Developing the OpenJDK Java Libraries with Eclipse JDT

EclipseCon 2019

Christoph Langer, SAP SE
cristoph.langer@sap.com

@TheRealClanger
RealCLanger
Introduction & Motivation

The OpenJDK Source Structure and Build System

Basics of an Eclipse JDT Project

Eclipse JDT Projects for Developing OpenJDK

History, Shortcomings and Future Vision
Introduction & Motivation

SAP’s Java Engagement

- J2EE stack since the early 2000's (SAP NetWeaver)
- Java is the technology for several products and services
- Sun/Oracle Source Code Licensee since the early days
- Provide and maintain a commercial Hotspot JVM, called SAPJVM, for SAP customers & products
- JVM ports to several platforms
- Created supportability tooling, leveraging Eclipse, aka SAP JVM Profiler
- Joined the OpenJDK project in 2011
- Made significant contributions to the OpenJDK, e.g. IBM Power and z/390 port, improved NullPointerException
- 3rd largest contributor to the OpenJDK
- Provide a downstream OpenSource build of OpenJDK since 2017 – SapMachine.io
Introduction & Motivation

My personal (Java-)History

- Joined the SAP JVM team in 2014 (between Java 8 and Java 9)
- Working in lots of areas, e.g. hotspot, class library, Eclipse based tooling
- Active member in the OpenJDK community
- Co-maintainer of JDK11 Update release
- Friend of Eclipse 😊
Introduction & Motivation

- When I joined the SAP JVM team
  - I found Eclipse projects for SAP’s JDK source tree and the Eclipse tools
  - Projects were at Java 8 level and did not fit well to OpenJDK and Java9+
  - Most of my colleagues were/are developing C/C++, so they cared for CDT

- In the OpenJDK project
  - It’s hard to find documentation for Java IDE support at all
  - There’s NetBeans support, e.g. https://mail.openjdk.java.net/pipermail/jdk-dev/2019-June/003067.html
  - There’s IntelliJ support, e.g. http://hg.openjdk.java.net/jdk/jdk/file/f7df2861be47/bin/idea.sh

  – **But where’s Eclipse support?**

  – It's here: https://github.com/RealCLanger/OpenJDKEclipseProjects
Introduction & Motivation

The OpenJDK Source Structure and Build System

Basics of an Eclipse JDT Project

Eclipse JDT Projects for Developing OpenJDK

History, Shortcomings and Future Vision
The OpenJDK Source Structure and Build System

Source is hosted as Mercurial repository on hg.openjdk.java.net: http://hg.openjdk.java.net/jdk/jdk/

There’s a lot of C/C++ code
- The virtual machine is implemented to a large extent in C++
- There’s native C/C++ code for JDK libraries

There’s also a lot of Java code
- The Java class libraries
- Implementation of tools, such as jcmd, javac etc.
- Code to support the build process
- Regression tests (e.g. jtreg)

A JVM/JDK abstracts OS/Hardware specifics, but it means that it has to be implemented platform specific in many areas
The OpenJDK Source Structure and Build System

<repository root>
  
  doc
  make
  src
    hotspot
    java.base
    java.desktop
    ...
  test
    lib
    hotspot
    jdk
    ...

java.base

  share
    native
      classes
  unix
    native
      classes
  linux
    native
      classes
  ...
  (e.g. aix, macosx, solaris, windows)
    native
      classes
The OpenJDK Source Structure and Build System

Configure based build
1. Run configure to create a build configuration (also dependent on OS, hardware, toolchain, options)
2. Run gnumake (e.g. „make images“) to build the JDK image or other targets, e.g. run tests

Build generates several artefacts, among them generated sources:

```
<build output dir>
  hotspot
  jdk
  images
  support
  gensrc
  ...
```
Introduction & Motivation

The OpenJDK Source Structure and Build System

**Basics of an Eclipse JDT Project**

Eclipse JDT Projects for Developing OpenJDK

History, Shortcomings and Future Vision
Basics of an Eclipse JDT Project

.project file - general project definition
- Project name
- Project natures
- Linked Resources

.classpath file - information about the project’s classpath
- source folders
- referenced projects
- referenced libraries

.settings folder – contains further property files to configure project aspects
- org.eclipse.jdt.core.prefs for JDT settings

Whether the project represents a (JPMS) module is determined by the Java compiler level and the presence of a module-info.java
Introduction & Motivation

The OpenJDK Source Structure and Build System

Basics of an Eclipse JDT Project

**Eclipse JDT Projects for Developing OpenJDK**

History, Shortcomings and Future Vision
Eclipse JDT Projects for Developing OpenJDK

OpenSource on GitHub: https://github.com/RealCLanger/OpenJDEclipseProjects

Instructions on how to set them up in Eclipse can be found in README.md

Contains branches according to JDK version, currently:

- **master** (head, to be used for OpenJDK14 and OpenJDK 13)
- **jdk11** (to be used for OpenJDK 11u LTS release)
- **jdk8** (to be used for OpenJDK 8, not well maintained)

Requirements

- Minimal setup
  - Clone of projects repository
  - Clone of an OpenJDK source tree, referenced as Linked Resource **JDK_SRC**
- Error-free IDE
  - OpenJDK build directory or gensrc artefacts, e.g. from SapMachine CI, Linked Resource **JDK_BUILD_DIR**
- With test support
  - JTREG binary, e.g. from SapMachine CI, Linked Resource **JTREG_PATH**
Eclipse JDT Projects for Developing OpenJDK

Import projects, depending on Operating System and development requirements

- Modules with platform specific implementation
  - Import projects from platform specific folder (e.g. aix, linux, macosx, solaris, windows)
  - Contains module java.base and all other modules which have OS specific implementation parts
- Modules with unix-specific implementation
  - Import projects from folder unix
- Modules with same implementation across all platforms
  - Import projects from folder common
- JVMCI and AOT related modules
  - Import from folder aotgraal
  - Not supported on every platform
- Tests
  - Import projects from subfolders of test
  - lib contains a common test library
  - Specific tests in folders hotspot, jdk and langtools
Eclipse JDT Projects for Developing OpenJDK

Demo?
Introduction & Motivation

The OpenJDK Source Structure and Build System

Basics of an Eclipse JDT Project

Eclipse JDT Projects for Developing OpenJDK

History, Shortcomings and Future Vision
History, Shortcomings and Future Vision

History

Several bugs needed to be fixed in both, Eclipse and OpenJDK to get IDE error-free, e.g.:

- **Eclipse**
  - [475996](#) - NullPointerException in ASTNode.checkInvocationArgument (ASTNode.java:340)
  - [540541](#) - Interactive compilation of project that depends on java.base as project fails to resolve java.base correctly
  - [541328](#) - Package from another modular project that is made visible via add-export project setting is not accessible
  - [547479](#) - Java 9+ (JPMS) Intermittent compilation error "The type ... is not accessible"

- **OpenJDK**
  - [JDK-8192961](#) Remove some double semicolons
  - [JDK-8211218](#) remove double semicolon in src/java.desktop/macosx/classes/sun/font/CFont.java
  - [JDK-8223553](#) Fix code constructs that do not compile with the Eclipse Java Compiler
  - [JDK-8224698](#) ConcurrentSkipListMap.java does not compile with the Eclipse Java Compiler
  - [JDK-8232370](#) Refactor some com.sun.jdi tests to enable IDE integration
History, Shortcomings and Future Vision

Shortcomings

- Changes in OpenJDK often cause the need for manual maintenance effort in the projects set
- Each test (folder) needs an own Eclipse project to be manually created, because test classes mostly don’t belong to a package (top level package)
- Importing projects with reference to source tree and build directory, choosing a certain platform seems still a bit cumbersome
- Running tests (jtreg framework) out of Eclipse is not well integrated
- It’s always a challenge to adopt hot changes in the Java language because Eclipse JDT support might not be ready
  – E.g. OpenJDK13 was released in September 2019, before Eclipse 2019-09 was released. And even Eclipse 2019-09 did not yet have support for Java 13
- Lack of users/maintainers
Future Vision

- Create a project generator that can be run by the build system and generates the exact set of projects, tailored to the build, including tests.
- Provide better jtreg runtime integration
- Get more users/adopters
  - Add Eclipse JDT project support to OpenJDK
  - There are plans to open up an [OpenJDK IDE tools group](https://example.com) with an own mailing list for IDE related topics -> use this forum to discuss and advertise Eclipse JDT
Questions?
Thank you.

Contact information:

Christoph Langer
christoph.langer@sap.com

@TheRealClanger
RealICLanger