HITACHI
Inspire the Next

開発・運用時のガイド

JDK11 への移行に伴う留意点

2019.10
October

はじめに

本書は、開発・運用フェーズで使用するドキュメントとして、Java™ Development Kit 8 から Java™ Development Kit 11 への移行に伴う留意点について記述しています。

１．対象とする読者

本書は、Java™ Development Kit 11 を使用し、システムを設計・構築・運用する立場にある方を対象としています。

■商標類

・HITACHI は、（株）日立製作所の商標または登録商標です。

・Oracle と Java は、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。

・その他記載の会社名、製品名は、それぞれの会社の商標もしくは登録商標です。

■英略語の表記

本書では、英略語を次のように表記しています。

表記	製品名
Java	Java™
Java EE	Java™ Platform, Enterprise Edition
Java SE	Java™ Platform, Standard Edition
JDK	Java™ Development Kit
	JDK™

■発行元

株式会社日立製作所 サービス＆プラットフォームビジネスユニット サービスプラットフォーム事業本部

目　次

1 JDK11 への移行に伴う留意事項

JDK11 への移行に伴う留意事項を説明します。

本章の構成

1.1 JDK11 への移行に伴う留意事項

1.1 JDK11 への移行に伴う留意事項

本節では，JDK8 から JDK11 に移行した際の留意事項を記載します。

(1) Module System について

「Module System」の導入に伴い，一部の内部 API がカプセル化され，利用できなくなりました(このような内部 API は，JDK 8 において非サポートとされていたものです)。そのため，そのような内部 API を使用したソースコードを JDK 9 以降でコンパイル，もしくは Java アプリケーションを実行した場合，エラーや警告が発生する場合があります。

詳細は以下の Web ページを参照してください。
http://www.oracle.com/technetwork/java/javase/9-relnote-issues-3704069.html#JDK-8142968

内部 API を使用しているかどうかは, 事前に jdeps コマンドを利用することで確認できます。以下に jdeps コマンドの使用例を示します。

```
>jdeps Sample1.class
Sample1.class -> java.base
   <unnamed>       ->      java.lang                       java.base
   <unnamed>       ->      jdk.internal.misc               JDK internal API (java.base)
```

上記は内部 API を使用している Sample1 クラスを対象に jdeps コマンドを実行したものです。この結果から，”JDK internal API”という文字列が出力されており，Sample1.class の内部では jdk.internal.misc パッケージ内の内部 API にアクセスしていることがわかります。
ただし，jdeps コマンドではリフレクションを利用した内部 API 呼び出しを確認することができないのでご注意ください。

(2) クラスファイルのバージョンについて

Java SE 11 からクラスファイルのバージョンが 55 になりました。
これにより，クラスファイルフォーマット仕様に対し，Module System に関連した各 Attribute の追加や NestHost/NestMembers の Attribute，ConstantDynamic バイトコードが追加されています。

クラスファイル変換等でクラスファイルを出力する場合，出力するクラスファイルのバージョンとクラスファイルのフォーマットが一致している必要があります。一致していない場合はクラスロード時に java.lang.VerifyError が発生するため，クラスファイルの読み込みや書き換え等を行っている，もしくは，そのようなツールを使用している場合は新仕様対応を行ってください。
なお，クラスファイルのバージョンはバイナリエディタや javap コマンドを使用してクラスファイル

内の”major version”から確認してください。

また，javac コマンドのクロスコンパイルオプション(source/ target もしくは release)を用いることで指定したバージョンのクラスファイルを作成することもできます。
ただし，その場合はクラスバージョンに対応する Java SE の機能までしか使用することはできません。
クロスコンパイルオプション(source/target)の指定可能なバージョンは”6”もしくは”1.6”以上です。
release オプションは”6”以上です。ただし，”6”もしくは”1.6”の指定は非推奨であり，指定すると警告メッセージが出力されます。

(3) ロケール依存サービスについて

ロケール依存サービスのデフォルトが，JDK9 以降では CLDR(Common Locale Data Repository)となります。これにより、数値や貨幣、時刻、日付が JDK8 以前の表記と変わることがあります。以下に JDK8 以前と JDK9 以降で表記が変わる Java アプリケーションの一例を示します。

```
import java.util.*;
import java.text.*;
public class LocaleTest {
    public static void main(String args[]) {
        DateFormat df = DateFormat.getDateInstance(DateFormat.LONG, Locale.JAPAN);
        System.out.println(df.format(new Date()));
    }
}
```

上記、LocaleTest を実行した場合、下記のように出力されます。
JDK8 以前：2017/12/14
JDK9 以降：2017 年 12 月 14 日

なお、以下のように java.locale.providers プロパティに COMPAT を指定することで、JDK8 と同じロケールデータを使用し、JDK8 と同じ表記にできます。

```
>java -Djava.locale.providers=COMPAT,CLDR LocaleTest
```

(4) システムクラスローダについて

JDK9 以降では，システムクラスローダが java.net.URLClassLoader(以降，URLClassLoader と呼びます。)を継承しなくなりました。そのため，Java アプリケーションにおいて，システムクラスローダが URLClassLoader を継承していることを前提としている場合には，エラーが発生する場合があります。

エラーが発生するケースの例としては，以下(a)(b)のケース等が有ります。

(a) システムクラスローダインスタンスを URLClassLoader でキャストしている
(b) システムクラスローダインスタンスに対し，URLClassLoader クラスのメソッドを呼び出している

JDK9 以降では，システムクラスローダが URLClassLoader を継承していることを前提としてるかどうかに注意してください。上記(a)(b)のケースに該当している等の場合，キャスト型やクラス・データ構造の変更が必要となります。

(5) String 関連の変更

JDK9 では、String 関連でいくつかの変更が有り JDK8 と JDK11 に移行した際には，メモリ使用量や実行性能に影響がでる場合があります。

JDK9 での変更点として,「JEP254 Compact Strings」により java.lang.String の内部データが char 配列から byte 配列に変更されました。これにより内部データとして割り当たるメモリ量が最大で半分となります。また,「JEP 280 Indify String Concatenation」により，文字列結合処理が大幅に改良されています。

これらは JavaVM の内部処理の変更なので, 基本的には Java アプリケーションの変更等は必要ありません。しかし，文字列を頻繁に使用する Java アプリケーションでは，メモリ使用量や実行性能が大幅に変化する可能性があります。そのため，JDK8 から JDK11 に移行する際には，トータルテストや性能測定等で問題ないかどうかを確認することを推奨致します。

(6) -Xbootclasspath オプションと-Xbootclasspath/p オプションの削除

JDK 9 以降では-Xbootclasspath オプションと-Xbootclasspath/p オプションは削除されました。-Xbootclasspath/a オプションは削除されていません。

(7) 推奨標準機構の削除

JDK 9 以降では推奨標準機構が削除され, java.endorsed.dirs システムプロパティと jdk/lib/endorsed ディレクトリは使用できません。JavaVM の起動時，このシステムプロパティ指定もしくはディレクトリが存在する場合は，以下のようなメッセージが出力され JavaVM の起動に失敗するのでご注意ください。

[jdk/lib/endorsed ディレクトリが存在する場合]

```
<JAVA_HOME>/lib/endorsed is not supported. Endorsed standards and standalone APIs
in modular form will be supported via the concept of upgradeable modules.
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.
```

推奨標準機構で指定していたクラスファイルは，クラスパス(-cp)もしくはブートクラスパス(-XbootClasspath/a)，--patch-module オプション等を利用してください。

(8) 拡張機能機構の削除

JDK 9 以降では拡張機能機構が削除され， java.ext.dirs システムプロパティと jdk/lib/ext ディレクトリは使用できません。JavaVM の起動時，このシステムプロパティ指定もしくはディレクトリが存在する場合は，以下のようなメッセージが出力され JavaVM の起動に失敗するのでご注意ください

[jdk/lib/ext ディレクトリが存在する場合]

```
<JAVA_HOME>/lib/ext exists, extensions mechanism no longer supported; Use -classpath instead.
.Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.
```

拡張機能機構で指定していたクラスファイルは，クラスパス(-cp)もしくはブートクラスパス(-XbootClasspath/a)，--patch-module オプション等を利用してください。

(9) javah

javah コマンドは JDK 10 で削除されました。javah コマンドの機能は, javac コマンドの-h オプションに移管されています。以下に javac コマンドの-h オプションの使用方法を記載します。

```
>javac -h [ヘッダファイル格納ディレクトリ] [コンパイル対象ファイル]
```

(10) java.lang.Class#getAnnotation メソッド呼び出しで発生する例外の変更

アノテーションに指定したクラスがクラスパスに存在しない場合に，当該アノテーションを java.lang.Class クラスの getAnnotation メソッドで取得しようとすると，JDK10 以前は java.lang.ArrayStoreException(以降，ArrayStoreException)をスローしていました。JDK11 以降から，同様の処理を実施しても ArrayStoreException をスローせず，getAnnotation メソッドで取得したアノテーションの値を読み込もうとしたタイミングで，java.lang.TypeNotPresentException(以降，TypeNotPresentException)をスローするようになりました。もし，ArrayStoreException を前提とした処理がある場合には，非互換性が発生する可能性があります。その場合は，TypeNotPresentException を明示的にキャッチするようにしてください。

本変更の詳細は以下のページでご確認ください。
https://www.oracle.com/technetwork/java/javase/11-relnote-issues-5012449.html#JDK-7183985

(11) CORBA と JavaEE 関連のモジュールについて

以下の表に示す，Java EE 向けモジュールと CORBA モジュールは JDK 11 で削除されました。

#	モジュール名
1	java.xml.ws
2	java.xml.bind
3	java.activation
4	java.xml.ws.annotation
5	java.corba
6	java.transaction
7	java.se.ee
8	jdk.xml.ws
9	jdk.xml.bind

モジュールの削除に伴い，上記モジュールに関連したコマンド群やシステムプロパティも削除されています。詳細は，Web ページを参照してください。
https://www.oracle.com/technetwork/java/javase/11-relnote-issues-5012449.html#JDK-8190378

(12) その他

その他の移行に関する留意事項については，以下の Web ページを参照してください。
https://docs.oracle.com/javase/jp/11/migrate/index.html#JSMIG-GUID-561005C1-12BB-455C-AD41-00455CAD23A6

－以上－