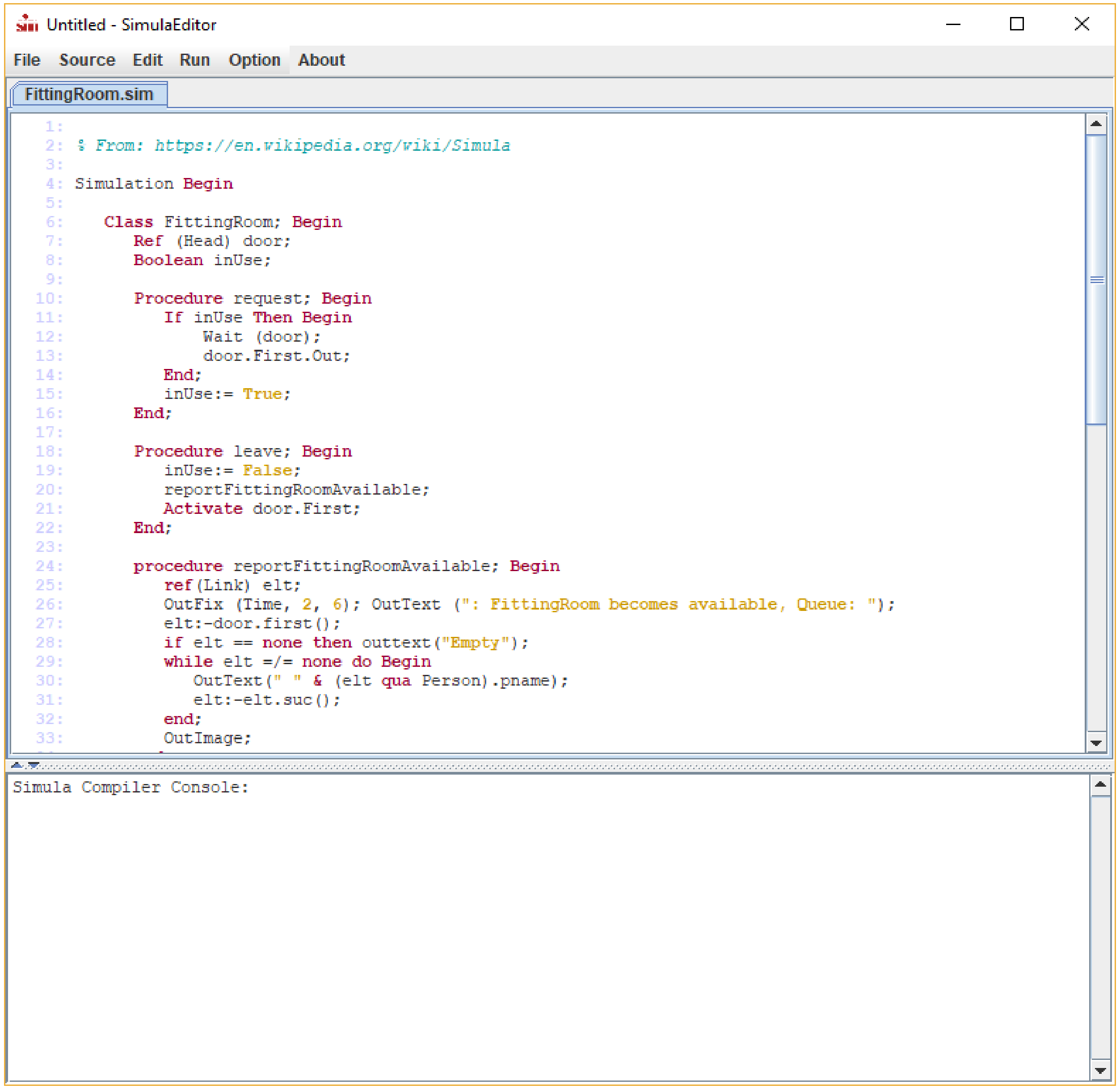


```
Simula Compiler Console:
```

Select [File] [Open] to open

a sample Simula Source File

Context Sensitive Editing



```

$ From: https://en.wikipedia.org/wiki/Simula

Simulation Begin

   Class FittingRoom; Begin
      Ref (Head) door;
      Boolean inUse;

      Procedure request; Begin
         If inUse Then Begin
             Wait (door);
             door.First.Out;
         End;
         inUse:= True;
      End;

      Procedure leave; Begin
         inUse:= False;
         reportFittingRoomAvailable;
         Activate door.First;
      End;

      procedure reportFittingRoomAvailable; Begin
         ref(Link) elt;
         OutFix (Time, 2, 6); OutText (": FittingRoom becomes available, Queue: ");
         elt:-door.first();
         if elt == none then outtext("Empty");
         while elt =/= none do Begin
            OutText(" " & (elt qua Person).pname);
            elt:-elt.suc();
         end;
         OutImage;
```

```
Simula Compiler Console:
```

This is a preliminary Editor you should use with care.

To Compile and Run select

[Run] [Run]

... enjoy

Directories during Compilation

Anta at Simula Systemet er installert i direktoriet <release-dir>
som typisk vil være   C:\Users\Øystein\Simula\Simula-<version>\   på en Windows Maskin

Da vil <simula.home> være C:\Users\Øystein\Simula og dette er lagret i filen
    C:\Users\Øystein\.simula/simulaProperties.prop

Anta videre at vi kompilerer en simula-source <source-dir>/simtest.sim

Og anta til slutt at <temp> er resultatet av kall på `System.getProperty("java.io.tmpdir")`

Følgende directories er da i bruk under kompilering:

• SourceFile Dir:	<source-dir>
• TempDir for .Java:	<temp>\simula\src\simprog/
• TempDir for .Class:	<temp>\simula\bin\simprog/
• SimulaRtsLib:	<release-dir>/rts/
• OutputDir:	<source-dir>/bin/

Hvis compiler option 'verbose' er satt vil disse verdiene bli skrevet ut ved oppstart av kompileren.

Compiler Steps
(see SimulaCompiler.java)

- Set up directory files

- SourceFile Dir:	<source-dir>
- TempDir for .Java:	<temp>\simula\src\simprog/
- TempDir for .Class:	<temp>\simula\bin\simprog/
- SimulaRtsLib:	<release-dir>/rts/
- OutputDir:	<source-dir>/bin/

- Do Parsing

  Read source file through the scanner building program syntax tree

- Do Checking

  Traverse the syntax tree performing semantic checking

- Do JavaCoding

  Traverse the syntax tree generating .java code

- Call Java Compiler to generate .class files
- Do `ByteCodeEngineering updating .class files`
- `Create executable .jar of program`
- `Execute .jar file`

S-Port Switch Statement

Utvidelse arvet fra S-Port Simula.

```
switch-statement = SWITCH ( lowKey : hiKey ) switchKey BEGIN { switch-case } [ none-case ] END

     switch-case = WHEN <caseKey-list> do <statement> ;

     none-case   = WHEN NONE do <statement> ;

     <caseKey-list> = caseKey { , caseKey }

     caseKey = caseConstant  | caseConstant : caseConstant


     lowKey = integer-or-character-expression

     hiKey = integer-or-character-expression

     switchKey = integer-or-character-expression

     caseConstant = integer-or-character-constant
```

Oversettes til et Java Switch Statement med **break** etter hvert enkelt <statement>

Forekommer i S-Port kompilatoren til Peter – men jeg har ikke funnet beskrivelse av semantikken.

Nytt Switch Statement i Java 12

In the following code, the many break statements make it unnecessarily verbose, and this visual noise often masks hard to debug errors, where missing break statements mean that accidental fall-through occurs.

```
switch (day) {

    case MONDAY: case FRIDAY: case SUNDAY:

        System.out.println(6);  break;

    case TUESDAY:

       System.out.println(7);  break;

    case THURSDAY: case SATURDAY:

        System.out.println(8);  break;

    case WEDNESDAY:

        System.out.println(9);  break;

}
```

We propose to introduce a new form of switch label, written "case L ->" to signify that only the code to the right of the label is to be executed if the label is matched. For example, the previous code can now be written:

```
switch (day) {

    case MONDAY, FRIDAY, SUNDAY -> System.out.println(6);

    case TUESDAY                -> System.out.println(7);

    case THURSDAY, SATURDAY     -> System.out.println(8);

    case WEDNESDAY              -> System.out.println(9);

}
```

Se: https://openjdk.java.net/jeps/325

Morsomt å se at de nå foreslår nesten det samme Switch Statement som Peter hadde i S-Port. De mangler bare intervaller:

F.eks: `case MONDAY : FRIDAY  ->  System.out.println(" ukedag ");`

Signering av Setup.jar

- MAC krever signering
- Windows advarer pga. Manglende signering
- Har vi tilgang til serfikat ?
- Kan Eyvind gjøre dette ?
- ... men først når vi er på nivå – beta 1.0
- Må også få på plass:

A newer version og Simula is available

do you want to download now ?

UTF-8 ?!

- Blir standard i Java 13 planlagt ferdig ???
- Målsetning: Use UTF-8 as the Java virtual machine's default charset so that APIs that depend on the default charset behave consistently across all platforms
- Se: Use UTF-8 as default Charset
- Dagens situasjon er rimelig rotete ihvertfall på Windows maskiner.
- Jeg har tatt vekk en 'reflection' retting av 'DefaultCharset' da Java 10+ ikke synes noe særlig om den måten å patche !
- Problematikken forklart her: Java May Use UTF-8 as its Default Charset

Utsetter setting av UTF-8

Java 11 ?

- Naturlig å gå over til Java 11 nå ?
- Isåfall – Java 11 under GNU GPL
- Hva bruker dere nå ?

Mer info: https://jdk.java.net/11/

Project LOOM

Project Loom is to intended to explore, incubate and deliver Java VM features and APIs built on top of them for the purpose of supporting easy-to-use, high-throughput lightweight concurrency and new programming models on the Java platform. This is accomplished by the addition of the following constructs:

- Fibers (lightweight user-mode threads)
- Delimited continuations
- Tail-call elimination

Ser ut til at ' Delimited continuations ' passer til Simula's Detach / Call / Resume

Det foreligger en implementasjon og vi kan prøve det.

Ingen tidsplan for inkludering i Java ennå ? Jeg har prøvd å spørre.

Mer info her: https://wiki.openjdk.java.net/display/loom

- Source her: Continuation.java

LOOM: Delimited continuations

```
public class Continuation implements Runnable {
    public Continuation(ContinuationScope scope, Runnable target)
    public final void run()
    public static void yield(ContinuationScope scope)
    public boolean isDone()
       ....
}
```

A continuation object is constructed by passing two arguments to the constructor: a Runnable target that serves as the body of the continuation, and a java.lang.ContinuationScope. The scope is the delimited continuation's prompt, that allows continuations to be nested. One could think of such "scoped continuations" as nested try/catch blocks, where the scope is the type of the exception thrown, which determines the handler called.

A continuation is started by calling Continuation.run, which would start executing the body in the continuation's target, and returns either when the continuation terminates (the body runs to completion, and terminates either normally or abnormally), or when it yields on the continuation's scope. To query the reason for run returning, use Continuation.isDone, which returns true if the body has terminated, or false if it has yielded.

A call to the static Continuation.yield suspends the current continuation and all enclosing continuations up until the innermost one with the scope passd to yield, causing the run method of that continuation to return.

The Continuation class is implemented natively in Hotspot (except for scoping; that is implemented in Java). Every continuation has its own stack. From the perspective of the implementation, starting or continuing a continuation mounts it and its stack on the current thread – conceptually concatenating the continuation's stack to the thread's – while yielding a continuation unmounts or dismounts it.

Jfokus Developers Conference
4-6 February 2019
Stockholm, Sweden

Link: https://www.jfokus.se/jfokus19/

Min mann, Rafael Winterhalter, i Java-miljøet sier:

- Jeg skal på java language summit i stockholm starten av februar.
- De fra LOOM teamet er alle der.
- Skal sikkert finne ut noe om det. ellers kom deg en tur, da introduserer jeg deg

Er det noen som har lyst/kan dra dit ?