A Java SE 9 és a JDK 9 újdonságai

Jeszenszky Péter
Debreceni Egyetem, Informatikai Kar
jeszenszky.peter@inf.unideb.hu

Utolsó módosítás: 2018. május 6.

A fejlesztés ütemterve

- A fejlesztés javasolt ütemterve (2015. május 5.):
  - *Proposed schedule for JDK 9*
    http://mail.openjdk.java.net/pipermail/jdk9-dev/2015-May/002172.html
- Az eredetileg 2016. szeptember 22-re kitűzött kiadást többször is elhalasztották.
  - Lásd:
    - *Proposed schedule change for JDK 9* (2015. december 1.)
      http://mail.openjdk.java.net/pipermail/jdk9-dev/2015-December/003149.html
      - Az új célkitűzés: 2017. március 23.
    - *Proposed schedule change for JDK 9* (2016. szeptember 13.)
      http://mail.openjdk.java.net/pipermail/jdk9-dev/2016-September/004887.html
      - Az új célkitűzés: 2017. július
    - *Proposed schedule change for JDK 9* (2017. május 30.)
      http://mail.openjdk.java.net/pipermail/jdk9-dev/2017-May/005864.html
      - Az új célkitűzés: 2017. szeptember 21.

Megjelenés

- A kiadás többszöri elhalasztása után végül 2017. szeptember 21-én jelent meg a JDK 9.
  - Lásd:
    - *JDK 9: General Availability* http://mail.openjdk.java.net/pipermail/announce/2017-September/000230.html
    - *Oracle Announces Java SE 9 and Java EE 8* https://www.oracle.com/corporate/pressrelease/java-se-9-and-ee-8-092117.html

Újdonságok

- Lásd:
  - *What's New in Oracle JDK 9*
    https://docs.oracle.com/javase/9/whatsnew/
  - *JDK 9 – Features*
    http://openjdk.java.net/projects/jdk9/

Változások

- Lásd:
  - *Java Platform, Standard Edition Oracle JDK 9 Migration Guide*
    https://docs.oracle.com/javase/9/migrate/

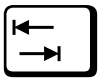
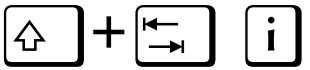
JShell (1)

- Egy interaktív parancssori eszköz, mely kiértékeli a begépelt deklarációkat, utasításokat, kifejezéseket és azonnal megjeleníti az eredményt.
  - Egyben egy API-t is biztosít ahhoz, hogy más alkalmazások is felhasználhassák ezt a funkciót.
    - Lásd a `jdk.jshell` modult. https://docs.oracle.com/javase/9/docs/api/jdk.jshell-summary.html
- A Java programozási nyelv tanulásához és Java nyelven történő prototípus készítéshez (*prototyping Java code*) fejlesztették ki.
- Úgynevezett *Read-Evaluate-Print Loop* (REPL) eszköz.
- Lásd:
  - *JEP 222: jshell: The Java Shell (Read-Eval-Print Loop)* http://openjdk.java.net/jeps/222
  - *Java Platform, Standard Edition Tools Reference – jshell* https://docs.oracle.com/javase/9/tools/jshell.htm
  - *Java Platform, Standard Edition Java Shell User's Guide* https://docs.oracle.com/javase/9/jshell/introduction-jshell.htm

JShell (2)

- Az alábbi Java **kódrész**ek (***snippet***) hajthatók végre:
  - Kifejezés
  - Utasítás
  - Osztály deklaráció
  - Interfész deklaráció
  - Metódus deklaráció
  - Változó deklaráció
  - Import deklaráció

JShell (3)

- Kényelmi funkciók:
  - Szerkeszthető parancs előzmények
  - Automatikus kiegészítés ( billentyű)
    - Nem feloldható azonosító esetén a  billentyűkombináció lenyomására a *JShell* lehetséges importokat javasol, melyek feloldják az azonosítót.
  - Elhagyható a pontosvessző utasítások végéről
  - Előre definiált importok
  - …

JShell (4)

- A környezet vezérlésére és információk megjelenítésére olyan parancsok szolgálnak, melyek egy `/` karakterrel kezdődnek.
  - Néhány parancs:
    - `/exit`
    - `/help` [<*parancs*>|<*téma*>] vagy `/?` [<*parancs*>|<*téma*>]
    - `/imports`
    - …

JShell (5)

- IDE integráció:
  - **Eclipse:** Legjobb tudomásom szerint jelenleg nincs a *JShell*-t az IDE-be integráló bővítmény.
    - A *JShell* használható például a *TM Terminal* bővítményen keresztül. https://marketplace.eclipse.org/content/tm-terminal
  - **IntelliJ IDEA**: *Tools* → *JShell Console...*
  - **NetBeans**: *Tools* → *Open Java Platform Shell*
    - Csak az innen letölthető kiadásban: http://bits.netbeans.org/download/trunk/nightly/latest/

Változások a nyelvben (1)

- Lásd:
  - *JEP 213: Milling Project Coin*
    http://openjdk.java.net/jeps/213
  - *Java Language Changes for Java SE 9*
    https://docs.oracle.com/javase/9/language/toc.htm

Változások a nyelvben (2)

- **Az aláhúzásjel nem érvényes azonosító:**
  - A Java SE 8-ban figyelmeztetést eredményezett.
- **Privát interfész metódusok**:
  - A nem absztrakt metódusokhoz hasonlóan metódus törzzsel rendelkeznek.
  - A nem absztrakt (`default`, `static`) metódusok közötti kódmegosztásra szolgálnak.
  - Nem öröklődnek és nem írhatók felül.
  - Lásd: *Exploring Java 9. Private Interface Methods* https://www.polidea.com/blog/Exploring-Java-9Private-Interface-Methods/

A folyamat API korszerűsítése (1)

- Az operációs rendszer folyamatok vezérlésére és kezelésére szolgáló API továbbfejlesztése.
  - Az API-t a `java.lang.Process` és `java.lang.ProcessBuilder` osztályok valamint a `java.lang.ProcessHandle` és `java.lang.ProcessHandle.Info` interfészek alkotják, az utóbbi kettő a Java SE 9-ben bevezetett újdonság.
- Lásd:
  - *JEP 102: Process API Updates* http://openjdk.java.net/jeps/102
  - *Java Platform, Standard Edition Core Libraries – Process API* https://docs.oracle.com/javase/9/core/process-api1.htm

A folyamat API korszerűsítése (2)

- `java.lang.ProcessHandle` https://docs.oracle.com/javase/9/docs/api/java/lang/ProcessHandle.html
  - Folyamatok azonosítására és vezérlésére szolgál.
- `java.lang.ProcessHandle.Info` https://docs.oracle.com/javase/9/docs/api/java/lang/ProcessHandle.Info.html
  - Egy folyamatról szolgáltat információkat.

A folyamat API korszerűsítése (3)

- Példa:

```
final String na = "<not available>";

ProcessHandle handle = ProcessHandle.current();
System.out.printf("Process ID: %d%n", handle.pid()); // 7326

ProcessHandle.Info info = handle.info();
System.out.printf("Command: %s%n",
    info.command().orElse(na)); // /usr/lib/jvm/java-9-oracle/bin/java
System.out.printf("Command line: %s%n", info.commandLine()
    .orElse(na)); // /usr/lib/jvm/java-9-oracle/bin/java ProcessExample
System.out.printf("Start instant: %s%n", info.startInstant()
    .map(instant -> instant.toString())
    .orElse(na)); // 2018-02-13T10:12:22.090Z
System.out.printf("Total CPU duration: %s%n", info.totalCpuDuration()
    .map(duration -> duration.toString())
    .orElse(na)); // PT0.16S
System.out.printf("User: %s%n", info.user().orElse(na)); // jeszy
```

A folyamat API korszerűsítése (4)

- Példa:

```
Process p = new ProcessBuilder("echo", "Hello, world!")
   .inheritIO()
   .start();
CompletableFuture<Process> onExit = p.onExit();
onExit.get();
onExit.thenAccept(process ->
   System.out.printf("PID %d terminated%n", process.pid()));
```

Kényelmi gyártó metódusok kollekciókhoz (1)

- Statikus gyártó metódusok kisszámú elemből álló kollekciók és asszociatív tömbök létrehozásához.
- Lásd a `java.util.List`, `java.util.Set` és `java.util.Map` interfészek `of()` metódusait valamint a `java.util.Map` interfész `ofEntries()` metódusát.
- Szerkezetileg megváltoztathatatlan (*immutable*) példányokat hoznak létre.
- Lásd:
  - *JEP 269: Convenience Factory Methods for Collections* *http://openjdk.java.net/jeps/269*
  - *Java Platform, Standard Edition Core Libraries – Creating Immutable Lists, Sets, and Maps* https://docs.oracle.com/javase/9/core/creating-immutable-lists-sets-and-maps.htm

Kényelmi gyártó metódusok kollekciókhoz (2)

- Példa:

```
// JDK 8:
Set<String> beatles = new HashSet<String>(
   Arrays.asList("John", "Paul", "Ringo", "George"));
beatles = Collections.unmodifiableSet(beatles);
```

```
// JDK 9:
Set<String> beatles = Set.of("John", "Paul", "Ringo",
   "George");
```

Kényelmi gyártó metódusok kollekciókhoz (3)

- Példa:

```
// JDK 8:
Map<Integer, String> redirects = new HashMap<Integer, String>();
redirects.put(300, "Multiple Choices");
redirects.put(301, "Moved Permanently");
redirects.put(302, "Found");
redirects.put(303, "See Other");
redirects.put(304, "Not Modified");
redirects = Collections.unmodifiableMap(redirects);
```

```
// JDK 9:
Map<Integer, String> redirects2 = Map.of(300, "Multiple Choices",
    301, "Moved Permanently",
    302, "Found",
    303, "See Other",
    304, "Not Modified");
```

Kényelmi gyártó metódusok kollekciókhoz (4)

- Példa: (folytatás)

```
// JDK 9:
import static java.util.Map.entry;

Map<Integer, String> redirects = Map.ofEntries(
   entry(300, "Multiple Choices"),
   entry(301, "Moved Permanently"),
   entry(302, "Found"),
   entry(303, "See Other"),
   entry(304, "Not Modified"));
```

Inkubáló modulok

- Arra szolgálnak, hogy nem végleges, inkubáló (*incubating*) API-kat adjanak a fejlesztők kezébe.
- Egy inkubáló API egy olyan API, melynél kívánatos a szabványosítás vagy véglegesítés elhalasztása a következő kiadásra az API-val kapcsolatos további tapasztalatok szerzése céljából.
- Korlátozott egy API inkubációs élettartama: az API szabványosításra vagy véglegesítésre kerül a következő kiadásban, vagy pedig eltávolításra kerül.
- Az inkubáló modulok neve a `jdk.incubator.` előtaggal kezdődik.
  - Példa: `jdk.incubator.httpclient` https://docs.oracle.com/javase/9/docs/api/jdk.incubator.httpclient-summary.html
- Lásd:
  - *JEP 11: Incubator Modules* http://openjdk.java.net/jeps/11

HTTP/2 kliens (1)

- A HTTP/2 és a WebSocket technológiákat implementáló új HTTP kliens API definiálása, mely helyettesítheti a `java.net.HttpURLConnection` osztályt.
- Az API-t a `jdk.incubator.httpclient` inkubáló modul szolgáltatja: https://docs.oracle.com/javase/9/docs/api/jdk.incubator.httpclient-summary.html
- A modul nem kerül alapértelmezésben feloldásra sem fordítási, sem végrehajtási időben!
  - A modul feloldásához a `javac` és `java` parancsokhoz meg kell adni az `--add-modules jdk.incubator.httpclient` parancssori opciót.
- Lásd:
  - *JEP 110: HTTP/2 Client (Incubator)* http://openjdk.java.net/jeps/110
  - *JDK HTTP Client – Examples and Recipes* https://blogs.oracle.com/java/jdk-http-client

HTTP/2 kliens (2)

- Példa:

```
import jdk.incubator.http.HttpClient;
import jdk.incubator.http.HttpRequest;
import jdk.incubator.http.HttpResponse;
import jdk.incubator.http.HttpResponse.BodyHandler;

HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
    .uri(URI.create("http://freegeoip.net/json/"))
    .GET()
    .build();
HttpResponse<String> response = client.send(request,
    BodyHandler.asString());
System.out.println(response.body());
// {"ip":"193.6.168.26","country_code":"HU",
// "country_name":"Hungary","region_code":"HB",
// "region_name":"Hajdú-Bihar","city":"Debrecen",
// "zip_code":"4032","time_zone":"Europe/Budapest",
// "latitude":47.7675,"longitude":21.24,"metro_code":0}
```

Nemzetköziesítés

- **UTF-8 `.properties` állományok**:
  - Tulajdonság erőforrás csomagok (`java.util.PropertyResourceBundle`) betöltéséhez az ISO-8859-1 helyett az UTF-8 karakterkódolás használata.
  - Lásd:
    - `java.util.PropertyResourceBundle` https://docs.oracle.com/javase/9/docs/api/java/util/PropertyResourceBundle.html
    - *JEP 226: UTF-8 Properties Files* http://openjdk.java.net/jeps/226
- **Unicode 8.0 támogatás**:
  - Az Unicode 8.0 támogatása. A JDK 8 az Unicode 6.2 szabványt támogatta.
  - Lásd: *JEP 267: Unicode 8.0* http://openjdk.java.net/jeps/267

Javadoc

- **HTML5 Javadoc**:
  - A javadoc eszköznek megadható `-html5` opció HTML5 kimenetet eredményez.
  - Lásd: *JEP 224: HTML5 javadoc* http://openjdk.java.net/jeps/224
- **Javadoc keresés**:
  - Funkció biztosítása a dokumentációban való kereséshez.
  - A következők indexelése és kereshetősége: modulok, csomagok, típusok és tagok nevei, valamint az `@index` szövegközi címkével megjelölt keresőkifejezések.
  - Lásd: *JEP 225: Javadoc Search* http://openjdk.java.net/jeps/225
- **Modulrendszer**:
  - A `javadoc` eszköz támogatja a dokumentációs megjegyzéseket modul deklarációkban.
  - Új parancssori opciók a `javadoc` eszközhöz a modulok kezeléséhez.
  - Lásd: *Module System* https://docs.oracle.com/javase/9/javadoc/javadoc.htm

Karakterláncok ábrázolása

- **Tömör karakterláncok**:
  - Karakterláncok helytakarékosabb belső ábrázolása.
  - Korábban a `java.lang.String` osztály egy `char` tömbben tárolta a karaktereket.
    - A `char` típus tárolási mérete 2 bájt.
  - Az új belső ábrázolásmód egy bájt tömböt és egy, a karakterkódolást jelző bájtot használ.
  - Lásd: *JEP 254: Compact Strings*
    http://openjdk.java.net/jeps/254

Platform naplózó API és szolgáltatás

- Minimális naplózó API biztosítása a platform osztályok számára üzenetek naplózásához.
- Lehetővé teszi egy könyvtár vagy alkalmazás számára a platform naplóüzenetek továbbítását a választása szerinti naplózó keretrendszernek.
- Nem általános célú naplózó interfész, hanem kifejezetten a JDK osztályok számára készült!
- Alapértelmezésben a `java.util.logging` csomag használata a naplózáshoz.
- Lásd:
  - *JEP 264: Platform Logging API and Service* http://openjdk.java.net/jeps/264
  - `java.lang.System.Logger` https://docs.oracle.com/javase/9/docs/api/java/lang/System.Logger.html
  - `java.lang.System.LoggerFinder` https://docs.oracle.com/javase/9/docs/api/java/lang/System.LoggerFinder.html

Java fordító

- **Fordítás régebbi platform verziókra**:
  - A `javac` program két parancssori opciót (`-source` és `-target`) biztosít a Java nyelv a fordító által elfogadott verziójának és az általa előállított `.class` állományok verziójának megadásához.
    - Azonban a `javac` alapértelmezésben a legújabb platform API-kat használja a fordításhoz.
    - A lefordított program így véletlenül olyan API-kat használhat, melyek csak a platform aktuális verziójában állnak rendelkezésre. Az ilyen programok nem futtathatók a platform régebbi verzióin!
  - Az új `--release` parancssori opció úgy konfigurálja a fordítót, hogy az olyan `.class` állományokat hoz létre, melyek működni fognak az adott platform verzió implementációin.
    - Paraméterként 6, 7, 8 vagy 9 adható meg.
  - Lásd:
    - *JEP 247: Compile for Older Platform Versions* http://openjdk.java.net/jeps/247
    - *Java Platform, Standard Edition Tools Reference – javac* https://docs.oracle.com/javase/9/tools/javac.htm

Java virtuális gép

- **Egyesített Java virtuális gép naplózás**:
  - Közös naplózási rendszer bevezetése a Java virtuális gép összes komponenséhez.
  - Új parancssori opció (`-Xlog`) bevezetése a naplózás vezérléséhez.
    - Példa: `-Xlog:help`, `-Xlog:disable`, `-Xlog`, …
  - Lásd:
    - *JEP 158: Unified JVM Logging* http://openjdk.java.net/jeps/158
    - *Java Platform, Standard Edition Tools Reference – java – Enable Logging with the JVM Unified Logging Framework* https://docs.oracle.com/javase/9/tools/java.htm#GUID-BE93ABDC-999C-4CB5-A88B-1994AAAC74D5

Szemétgyűjtés (1)

- **A G1 az alapértelmezett szemétgyűjtő**:
  - Szerver gépeken a *Garbage-First* (G1) szemétgyűjtő használata az alapértelmezés.
    - Szerver gép: legalább 2 fizikai processzorral és legalább 2 GB fizikai memóriával rendelkező számítógép.
  - A G1 teljesítményét növelő és használatát megkönnyítő javítások.
    - Például bizonyos beállítások automatikus meghatározása, melyeket korábban manuálisan kellett beállítani az optimális eredményhez.
  - Lásd:
    - *JEP 248: Make G1 the Default Garbage Collector* http://openjdk.java.net/jeps/248
    - *Java Platform, Standard Edition – HotSpot Virtual Machine Garbage Collection Tuning Guide* https://docs.oracle.com/javase/9/gctuning/

Szemétgyűjtés (2)

- **Egyesített GC naplózás**:
  - A szemétgyűjtéshez kötődő naplózás újraimplementálása a JEP 158-ban bevezetett Java virtuális gép naplózó keretrendszer felhasználásával.
  - Lásd: *JEP 271: Unified GC Logging* http://openjdk.java.net/jeps/271

Szemétgyűjtés (3)

- **Egyesített GC naplózás**:
  - Példa:

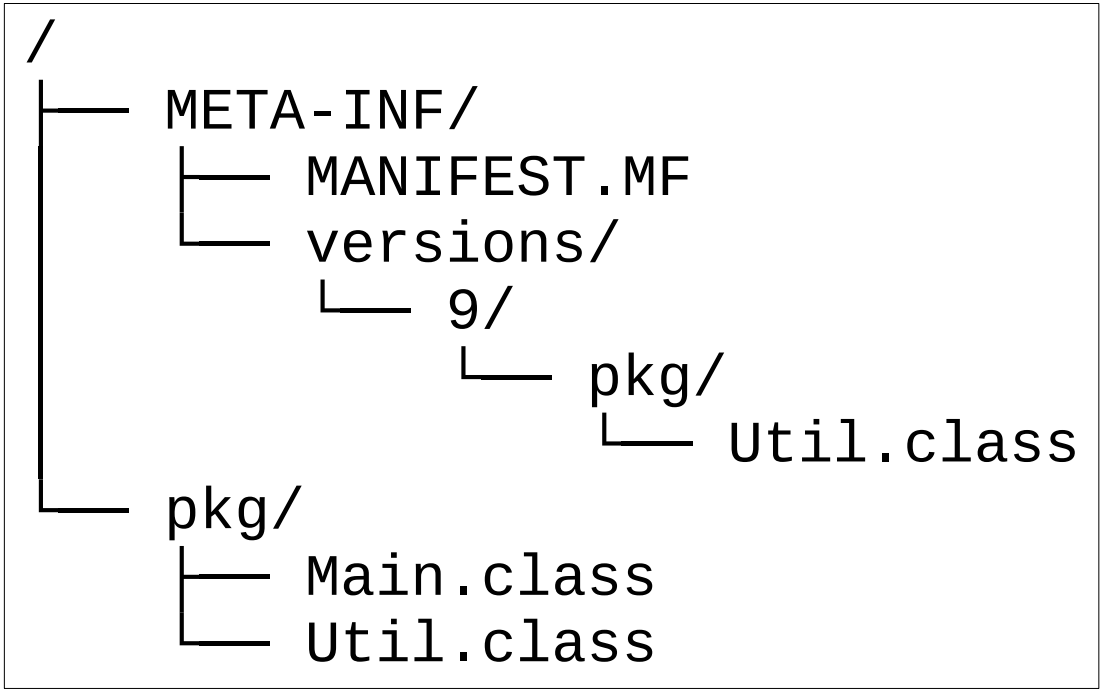
```
$ java -Xlog:gc Main
$ java -Xlog:gc=debug Main
$ java -Xlog:gc=trace:file=gctrace.txt Main
$ java -Xlog:gc* Main
```

Multi-Release JAR (MRJAR) állományok (1)

- A JAR állományformátum kiterjesztése, mely lehetővé teszi, hogy egy JAR állományban egyidejűleg létezzenek `.class` állományok a Java platform különböző kiadásaihoz készült verziói.
- Egy MRJAR állomány fő szakaszában meg kell adni az alábbi attribútumot:
  - `Multi-Release: true`
- Lásd:
  - *JEP 238: Multi-Release JAR Files* http://openjdk.java.net/jeps/238
  - *Java Platform, Standard Edition Tools Reference – jar* https://docs.oracle.com/javase/9/tools/jar.htm
  - *JAR File Specification – Multi-release JAR files* https://docs.oracle.com/javase/9/docs/specs/jar/jar.html

Multi-Release JAR (MRJAR) állományok (2)

- Példa MRJAR állomány könytárszerkezetére:
  - Egy, az MRJAR állományokat nem támogató JDK számára csak a gyökérkönyvtárban lévő osztályok és erőforrások láthatóak!



Könyvtárszerkezet

- Lásd:
  - *Java Platform, Standard Edition Installation Guide – Installed Directory Structure of JDK and JRE* https://docs.oracle.com/javase/9/install/installed-directory-structure-jdk-and-jre.htm
  - *JEP 220: Modular Run-Time Images* http://openjdk.java.net/jeps/220

Appletek elavulttá tétele

- Az applet API és a Java böngésző bővítmény elavultnak jelölése.
  - Lásd:
    - *Moving to a Plugin-Free Web* https://blogs.oracle.com/java-platform-group/moving-to-a-plugin-free-web
    - *JEP 289: Deprecate the Applet API* http://openjdk.java.net/jeps/289
- A *Java Web Start* használata a javasolt alternatíva.
  - Lásd:
    - *Java Platform, Standard Edition Deployment Guide – Java Web Start Technology* https://docs.oracle.com/javase/9/deploy/java-web-start-technology.htm
    - *Migrating Java Applets to Java Web Start and JNLP* https://docs.oracle.com/javase/9/deploy/migrating-java-applets-jnlp.htm

Eltávolított eszközök

- *Java DB*
  - Lásd: *Remove Java DB from JDK 9*
    https://bugs.openjdk.java.net/browse/JDK-8080959
- `native2ascii`
  - Lásd: *JEP 226: UTF-8 Property Resource Bundles*
    http://openjdk.java.net/jeps/226

A jövőben eltávolításra kerülő elavult modulok

- A Java EE és CORBA technológiákat tartalmazó modulok elavultnak és a jövőben eltávolításra kerülőnek jelölése:
  - `java.activation` (*JavaBeans Activation Framework*), `java.xml.bind` (JAXB), `java.xml.ws` (JAX-WS), …
- Ezek a modulok fordításnál és végrehajtásnál nem kerülnek feloldásra!
  - A modulok feloldásához a `javac` és `java` parancsokhoz meg kell adni az `--add-modules` parancssori opciót.
- Lásd: *JEP 320: Remove the Java EE and CORBA Modules* http://openjdk.java.net/jeps/320

Project Jigsaw (1)

- A projekt célja egy szabványos modulrendszer tervezése és implementálása volt a Java SE platformhoz, valamint ennek alkalmazása magára a platformra és a JDK-ra.
- Honlap: http://openjdk.java.net/projects/jigsaw/

Project Jigsaw (2)

- Elsődleges célok:
  - Megkönnyíteni a fejlesztők számára könyvtárak és nagy alkalmazások létrehozását és karbantartását.
  - Általában a Java SE platform implementációk, főleg a JDK biztonságának és karbantarthatóságának javítása.
  - Lehetővé tenni az alkalmazások teljesítményének növelését.
  - Lehetővé tenni a Java SE platform és a JDK kis számítási teljesítményű eszközökre való szabását.

Project Jigsaw (3)

- A projekt céljainak eléréséhez az alábbi két alapvető lehetőség biztosítása:
  - **Megbízható konfigurálás** (***reliable configuration***): a törékeny, hibára hajlamos *class path* mechanizmus helyettesítése a programkomponensek számára egy olyan eszközzel, melynek révén explicit módon deklarálhatják az egymástól való függéseiket.
    - Lásd: JAR pokol (*JAR hell*)
  - **Erős egységbezárás** (***strong encapsulation***): lehetővé tenni egy komponens számára annak deklarálását, hogy mely nyilvános típusai elérhetők más komponensek számára.

Project Jigsaw történet

- A projekt 2008 augusztusában indult.
- Eredetileg a JDK 7-hez szánták, majd előbb a JDK 8-ra, végül a JDK 9-re halasztották el a kivitelezést.
- 2014-ben indult a tervezés és megvalósítás a Java 9-hez.
- A Java platform modulrendszert meghatározó JSR 376 specifikációt 2014 decemberében hagyta jóvá a JCP végrehajtó bizottsága.
- A projekt fejlesztése nem a várt ütemben haladt, ezért a JDK 9 eredetileg 2016. szeptember 22-re kitűzött kiadást többször is elhalasztották.
- A projekttel kapcsolatos munka 2017 júliusában fejeződött be, az eredmény a 2017. szeptember 21-én megjelent JDK 9 részeként elérhető általános használatra.
- Lásd:
  - *Project Jigsaw: Context & History* http://openjdk.java.net/projects/jigsaw/history
  - *Project Jigsaw – Development history* http://openjdk.java.net/projects/jigsaw/

Project Jigsaw specifikációk

- JEP-ek:
  - *JEP 200: The Modular JDK* http://openjdk.java.net/jeps/200
  - *JEP 201: Modular Source Code* http://openjdk.java.net/jeps/201
  - *JEP 220: Modular Run-Time Images* http://openjdk.java.net/jeps/220
  - *JEP 260: Encapsulate Most Internal APIs* http://openjdk.java.net/jeps/260
  - *JEP 261: Module System* http://openjdk.java.net/jeps/261
  - *JEP 282: jlink: The Java Linker* http://openjdk.java.net/jeps/282
- JSR:
  - *JSR 376: Java Platform Module System*
    - http://openjdk.java.net/projects/jigsaw/spec/
    - https://jcp.org/en/jsr/detail?id=376

Modul fogalma (1)

- Elég szorosan összetartozó csomagok egy modulba csoportosíthatók.
- Egy modul kód és adatok egy névvel rendelkező, önleíró gyűjteménye. A kódja csomagokba van szervezve, melyek típusokat (osztályokat és interfészeket) tartalmaznak. Adatként erőforrásokat és más típusú statikus információkat tartalmaz.
  - Lásd: Mark Reinhold. *The State of the Module System.* http://openjdk.java.net/projects/jigsaw/spec/sotms/

Modul fogalma (2)

- Egy modul exportáltként jelölheti meg néhány vagy minden csomagját, ami azt jelenti, hogy a típusaik hozzáférhetők a modulon kívüli kódból.
  - Egy felső szintű típus akkor, és csak akkor hozzáférhető a deklaráló modulon kívülről, ha nyilvánosnak deklarált és egy exportált csomag tagja.
  - Ha egy csomagot nem exportál egy modul, akkor csak a modulon belüli kód számára hozzáférhetők a típusai.
- Ha egy modul egy másik modul által exportált csomagokhoz szeretne hozzáférni, akkor explicit módon függenie kell tőle.

Modul deklarációk (1)

- James Gosling, Bill Joy, Guy Steele, Gilad Bracha, Alex Buckley, Daniel Smith. *The Java Language Specification – Java SE 9 Edition*. 7 August 2017. https://docs.oracle.com/javase/specs/jls/se9/html/
  - Lásd a következő részt: *7.7. Module Declarations* https://docs.oracle.com/javase/specs/jls/se9/html/jls-7.html

Modul deklarációk (2)

- Egy modul deklaráció egy nevesített modult határoz meg.
- Egy olyan modulnevet vezet be, mely modulok közötti kapcsolatok kifejezésére használható más modulok deklarációiban.
- Egy modulnév egy vagy több Java azonosítóból áll, melyeket `.` karakterek választanak el.
- Egy modul deklaráció direktívái határozzák meg a modul függését más moduloktól (`requires`), a más modulok számára elérhetővé tett csomagokat (`export` és `opens`), az általa felhasznált (`uses`) és nyújtott (`provides`) szolgáltatásokat.

Modul deklarációk (3)

- A modul deklarációt megelőzhetik import deklarációk.
  - Lehetővé teszik, hogy a modul deklarációban a modul vagy más modulok csomagjainak típusaira és a típusok statikus tagjaira az egyszerű nevükkel hivatkozhassunk.

Modul deklarációk (4)

- Példa:

```
module java.prefs {
   requires java.xml;

   exports java.util.prefs;

   uses java.util.prefs.PreferencesFactory;
}
```

Modul deklarációk (5)

- Példa:

```
module java.sql {
    requires transitive java.logging;
    requires transitive java.xml;

    exports java.sql;
    exports javax.sql;
    exports javax.transaction.xa;

    uses java.sql.Driver;
}
```

Modul deklarációk (6)

- Példa:

```
module java.logging {
   exports java.util.logging;

   provides jdk.internal.logger.DefaultLoggerFinder with
      sun.util.logging.internal.LoggingProviderImpl;
}
```

Modul deklarációk (7)

- **`requires`**:
  - A `requires` direktíva egy olyan modult határoz meg, melytől az aktuális modul függ.
  - A `java.base` modul kivételével minden modul implicit módon függ a `java.base` modultól.
  - A `requires` kulcsszót követheti a `transitive` módosító.
    - Az aktuális modultól függő modulok implicit módon függenek a `requires transitive` direktívával meghatározott modultól.
  - A `requires` kulcsszót követheti a `static` módosító.
    - Ez azt jelenti, hogy a függőség fordítási időben kötelező, futási időben opcionális.

Modul deklarációk (8)

- **`exports`**:
  - Az `export` direktíva egy, az aktuális modul által exportált csomag nevét határozza meg.
  - Más modulok számára a csomag `public` és `protected` típusai, valamint azok `public` és `protected` tagjai hozzáférhetők fordítási és futási időben.
    - Reflektív hozzáférés is engedélyezett ezekhez a típusokhoz és tagjaikhoz.

Modul deklarációk (9)

- **`uses`**:
  - A `uses` direktíva egy olyan szolgáltatást határoz meg, melyhez az aktuális modul szolgáltatókat találhat a `java.util.ServiceLoader` osztály révén.
- **`provides`**:
  - A `provides` direktíva egy olyan szolgáltatást határoz meg, melyhez a `with` záradék sorol fel egy vagy több szolgáltatót a `java.util.ServiceLoader` osztályhoz.

java.util.ServiceLoader (1)

- Szolgáltatások implementációinak betöltésére szolgáló lehetőség.
- Egy szolgáltatás egy olyan közismert interfész vagy (absztrakt) osztály, melyhez 0, 1 vagy több szolgáltató létezik.
- Egy szolgáltató (*service provider*, vagy röviden *provider*) egy olyan osztály, mely a közismert interfészt implementálja vagy alosztálya a közismert (absztrakt) osztálynak.
- Az alkalmazások kódja csak a szolgáltatásra hivatkozik, nem pedig a szolgáltatókra.
- Lásd: `java.util.ServiceLoader` https://docs.oracle.com/javase/9/docs/api/java/util/ServiceLoader.html

java.util.ServiceLoader (2)

- Példa:

```
import java.sql.Driver;
import java.util.ServiceLoader;
import static java.util.ServiceLoader.Provider;

ServiceLoader<Driver> loader =
   ServiceLoader.load(Driver.class);
for (Driver driver : loader) {
   System.out.println(driver.getClass().getName());
}
// org.hsqldb.jdbc.JDBCDriver
// org.apache.derby.jdbc.AutoloadedDriver
// org.sqlite.JDBC
```

java.util.ServiceLoader (3)

- Példa:

```
import java.util.ServiceLoader;
import static java.util.ServiceLoader.Provider;

ServiceLoader<java.security.Provider> loader =
    ServiceLoader.load(java.security.Provider.class);
loader.stream()
    .map(Provider::type)
    .map(Class::getName)
    .forEach(System.out::println);
// sun.security.smartcardio.SunPCSC
// com.sun.deploy.security.MozillaJSSProvider
// sun.security.ec.SunEC
// com.sun.security.sasl.gsskerb.JdkSASL
// sun.security.pkcs11.SunPKCS11
// sun.security.jgss.SunProvider
// org.jcp.xml.dsig.internal.dom.XMLDSigRI
// sun.security.provider.certpath.ldap.JdkLDAP
// com.sun.security.sasl.Provider
```

java.util.ServiceLoader (4)

- Egy modulban fejlesztett szolgáltatót a modul deklaráció egy `provides` direktívájában kell megadni.
  - A `provides` direktíva a szolgáltatást és a szolgáltatót is megadja.
- Egy JAR állományba csomagolt szolgáltatót egy szolgáltató konfigurációs állomány azonosít a `META-INF/services` könyvtárban, melynek neve a szolgáltatás teljesen minősített neve.
  - Az állomány a szolgáltatók teljesen minősített nevét tartalmazza, soronként egyet.

Olvashatóság (1)

- Ha az *A* modul függ a *B* modultól, akkor azt mondjuk, hogy *A* **olvassa** (***reads***) *B*-t, és hogy *B* **olvasható** (***readable***) *A* által.
  - Ilyenkor az *A* modul hozzáfér a *B* modul által exportált csomagok nyilvános felső szintű típusaihoz.
- Definíció szerint minden modul olvassa önmagát.
  - Az olvashatóság reflexív reláció.

Olvashatóság (2)

- Az olvashatóság nem tranzitív reláció!
  - Példa: az *A* modul nem olvassa a *C* modult!

```
module A {
   requires B;
}

module B {
   requires C;
}
```

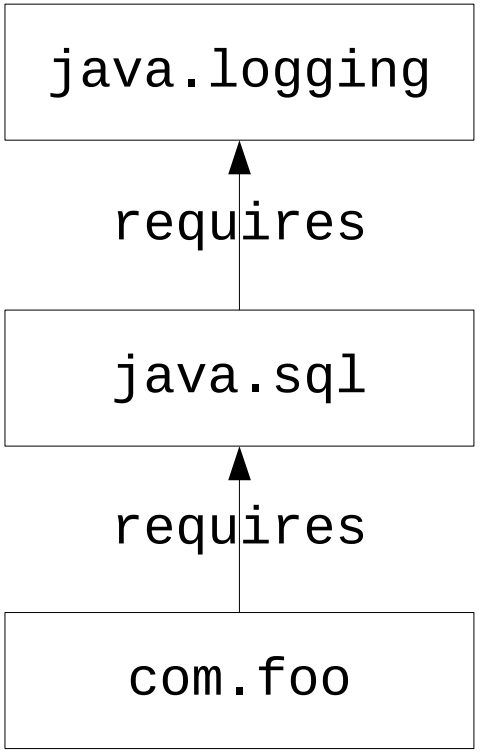
Olvashatóság (3)

- Az olvashatóság nem tranzitív reláció!
  - Példa:

```
module java.logging {
    exports java.util.logging;
}

module java.sql {
    requires java.logging;
    requires java.xml;
    exports java.sql;
    exports javax.sql;
    exports javax.transaction.xa;
}

module com.foo {
    requires java.sql;
    exports com.foo;
}
```



Olvashatóság (4)

- Az olvashatóság nem tranzitív reláció!
  - Példa (folytatás):
    - A `com.foo` modulban fordítási hibát eredményez az alábbi kódrész megjelölt metódushívása, mivel a modul nem olvassa a `java.logging` modult.

```
import java.sql.*;
import java.util.Properties;

String url;
Properties info;

Driver driver = DriverManager.getDriver(url);
Connection conn = driver.connect(url, info);
driver.getParentLogger().info("Connection acquired");
```

Olvashatóság (5)

- Az olvashatóság nem tranzitív reláció!
  - Példa (folytatás):
    - A probléma egy lehetséges megoldása a `com.foo` modult függővé tenni a `java.logging` modultól is:

```
module com.foo {
   requires java.logging;
   requires java.sql;
   exports com.foo;
}
```

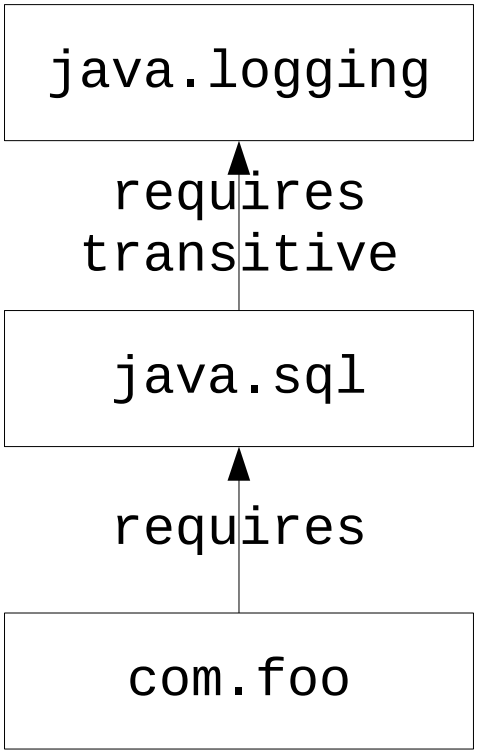
Olvashatóság (6)

- Az olvashatóság nem tranzitív reláció!
  - Példa (folytatás): a probléma elegáns megoldása:

```
module java.logging {
    exports java.util.logging;
}

module java.sql {
    requires transitive java.logging;
    requires transitive java.xml;
    exports java.sql;
    exports javax.sql;
    exports javax.transaction.xa;
}

module com.foo {
    requires java.sql;
    exports com.foo;
}
```



Olvashatóság (6)

- **Implicit olvashatóság** (***implied readability***):
  - A `requires transitive` direktíva azt fejezi ki, hogy aktuális modultól függő modulok által is olvasható a függőség.
    - A példában tehát a `com.foo` modul is olvassa `java.logging` modult.

Olvashatóság (7)

- Egyetlen modul sem olvashat két vagy több olyan modult, melyek ugyanazt a csomagot exportálják.
  - Ennek megsértése fordítási hibát eredményez.

Névtelen modulok

- Egy speciális, úgynevezett **névtelen modul**hoz (***unnamed module***) tartozik minden olyan típus, mely nem tartozik valamely nevesített modulhoz.
- A névtelen modul fogalma hasonló a névtelen csomag fogalmához.
- Azért szükséges, mert a Java SE 9 előtt nem volt lehetőség modulok deklarálására.
- A Java SE platform egy implementációja legalább egy névtelen modul használatát kell, hogy támogassa.
- Egy névtelen modul minden más modult olvas.
- Egy névtelen modul minden csomagját exportálja.

Modul útvonalak

- A `java`, `javac`, `jlink` és más parancssori eszközök számára a modulok megadása modul útvonalakban.
- A modul útvonalak olyan listák, melyek minden eleme az alábbiak valamelyike vagy az alábbiakat tartalmazó könyvtár:
  - A gyökér könyvtárában egy `module-info.class` állományt tartalmazó JAR állomány (úgynevezett moduláris JAR állomány).
  - Egy JMOD állomány.
  - Egy modult tartalmazó könyvtár.
- Egy modul útvonalban egy adott nevű modul csak egyszer megengedett!

JMOD állományok

- A JAR állományok lehetőségein túlmutató új formátum, mely támogatja például natív könyvtárak becsomagolását is.
- Kizárólag a fordítási és szerkesztési időben használhatók, végrehajtási időben nem!
  - Az utóbbihoz szükséges lenne például natív könyvtárak menet közbeni kitömörítésére és szerkesztésére.
- JMOD állományokba vannak csomagolva a JDK moduljai is.
  - Lásd a `$JAVA_HOME/jmods` könyvtárat.
- `jmod`: új parancssori eszköz JMOD állományok létrehozásához, manipulálásához és vizsgálatához.
  - Lásd: *Java Platform, Standard Edition Tools Reference* – `jmod` https://docs.oracle.com/javase/9/tools/jmod.htm

Szerkesztés

- A szerkesztés a fordítás és a futtatás közötti opcionális fázis, melynek során modulok egy halmazából és függőségeikből egy egyéni futásidejű lemezkép kerül létrehozása.
  - Lásd: *JEP 282: jlink: The Java Linker* http://openjdk.java.net/jeps/282
- `jlink`: a szerkesztésre szolgáló új parancssori eszköz.
  - Lásd: *Java Platform, Standard Edition Tools Reference* – `jlink` https://docs.oracle.com/javase/9/tools/jlink.htm

Moduláris JDK (1)

- Lásd: *JEP 200: The Modular JDK* http://openjdk.java.net/jeps/200
- A JSR 376-ban meghatározott és a JEP 261-ben implementált Java platform modulrendszer használata a JDK modularizációjához.
- A JDK olyan modulokra történő felosztása, melyek fordítási időben, összeállítási időben és futási időben különféle konfigurációkká kombinálhatók össze, ideértve többek között:
  - A teljes Java SE platformnak, a teljes JRE-nek és a teljes JDK-nak megfelelő konfigurációkat.
  - Olyan egyéni konfigurációkat, melyek csak bizonyos meghatározott modulokat tartalmaznak esetlegesen külső könyvtár és alkalmazás modulokkal kibővítve, valamint mindezen modulokhoz tranzitíven szükséges modulokat.

Moduláris JDK (2)

- Kétféle modul megkülönböztetése:
  - **Szabványos modulok**:
    - Olyan modulok, melyek specifikációit a JCP szabályozza.
    - A szabványos modulok neve a `java.` karakterlánccal kezdődik.
  - **Nem szabványos modulok**:
    - A JDK részét alkotó egyéb modulok.
    - A nem szabványos modulok neve a `jdk.` karakterlánccal kezdődik.

Moduláris JDK (3)

- A JDK moduláris szerkezete irányított gráffal jeleníthető meg.
  - A gráfban a csúcsok a modulokat ábrázolják, az élek pedig a modulok közötti függéseket.
- Lásd: https://bugs.openjdk.java.net/secure/attachment/72525/jdk.png

Moduláris JDK (4)

- A legalsó `java.base` modul olyan alapvető osztályokat tartalmaz, mint például a `java.lang.Object` és a `java.lang.String`.
  - Ez a modul nem függ egyetlen modultól sem és minden modul függ tőle.
- Az ábra tetejéhez közeli `java.se.ee` modul a Java SE platformot alkotó modulokat gyűjti egybe, beleértve azokat a modulokat is, melyek átfedőek a Java EE platformmal.
  - A modul az aggregátor modulok példája, melyek más modulok tartalmát gyűjtik össze, azokhoz nem adva hozzá saját tartalmat.
- A `java.se` aggregátor modul azokat a modulokat gyűjti össze, melyek a Java SE platformot alkotják és nem átfedőek a Java EE platformmal.

Moduláris JDK forráskód

- Lásd: *JEP 201: Modular Source Code* http://openjdk.java.net/jeps/201
- A JDK forráskódjának modulokba történő átszervezése (új könyvtárszerkezet kialakítása a JDK forráskódjához).

Moduláris futásidejű lemezképek (1)

- Lásd:
  - *JEP 220: Modular Run-Time Images* http://openjdk.java.net/jeps/220
  - Mark Reinhold. *Project Jigsaw: Modular run-time images*. https://mreinhold.org/blog/jigsaw-modular-images
- A JDK és a JRE futásidejű lemezképek (*run-time image*) szerkezetének átalakítása a modulok alkalmazásához, a teljesítmény a biztonság és a karbantarthatóság javításához.
- Egy olyan új URI-séma (`jrt`) meghatározása a futásidejű lemezképekben tárolt modulok, osztályok és erőforrások megnevezéséhez a lemezkép belső szerkezetének vagy formátumának felfedése nélkül.

Moduláris futásidejű lemezképek (2)

- A JDK 9 előtt kétféle futásidejű lemezkép megkülönböztetése: JRE és JDK.
  - A JDK lemezkép a JRE lemezkép egy másolatát tartalmazza a `jre/` alkönyvtárban.
- Az új lemezkép szerkezet megszünteti ezt a megkülönböztetést, a JRE és JDK lemezképe azonos szerkezetű.
  - A JDK lemezkép egyszerűen egy olyan futásidejű lemezkép, mely történetesen a teljes fejlesztői eszközkészletet tartalmazza.
- Egy moduláris lemezkép JAR állományok helyett inkább modulokból áll.

Moduláris futásidejű lemezképek (3)

- Az alábbi lehetőségek megszüntetése:
  - *Java Endorsed Standards Override Mechanism* https://docs.oracle.com/javase/8/docs/technotes/guides/standards/
    - Azt tette lehetővé, hogy a JCP-n kívül karbantartott szabványok újabb verzióinak vagy a Java SE platform részét képező, de attól függetlenül fejlődő különálló API-k implementációit telepítsük egy futásidejű lemezképbe.
  - *The Extension Mechanism for Support of Optional Packages* https://docs.oracle.com/javase/8/docs/technotes/guides/extensions/

Moduláris futásidejű lemezképek (4)

- Az `rt.jar` és `tools.jar` állományok eltávolítása:
  - A korábban a `lib/rt.jar`, `lib/tools.jar` és egyéb belső JAR állományokban tárolt `.class` és erőforrás állományok hatékonyabb módon kerülnek tárolásra implementáció specifikus állományokban a `lib/` könytárban.

Moduláris futásidejű lemezképek (5)

- Példa:

```
// JDK 8:
System.out.println(ClassLoader
    .getSystemResource("java/lang/Object.class"));
// jar:file:/usr/lib/jvm/java-8-oracle/jre/lib/rt.jar!
/java/lang/Object.class
```

```
// JDK 9:
System.out.println(ClassLoader
    .getSystemResource("java/lang/Object.class"));
// jrt:/java.base/java/lang/Class.class
```

Moduláris „Helló, világ!” program (1)

- Könyvtárszerkezet és forráskód:

```
src/
   org.hello/
      module-info.java
      org/
         hello/
            Main.java
```

```
// module-info.java:
module org.hello {
   exports org.hello;
}
```

Moduláris „Helló, világ!” program (2)

- Fordítás:

```
$ javac -d mods/org.hello \ célkönyvtár
  src/org.hello/module-info.java \
  src/org.hello/org/hello/Main.java
```

Moduláris „Helló, világ!” program (3)

- Futtatás:

```
$ java --module-path mods \ a modulokat tartalmazó könyvtár(ak)
  --module org.hello/org.hello.Main
```

Moduláris „Helló, világ!” program (4)

- JMOD készítés:

```
$ mkdir jmods
$ jmod create --class-path mods/org.hello \
  jmods/org.hello.jmod
```

Moduláris „Helló, világ!” program (5)

- Szerkesztés:

```
$ jlink --module-path $JAVA_HOME/jmods:jmods \
  --add-modules org.hello \
  --output dist \
  --strip-debug \
  --compress 2 \
  --no-header-files \
  --no-man-pages \
  --launcher hello=org.hello/org.hello.Main
```

Moduláris „Helló, világ!” program (6)

- A szerkesztés eredményeként kapott `dist` könyvtár szerkezete (a teljes méret mindössze kb. 30 MB):

```
bin/
    hello
    java
    keytool
conf/
legal/
lib/
```

Integrált fejlesztői környezetek

- **Eclipse IDE**: *Configure Eclipse for Java 9* https://wiki.eclipse.org/Configure_Eclipse_for_Java_9
- **IntelliJ IDEA**: *Getting Started with Java 9 Module System* https://www.jetbrains.com/help/idea/getting-started-with-java-9-module-system.html
- **NetBeans IDE**: Java 9 támogatás csak az innen letölthető kiadásban: http://bits.netbeans.org/download/trunk/nightly/latest/

Fejlesztőeszközök

- *Apache Maven JMod Plugin*
  https://maven.apache.org/plugins/maven-jmod-plugin/
- *Apache Maven JLink Plugin*
  https://maven.apache.org/plugins/maven-jlink-plugin/

További ajánlott irodalom

- Cay S. Horstmann. *Core Java SE 9 for the Impatient*. 2nd Edition. Addison-Wesley Professional, 2017. http://www.informit.com/store/core-java-se-9-for-the-impatient-9780134694726
- Kishori Sharan. *Java 9 Revealed: For Early Adoption and Migration*. Apress, 2017. https://www.apress.com/br/book/9781484225912
- Koushik Kothagal. *Modular Programming in Java 9*. Packt Publishing, 2017. https://www.packtpub.com/application-development/modular-programming-java-9
- Sander Mak, Paul Bakker. *Java 9 Modularity: Patterns and Practices for Developing Maintainable Applications*. O'Reilly Media, 2017. http://shop.oreilly.com/product/0636920049494.do