Building NetRexx Systems

21st International Rexx Language Symposium

René Vincent Jansen 2010-12-13

Agenda

- Commandline builds
  - nrc
  - ant
  - make

Agenda

- IDE builds
  - Emacs
  - JEdit
  - Eclipse

NetRexx

THE
NETREXX
LANGUAGE
NetRexx for Java
Facilitates internet programming
The definitive source for the NetRexx Programming Language
Contains complete language specification
Easy to learn
M. F. COWLISHAW

- The other Object Oriented successor to Classic Rexx
- 1995, Mike Cowlishaw
- Runs on the Java VM
- Compiles NetRexx to Java classes
- Added an interpreter in 2000
- Will be open sourced, probably ~~this~~ next year

Command line

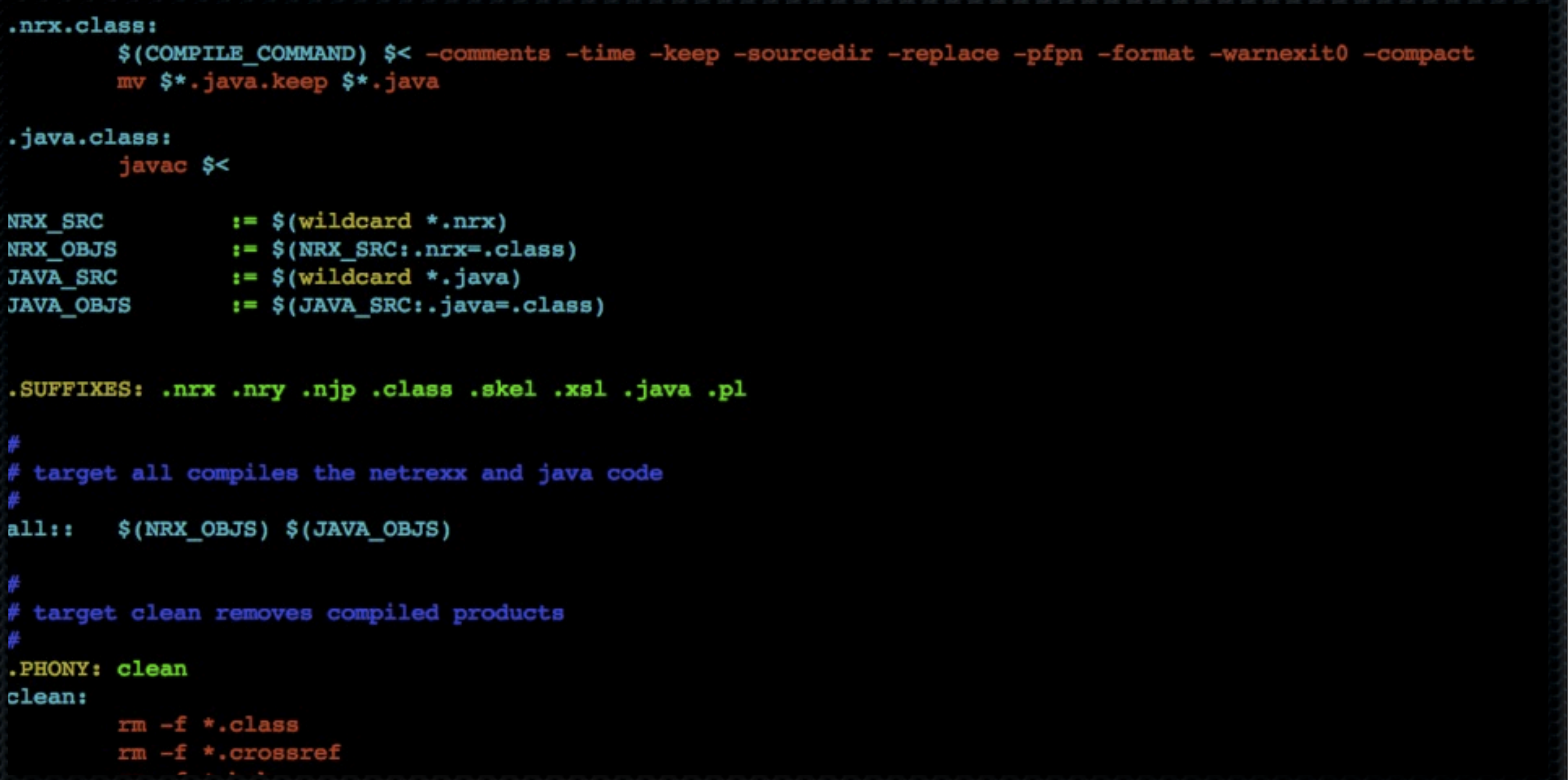
- Aka Shell aka Prompt
- Unix vs Windows differences
- java Com.ibm.netrexx.process.NetRexxC
- resolves dependencies when classes are compiled together

Make

- 1977, Stuart Feldman, Bell's Labs. ACM Software award 2003
- **make** is a utility that automatically builds executable programs and libraries from source code by reading files called makefiles which specify how to derive the target program
- The standard version nowadays is GNU Make

Make

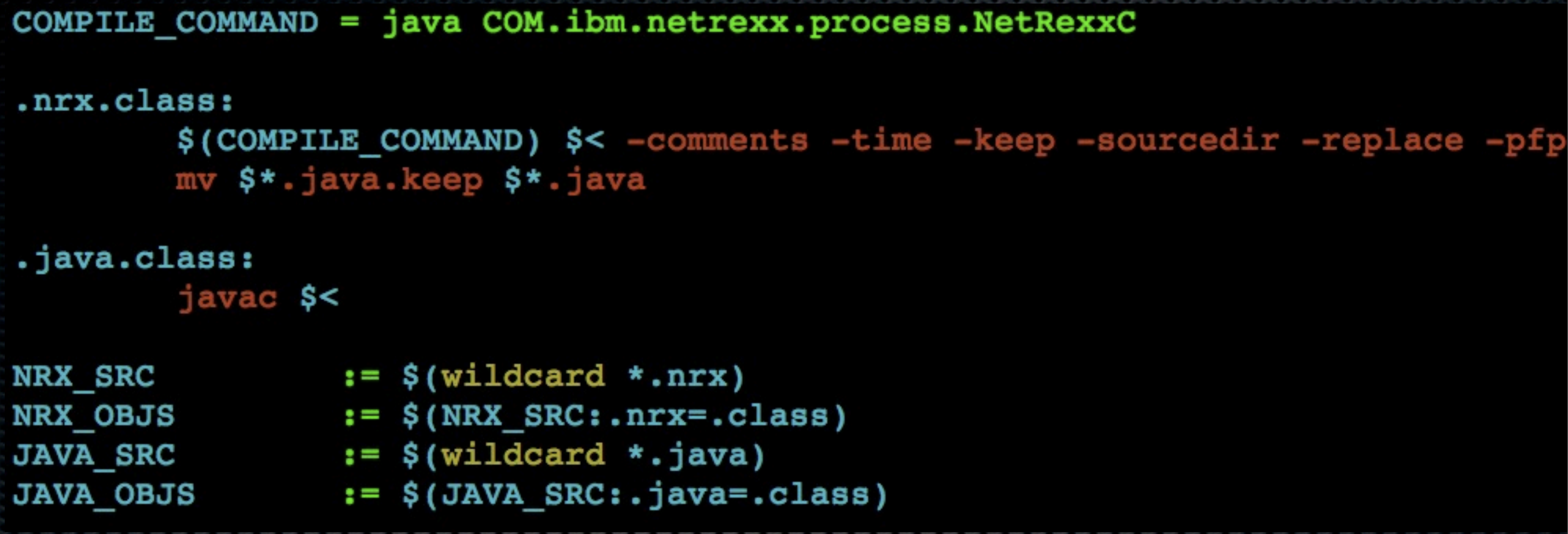
- 1977, Stuart Feldman, Bell’s Labs. ACM Software award 2003
- **make** is a utility that automatically builds executable programs and libraries from source code by reading files called makefiles which specify how to derive the target program
- The standard version nowadays is GNU Make



```
.nrx.class:
        $(COMPILE_COMMAND) $< -comments -time -keep -sourcedir -replace -pfpn -format -warnexit0 -compact
        mv $*.java.keep $*.java

.java.class:
        javac $<

NRX_SRC         := $(wildcard *.nrx)
NRX_OBJS        := $(NRX_SRC:.nrx=.class)
JAVA_SRC        := $(wildcard *.java)
JAVA_OBJS       := $(JAVA_SRC:.java=.class)


.SUFFIXES: .nrx .nry .njp .class .skel .xsl .java .pl

#
# target all compiles the netrexx and java code
#
all::   $(NRX_OBJS) $(JAVA_OBJS)

#
# target clean removes compiled products
#
.PHONY: clean
clean:
        rm -f *.class
        rm -f *.crossref
```

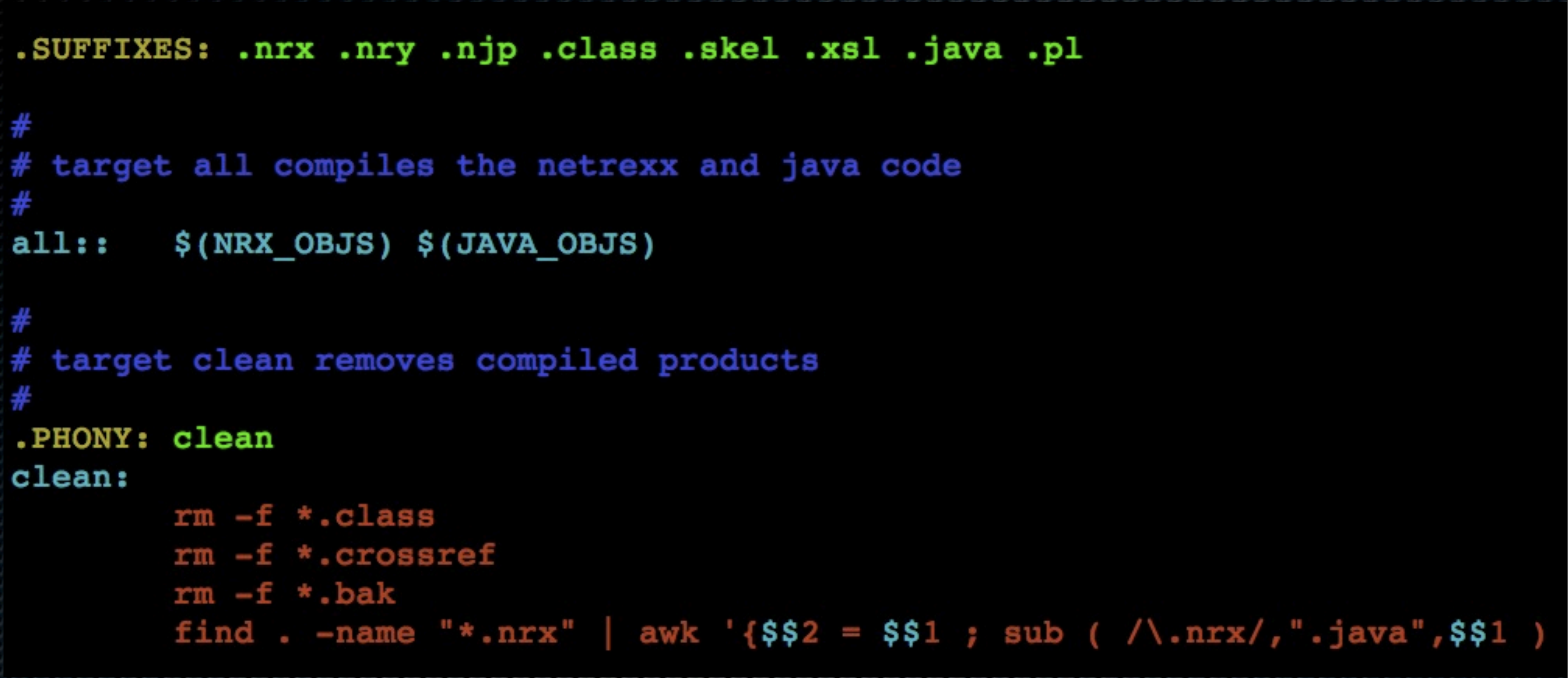
A makefile (called makefile)

will be explained in the next few slides



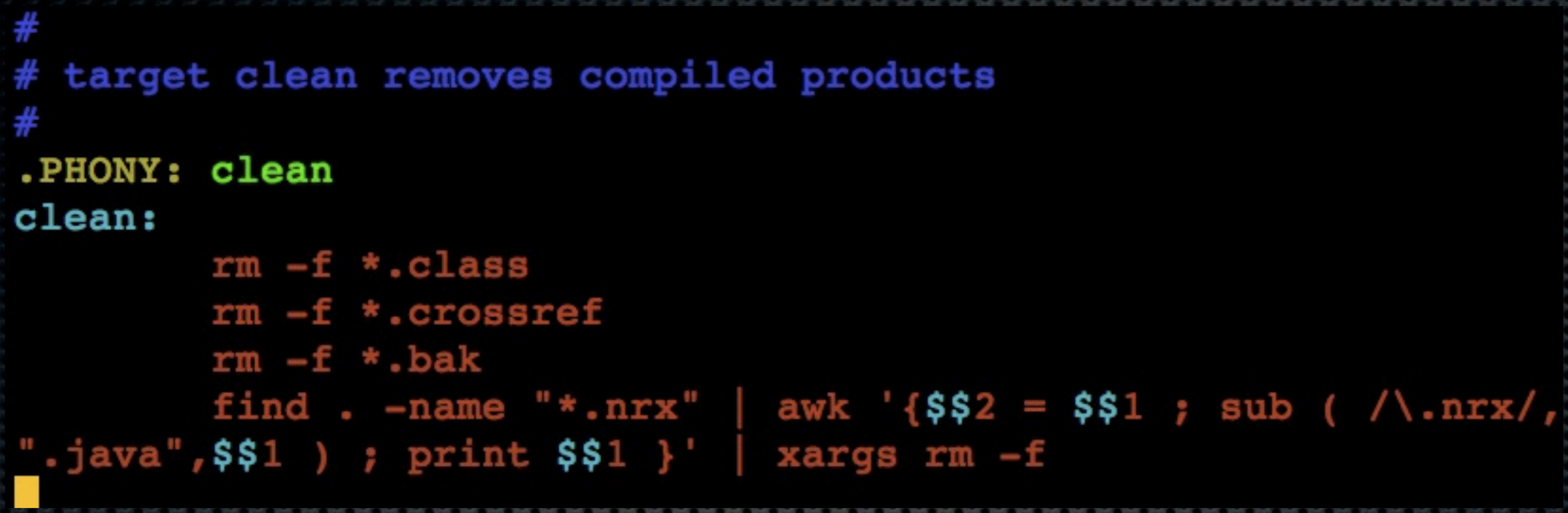
```
COMPILE_COMMAND = java COM.ibm.netrexx.process.NetRexxC

.nrx.class:
        $(COMPILE_COMMAND) $< -comments -time -keep -sourcedir -replace -pfp
        mv $*.java.keep $*.java

.java.class:
        javac $<

NRX_SRC         := $(wildcard *.nrx)
NRX_OBJS        := $(NRX_SRC:.nrx=.class)
JAVA_SRC        := $(wildcard *.java)
JAVA_OBJS       := $(JAVA_SRC:.java=.class)
```

Variables and Rules



```
.SUFFIXES: .nrx .nry .njp .class .skel .xsl .java .pl

#
# target all compiles the netrexx and java code
#
all::    $(NRX_OBJS) $(JAVA_OBJS)

#
# target clean removes compiled products
#
.PHONY: clean
clean:
        rm -f *.class
        rm -f *.crossref
        rm -f *.bak
        find . -name "*.nrx" | awk '{$$2 = $$1 ; sub ( /\.nrx/,".java",$$1 )
```

Suffixes and Targets



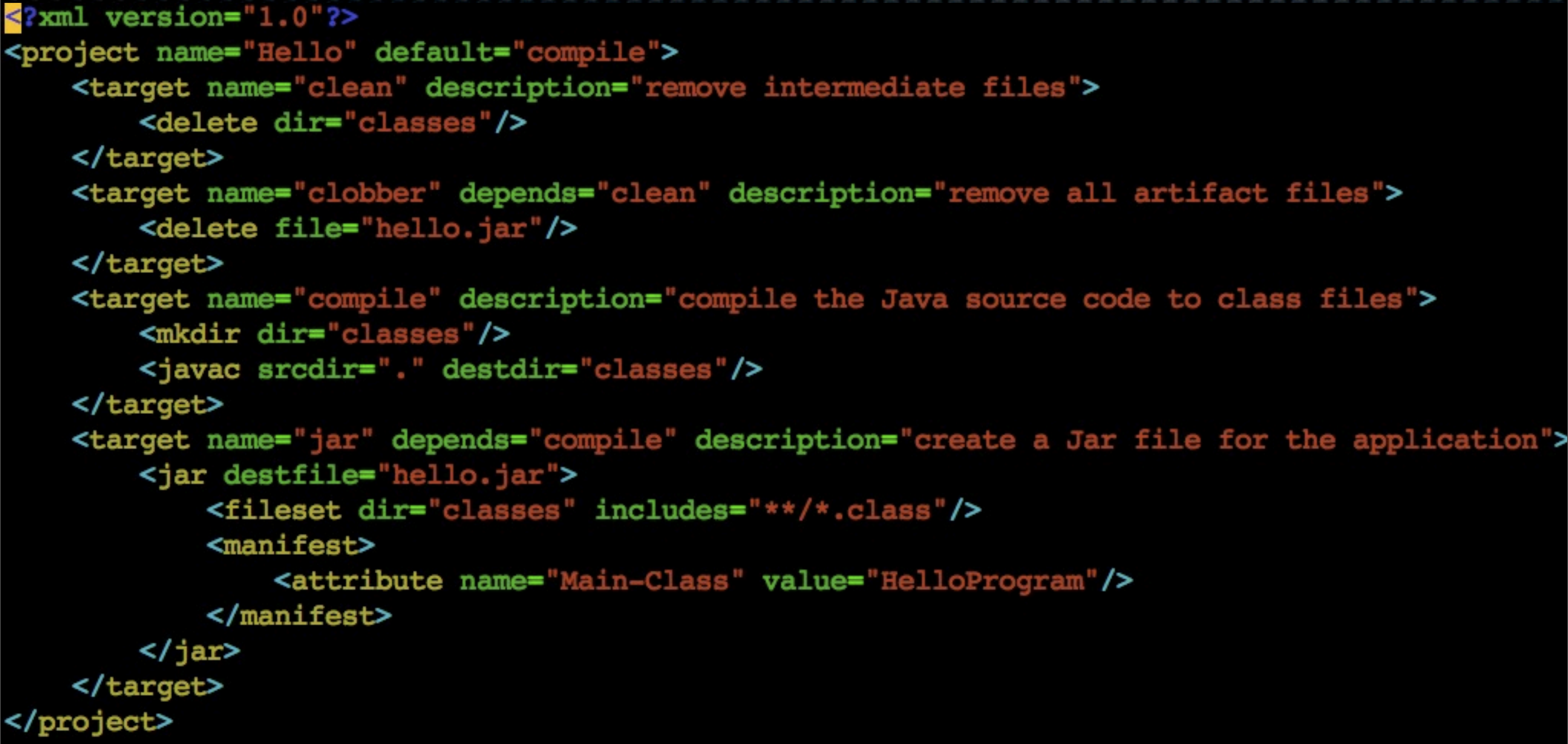
```
#
# target clean removes compiled products
#
.PHONY: clean
clean:
        rm -f *.class
        rm -f *.crossref
        rm -f *.bak
        find . -name "*.nrx" | awk '{$$2 = $$1 ; sub ( /\.nrx/,
".java",$$1 ) ; print $$1 }' | xargs rm -f
```

Take care not to remove sources that only have a ***.java** version

Enables the class-in-source model

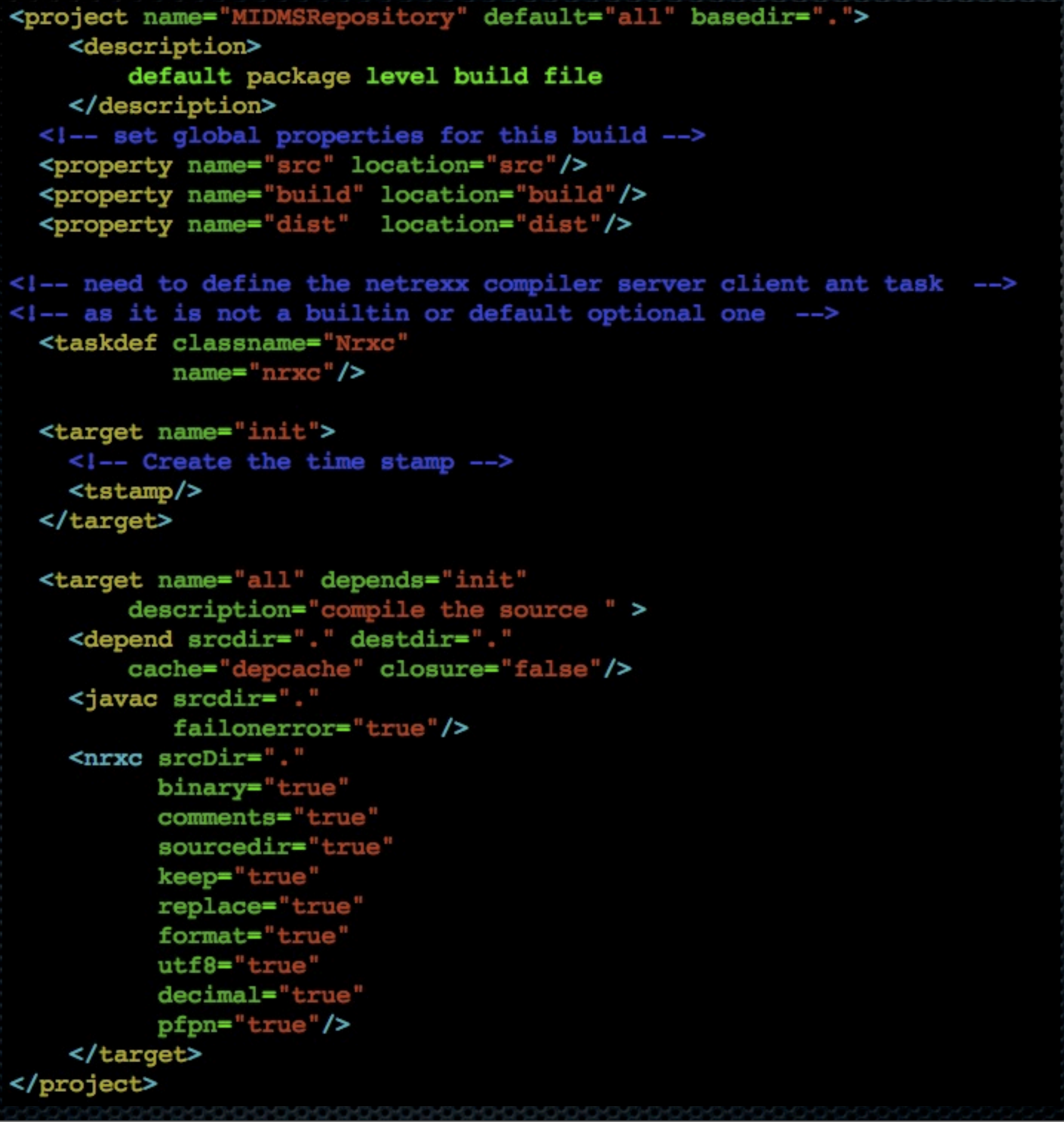
Ant - Another Neat Tool

- Java based - Cross Platform - Building utility
- James Duncan Davidson, July 19, 2000
- The most immediately noticeable difference between Ant and Make is that Ant uses XML to describe the build process and its dependencies, whereas Make uses Makefile format. By default the XML file is named build.xml



```
<?xml version="1.0"?>
<project name="Hello" default="compile">
    <target name="clean" description="remove intermediate files">
        <delete dir="classes"/>
    </target>
    <target name="clobber" depends="clean" description="remove all artifact files">
        <delete file="hello.jar"/>
    </target>
    <target name="compile" description="compile the Java source code to class files">
        <mkdir dir="classes"/>
        <javac srcdir="." destdir="classes"/>
    </target>
    <target name="jar" depends="compile" description="create a Jar file for the application">
        <jar destfile="hello.jar">
            <fileset dir="classes" includes="**/*.class"/>
            <manifest>
                <attribute name="Main-Class" value="HelloProgram"/>
            </manifest>
        </jar>
    </target>
</project>
```

Ant file (“build.xml”) for plain Java class



```
<project name="MIDMSRepository" default="all" basedir=".">
    <description>
        default package level build file
    </description>
  <!-- set global properties for this build -->
  <property name="src" location="src"/>
  <property name="build" location="build"/>
  <property name="dist"  location="dist"/>

<!-- need to define the netrexx compiler server client ant task  -->
<!-- as it is not a builtin or default optional one  -->
  <taskdef classname="Nrxc"
           name="nrxc"/>

  <target name="init">
    <!-- Create the time stamp -->
    <tstamp/>
  </target>

  <target name="all" depends="init"
        description="compile the source " >
    <depend srcdir="." destdir="."
        cache="depcache" closure="false"/>
    <javac srcdir="."
           failonerror="true"/>
    <nrxc srcDir="."
          binary="true"
          comments="true"
          sourcedir="true"
          keep="true"
          replace="true"
          format="true"
          utf8="true"
          decimal="true"
          pfpn="true"/>
    </target>
</project>
```

Sample **build.xml** with NetRexx support

NetRexx Ant Task

- Ant requires a Task for exceptional processing
- The optional tasks contains a NetRexx ant task
  - By Dion Gillard, who has left us too soon
- Its source available and can be adapted to project needs
- Arjan Bos has written a NetRexx-version of this task

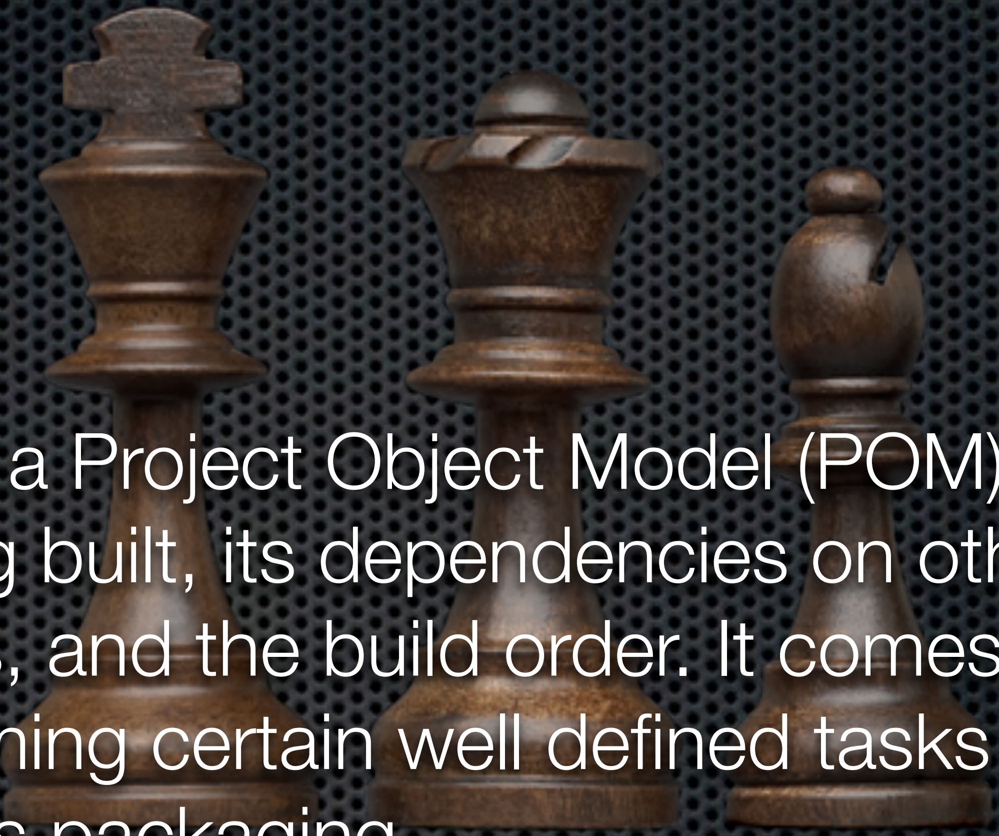
Ant remarks

- Ant will build whole directories
- Class-in-source is feasible with the Arjan Bos version of the NetRexx Ant Task
- The best small footprint alternative for cross-platform builds

Maven

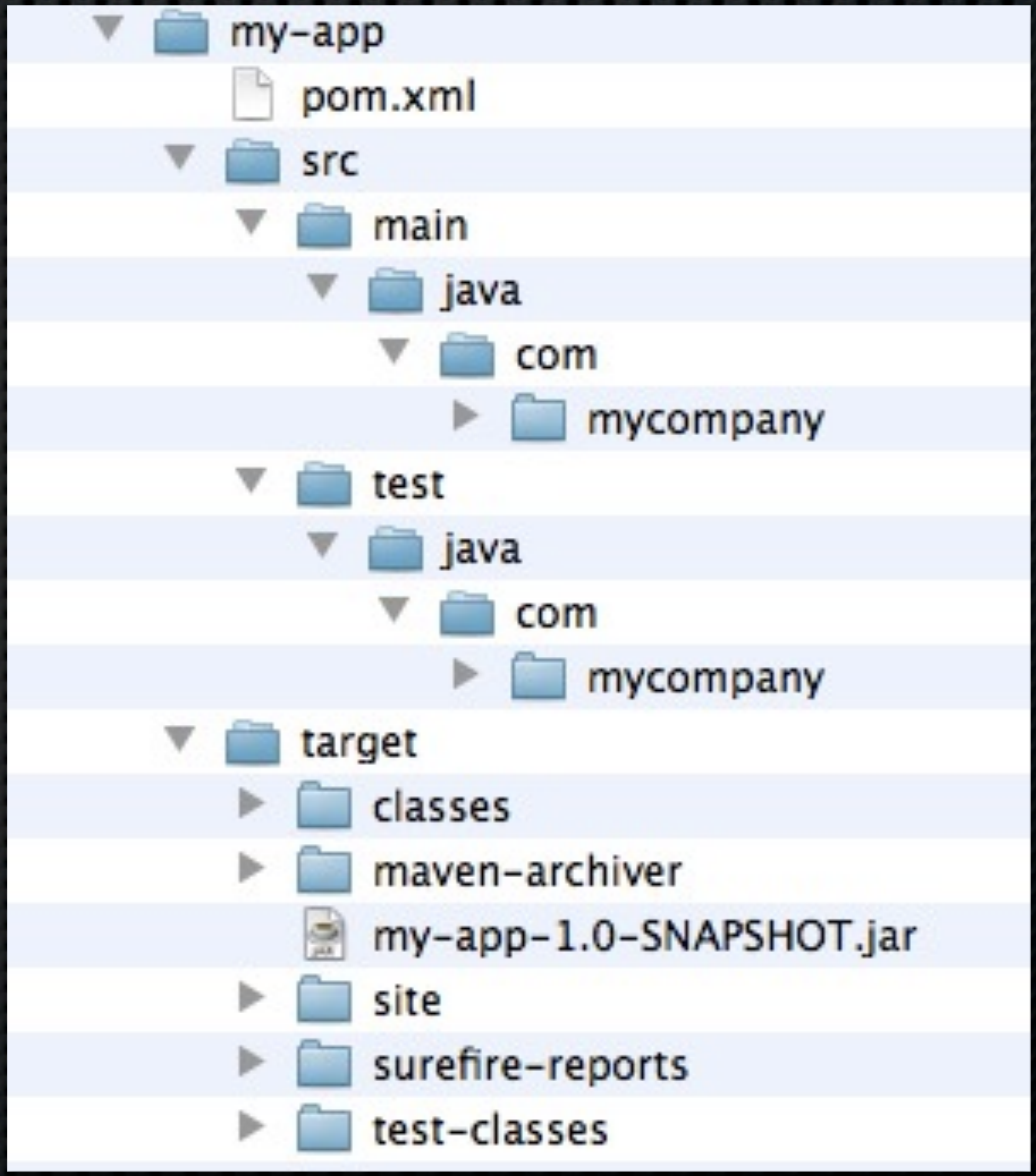
- Maven is a software tool for project management and build automation
- Yiddish for ‘accumulator of knowledge’
- Spinoff of Apache Jakarta Turbine Project

Maven



- Maven uses a construct known as a Project Object Model (POM) to describe the software project being built, its dependencies on other external modules and components, and the build order. It comes with pre-defined targets for performing certain well defined tasks such as compilation of code and its packaging
- Maven dynamically downloads Java libraries and Maven plug-ins from one or more repositories. Maven provides built-in support for retrieving files from the Maven 2 Central Repository and other Maven repositories, and can upload artifacts to specific repositories after a successful build. A local cache of downloaded artifacts acts as the primary means of synchronizing the output of projects on a local system

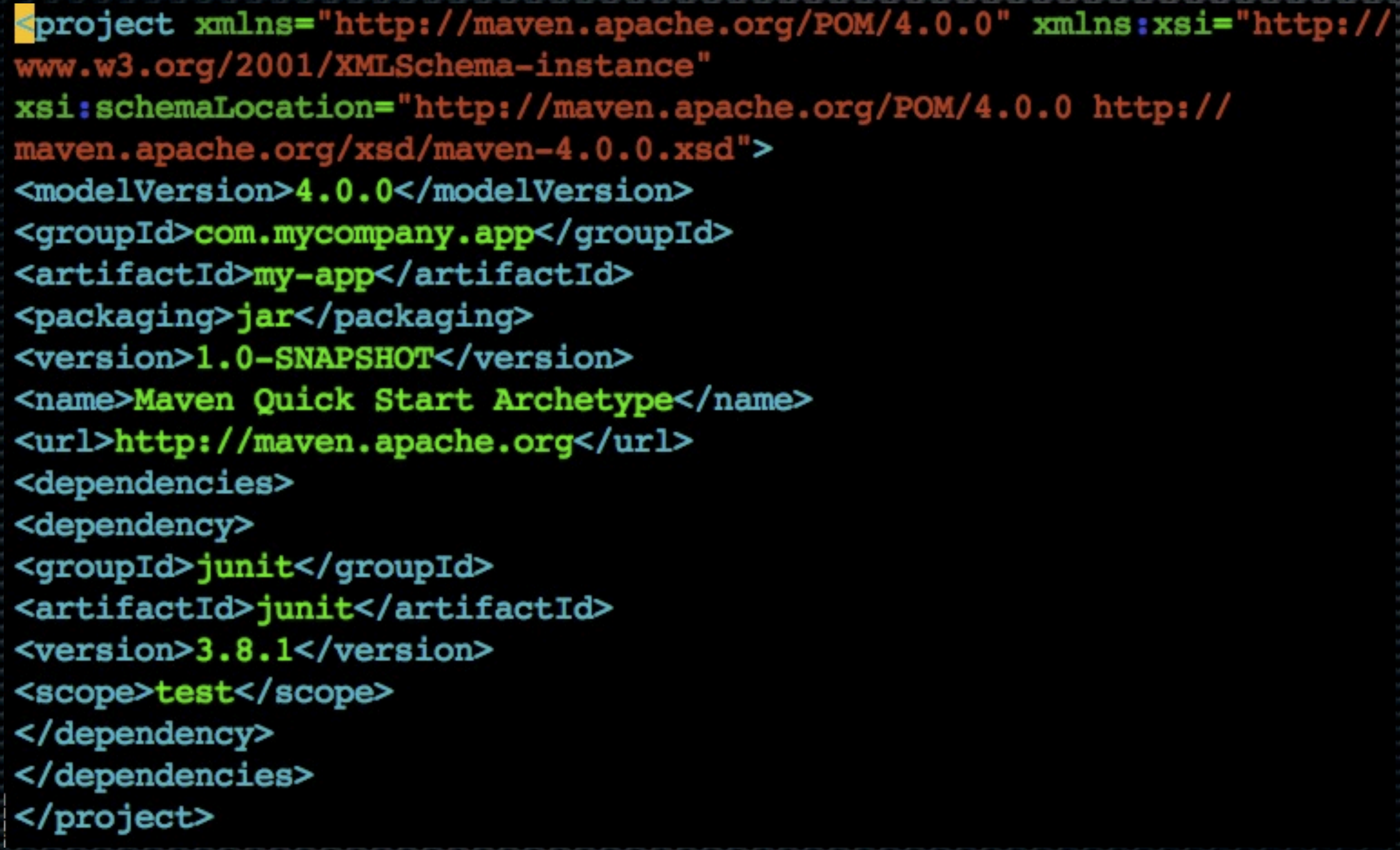
Maven sets up standard projects



```
▼ my-app
    pom.xml
  ▼ src
    ▼ main
      ▼ java
        ▼ com
          ▶ mycompany
    ▼ test
      ▼ java
        ▼ com
          ▶ mycompany
  ▼ target
    ▶ classes
    ▶ maven-archiver
      my-app-1.0-SNAPSHOT.jar
    ▶ site
    ▶ surefire-reports
    ▶ test-classes
```

```
mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

pom.xml belonging to example on last page



```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.mycompany.app</groupId>
<artifactId>my-app</artifactId>
<packaging>jar</packaging>
<version>1.0-SNAPSHOT</version>
<name>Maven Quick Start Archetype</name>
<url>http://maven.apache.org</url>
<dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>3.8.1</version>
<scope>test</scope>
</dependency>
</dependencies>
</project>
```

Maven

- downloads depencies: have to be connected to the internet

Emacs builds

- The venerable editor
- One of the two ‘One True Editors’
- Lisp-interpreter built in
- netrexx.el (by Arjan Bos) delivers build functionality

Emacs builds

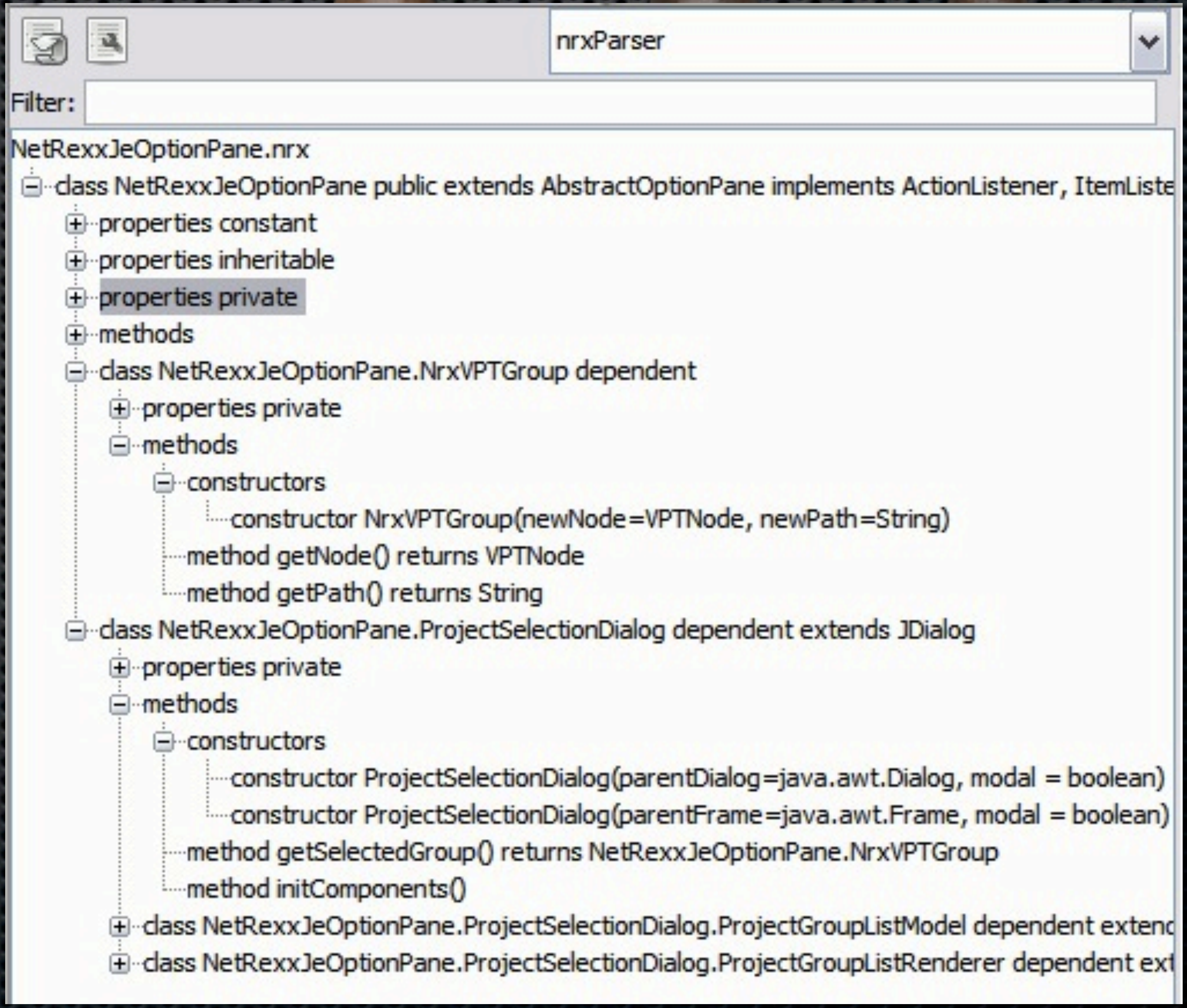
- Functionality:
- Syntax Coloring
- Press F9 to compile
- Keystroke lands emacs on first error line
- Boilerplate support
- Find corresponding **do** and **end**

Jedit Builds

- Original JEdit plugin - NetRexxJE - by Satguru P Srivastava
- NetRexxDE - David Requena
- NetRexxScript - Kermit Kiser

Jedit NetRexxDE

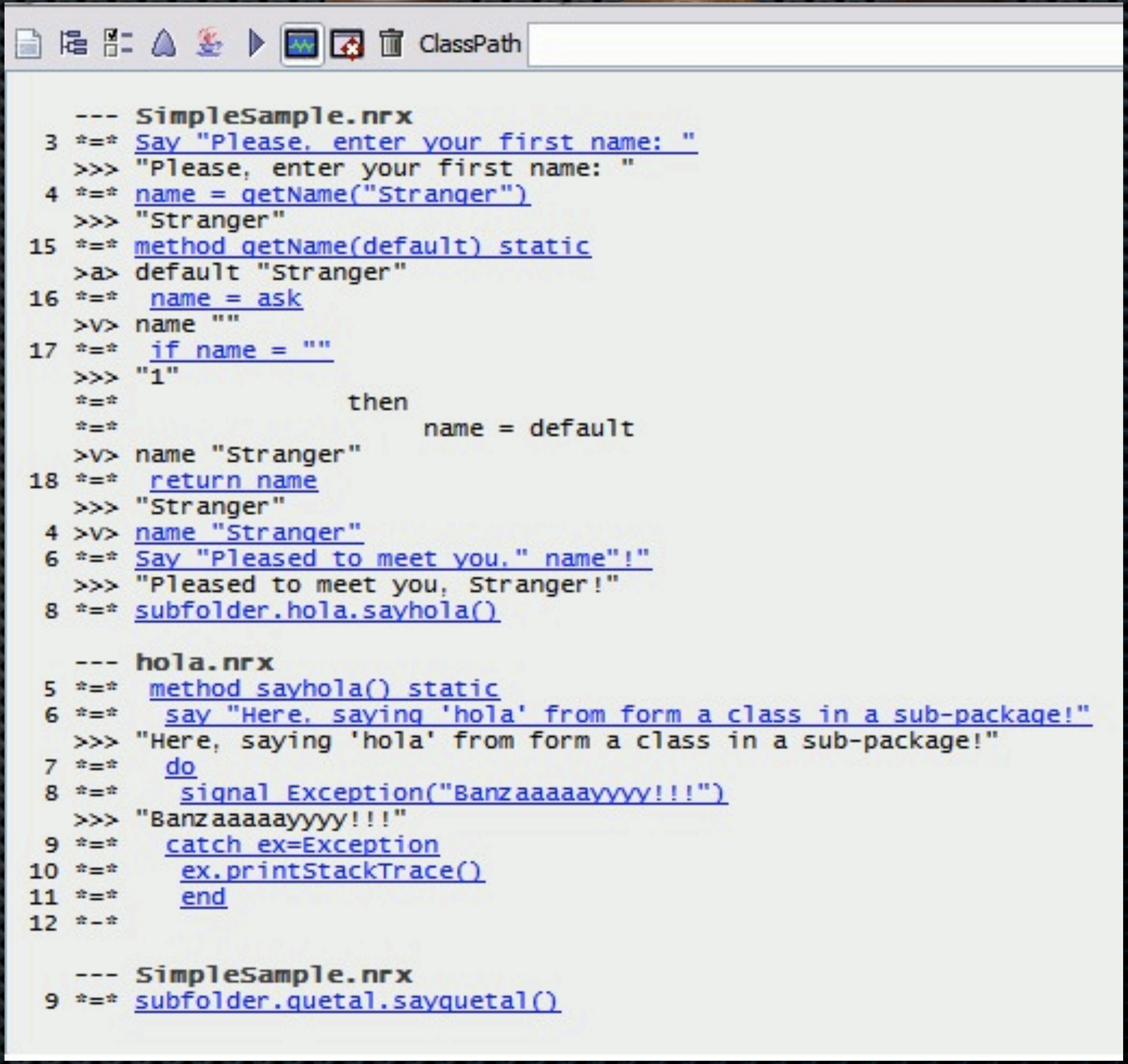
- Structure browser



```
NetRexxJeOptionPane.nrx
 ⊟ class NetRexxJeOptionPane public extends AbstractOptionPane implements ActionListener, ItemListe
     ⊞ properties constant
     ⊞ properties inheritable
     ⊞ properties private
     ⊞ methods
     ⊟ class NetRexxJeOptionPane.NrxVPTGroup dependent
         ⊞ properties private
         ⊟ methods
             ⊟ constructors
                 constructor NrxVPTGroup(newNode=VPTNode, newPath=String)
               method getNode() returns VPTNode
               method getPath() returns String
     ⊟ class NetRexxJeOptionPane.ProjectSelectionDialog dependent extends JDialog
         ⊞ properties private
         ⊟ methods
             ⊟ constructors
                 constructor ProjectSelectionDialog(parentDialog=java.awt.Dialog, modal = boolean)
                 constructor ProjectSelectionDialog(parentFrame=java.awt.Frame, modal = boolean)
               method getSelectedGroup() returns NetRexxJeOptionPane.NrxVPTGroup
               method initComponents()
         ⊞ class NetRexxJeOptionPane.ProjectSelectionDialog.ProjectGroupListModel dependent extend
         ⊞ class NetRexxJeOptionPane.ProjectSelectionDialog.ProjectGroupListRenderer dependent ext
```

Jedit NetRexxDE

- Trace Console



```
    --- SimpleSample.nrx
  3 *=* Say "Please, enter your first name: "
    >>> "Please, enter your first name: "
  4 *=* name = getName("Stranger")
    >>> "Stranger"
 15 *=* method getName(default) static
    >a> default "Stranger"
 16 *=*  name = ask
    >v> name ""
 17 *=*  if name = ""
    >>> "1"
    *=*               then
    *=*                    name = default
    >v> name "Stranger"
 18 *=*  return name
    >>> "Stranger"
  4 >v> name "Stranger"
  6 *=* Say "Pleased to meet you," name"!"
    >>> "Pleased to meet you, Stranger!"
  8 *=* subfolder.hola.sayhola()

    --- hola.nrx
  5 *=*  method sayhola() static
  6 *=*   say "Here, saying 'hola' from form a class in a sub-package!"
    >>> "Here, saying 'hola' from form a class in a sub-package!"
  7 *=*   do
  8 *=*    signal Exception("Banzaaaaayyyy!!!")
    >>> "Banzaaaaayyyy!!!"
  9 *=*   catch ex=Exception
 10 *=*    ex.printStackTrace()
 11 *=*    end
 12 *-*

    --- SimpleSample.nrx
  9 *=* subfolder.quetal.sayquetal()
```

Jedit NetRexxScript- Kermit Kiser

- Efficient execution of NetRexx scripts and macros in jEdit
- Optional caching of parsed scripts to improve performance
- Option to "preparse" all available scripts as a unit for performance (aka "greased lightning mode") and cross-script access support
- Option to suppress or allow trace output
- Options to execute a currently open NetRexx file including unsaved changes
- Optional dockable console window with command line to run scripts with parms and view say output or respond to ask input
- View say output with or without trace from the console window to aid debugging
- Option to "prefix" scripts automatically adds access to jEdit functions and variables
- Script parse errors are automatically highlighted in script source files and listed in the "Error List" as well as displayed in a console window
- User defined "classpath" option for making classes available to NetRexx scripts
- User defined "scriptpath" option for access to scripts outside of jEdit macro libraries
- Support for executing compiled NetRexx scripts or any "main" type Java class file as a macro or console command

Eclipse Builds

- One of the two leading IDE’s
- Eclipse plugin for NetRexx by Bill Finlason
- Syntax coloring
- Interpreting edit buffer
- Debugging code

Eclipse & Maven

- Launching Maven builds from within Eclipse
- • Dependency management for Eclipse build path based on Maven's pom.xml
- • Resolving Maven dependencies from the Eclipse workspace without installing to local Maven
- repository
- • Automatic downloading of the required dependencies and sources from the remote Maven
- repositories
- • Wizards for creating new Maven projects, pom.xml and to enable Maven support on existing
- projects
- • Quick search for dependencies in remote Maven repositories
- • Quick fixes in the Java editor for looking up required dependencies/jars by the class or package
- name
- • Integration with other Eclipse tools