JDK 開発者ガイド (Solaris 編)

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–0394–10
2005 年 1 月

原典: *JDK for Solaris Developer's Guide*

Part No: 817-7970–10

Revision A

050112@10536

目次

はじめに

このマニュアルでは、Solaris™ オペレーティングシステム用の Java™ 2 Platform Standard Edition 5 における新機能と強化された機能を紹介し、その概要を説明します。

対象読者

このマニュアルは、Solaris オペレーティングシステムで Java 2 Platform Standard Edition 5 を使用するアプリケーション開発者を対象としています。Java ソフトウェアは、企業環境におけるサーバーおよびクライアント側の Java 技術アプリケーションが優れた性能を発揮できるように最適化されています。

このマニュアルは J2SE™ 5 ドキュメントの一部です。J2SE 5 ドキュメントは http://java.sun.com/j2se/1.5.0/ja/docs/ja/index.html で入手できます。最終的なリリース段階では、このオンライン文書が Java 2 Platform Standard Edition 5 製品の正式な説明になります。

内容の紹介

第 1 章では、製品の特長と強化された機能を示します。

第 2 章では、互換性の問題について説明します。

関連項目

このリリースに関する情報は次のドキュメントにも記載されています。

- 『*JDK 5.0* リリースノート』http://java.sun.com/j2se/1.5.0/ja/relnotes.html にあるオンライン文書
- 『*JDK 5.0* ドキュメント』http://java.sun.com/j2se/1.5.0/ja/docs/ja/index.html にあるオンライン文書
- 『*Java 2 Platform Standard Edition 5.0 API* 仕様』http://java.sun.com/j2se/1.5.0/ja/docs/ja/api/index.html にあるオンライン文書

Sun のオンラインマニュアル

`docs.sun.com` では、Sun が提供しているオンラインマニュアルを参照することができます。マニュアルのタイトルや特定の主題などをキーワードとして、検索を行うこともできます。URL は、`http://docs.sun.com` です。

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 **P–1** 表記上の規則

字体または記号	意味	例
`AaBbCc123`	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	`.login` ファイルを編集します。 `ls -a` を使用してすべてのファイルを表示します。 `system%`
`AaBbCc123`	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	`system%` **`su`** `password:`

表 **P–1** 表記上の規則　　*(続き)*

字体または記号	意味	例
AaBbCc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、`rm` *filename* と入力します。
『 』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。
「 」	参照する章、節、ボタンやメニュー名、強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	`sun% grep '^#define \` `  XV_VERSION_STRING'`

コード例は次のように表示されます。

- C シェル

  ```
  machine_name% command y|n [filename]
  ```

- C シェルのスーパーユーザー

  ```
  machine_name# command y|n [filename]
  ```

- Bourne シェルおよび Korn シェル

  ```
  $ command y|n [filename]
  ```

- Bourne シェルおよび Korn シェルのスーパーユーザー

  ```
  # command y|n [filename]
  ```

[ ] は省略可能な項目を示します。上記の例は、*filename* は省略してもよいことを示しています。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: `Shift` キーを押します)。ただし、キーボードによっては `Enter` キーが `Return` キーの動作をします。

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば、`Ctrl-D` は `Control` キーを押したまま `D` キーを押すことを意味します。

第 1 章

新しい機能と強化された機能

Java Platform Standard Edition 5 バージョン 1.5.0 は、機能が大幅に向上されたメジャーリリースです。以下に示す機能は、前回のメジャーリリース (1.4.0) 以降に 1.5.0 で導入されたものです。

新機能の概要は、http://java.sun.com/developer/technicalArticles/releases/j2se15/ にある「J2SE 1.5 in a Nutshell」にも挙げられています。問題点については、http://java.sun.com/j2se/1.5.0/ja/relnotes.html にある 「JDK 5.0 リリースノート」を参照してください。

パフォーマンスの強化

パフォーマンスの強化についての概要は、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/performance/speed.html にある「JDK 5.0 での拡張」を参照してください。

Java 言語の機能

詳細については、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/language/index.html にある「JDK 5.0 での拡張」を参照してください。

汎用型

待ち望まれた型システムの拡張により、コンパイル時のタイプセーフを提供しながら、さまざまなタイプのオブジェクトに対して型またはメソッドを使用できるようになりました。汎用型はコレクションフレームワークにコンパイル時のタイプセーフを追加し、手間のかかるキャスティング作業を省きます。JSR 14 と、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/language/generics.html にある汎用型関連の文書を参照してください。

ループの拡張

この新しい言語構造により、コレクションと配列の繰り返し処理を行う際にイテレータとインデックス変数の手間やエラー発生を防ぐことができます。JSR 201 と、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/language/foreach.html にあるドキュメントを参照してください。

オートボクシング／アンボクシング

この機能は、プリミティブ型 (`int` など) とラッパー型 (`integer` など) との間で手動変換をする手間を省きます。JSR 201 と、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/language/autoboxing.html にあるドキュメントを参照してください。

型保証された列挙

この柔軟なオブジェクト指向の列挙型機能を使用することによって、任意のメソッドとフィールドを使用して列挙型を作成できます。この機能には、冗長性がなく、エラーを起こしにくい型保証された列挙パターン (*Effective Java*、Item 21) のあらゆるメリットがあります。JSR 201 と、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/language/enums.html にあるドキュメントを参照してください。

変数引数

この機能によって、可変長引数リストを受け付けるメソッドを呼び出す場合に引数リストを配列に手動で入れる必要がなくなります。JSR 201 と、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/language/varargs.html にあるドキュメントを参照してください。

Static のインポート

この機能を使用すると、*Constant Interface Antipattern* (不変のインターフェースアンチパターン) の短所なしで、クラス名で静的メンバーを修飾することを避けられます。JSR 201 と、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/language/static-import.html にあるドキュメントを参照してください。

注釈

この言語機能によって、ツールを使用してソースコード内の注釈からボイラープレートコードを生成可能になり、さまざまな状況でボイラープレートコードを記述する手間を省けます。これは、プログラマが何を行うべきかを記述しツールがそれを実行するコードを生成する、宣言的なプログラミングスタイルが可能になります。また、ソースファイルの変更に伴って最新状態に保持しなければならない付属ファイルを管理する必要もなくなります。それに代わって、情報はソースファイル内で管理できます。JSR 175 と、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/language/annotations.html にあるドキュメントを参照してください。

仮想マシン

クラスデータの共有

クラスデータの共有機能は、アプリケーションの起動時間の短縮とフットプリントの減少を目的としたものです。インストール処理では、システム `jar` ファイルからプライベートな内部表現に一連のクラスが読み込まれ、続いてその表現が共有アーカイブファイルにダンプされます。その後行われる JVM の起動では、その共有アーカイブがメモリー内にマップされます。このため、クラスを読み込む手間が省かれ、これらのクラス用の JVM のメタデータのほとんどを複数の JVM プロセスで共有できます。詳細については、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/vm/class-data-sharing.html にあるドキュメントを参照してください。

ガーベージコレクタのエルゴノミクス

この並行コレクタは、アプリケーションが必要とするメモリーを監視し、これに適合するように拡張されました。 アプリケーションのパフォーマンス目標を指定すると、JVM は、それらの目標に適合する最小のアプリケーションフットプリントで Java ヒープのサイズを調整します。適応性のあるこのポリシーは、最上のパフォーマンスを達成するために必要なコマンド行オプションの調整を不要にすることを目的にしています。ガベージコレクション機能の概要については、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/vm/gc-ergonomics.html にあるドキュメントを参照してください。

サーバークラスマシンの検出

アプリケーションの起動時に、起動用ウィンドウはアプリケーションがサーバークラスマシンで稼動しているかどうかを検出できます。
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/vm/server-class.html にあるドキュメントを参照してください。

スレッド優先順位の変更

スレッドの優先順位の割り当ては、優先順位が明示的に設定されていない Java スレッドとネイティブスレッドが、ある程度対等に競うことができるように変更されました。http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/vm/thread-priorities.html にあるドキュメントを参照してください。

致命的なエラーの処理

致命的なエラーを報告するメカニズムはより優れた診断を出力するように改良され、信頼性も高まりました。

高精度タイミングのサポート

相対的な時間測定のためにナノ秒精度のタイムソースにアクセスできるように、メソッド `System.nanoTime()` が追加されました。`System.nanoTime()` によって返されるタイム値の実際の精度は、プラットフォームによって異なります。

コアライブラリ

`lang` パッケージと `util` パッケージ

java.lang および java.util の拡張についての概要は、http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/lang/enhancements.html にあるドキュメントを参照してください。

ネットワークの機能拡張

追加されたネットワーク機能の概要については、http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/net/enhancements-1.5.0.html にあるドキュメントを参照してください。

セキュリティ

この J2SE リリースでは、セキュリティ機能が大幅に強化されています。このリリースでは、セキュリティトークンのサポート向上、セキュリティ標準のサポート増加 (SASL、OCSP、TSP)、スケーラビリティ (SSLEngine) とパフォーマンスの向上などが実現されているほか、暗号や Java GSS などの領域でも数多くの機能強化がなされています。詳細については、http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/security/index.html にあるドキュメントを参照してください。

国際化

以下の機能強化がなされました。

- 文字処理は、現在 Unicode 標準のバージョン 4.0 をベースとしています。これは、`java.lang` パッケージ内の `Character` クラスと `String` クラス、`java.text` パッケージ内の照合と双方向テキスト分析機能、`java.util.regex` パッケージ内の `character` クラスを始め、J2SE のさまざまな部分に影響を与えます。このアップグレードの一環として、JSR 204 エクスパートグループによって補助文字のサポートが指定され、J2SE 全体に実装されました。詳細については、『JAVA プラットフォームにおける補助文字のサポート』、『Java Specification Request 204』、`Character` クラスドキュメントなどを参照してください。
- `DecimalFormat` クラスは、精度を失うことなく `BigDecimal` 値と `BigInteger` 値のフォーマット設定および解析が行えるように拡張されました。これらの値のフォーマットは自動的に拡張されます。`BigDecimal` に構文解析するには、`setParseBigDecimal` メソッドを使用して有効にする必要があります。
- このリリースでは、`java.util` パッケージおよび `java.text` パッケージ内のすべてのロケール対応機能でベトナム語がサポートされるようになりました。サポートされるロケールと書記法の詳細については、「サポートされているロケール」を参照してください。

http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/intl/index.html にあるドキュメントも参照してください。

環境変数のサポート向上

`System.getenv(String)` メソッドは非推奨扱いではなくなりました。新しい `System.getenv()` メソッドを使用すると、`Map<String,String>` としてプロセス環境にアクセスできます。
http://java.sun.com/j2se/1.5.0/ja/docs/ja/api/java/lang/System.html#getenv(java.lang.String) にあるドキュメントを参照してください。

ProcessBuilder

サブプロセスの呼び出しには、`Runtime.exec` よりも新しい `ProcessBuilder` クラスが便利です。特に、`ProcessBuilder` を使用すると、変更されたプロセス環境 (親のプロセス環境に基づくが、一部変更されたもの) で サブプロセスの開始が容易になります。
http://java.sun.com/j2se/1.5.0/ja/docs/ja/api/java/lang/ProcessBuilder.html にあるドキュメントも参照してください。

Formatter

`printf` 形式のフォーマット文字列のインタプリタである `Formatter` クラスは、レイアウトの調整と整列、数値 / 文字列 / 日時データの共通フォーマット、ロケール固有の出力などのサポートを提供します。`byte`、`java.math.BigDecimal`、`java.util.Calendar` などの共通の Java 型がサポートされます。`java.util.Formattable` インタフェースを使用することで、任意のユーザー型のフォーマットを限定的にカスタマイズできます。

http://java.sun.com/j2se/1.5.0/ja/docs/ja/api/java/util/Formatter.html にあるドキュメントを参照してください。

Scanner

`java.util.Scanner` クラスを使用して、テキストをプリミティブまたは `String` に変換できます。このクラスは `java.util.regex` パッケージをベースとしているため、このクラスを使用してストリーム、ファイルデータ、文字列、または `Readable` インタフェースの実装に対して正規表現に基づく検索を行うこともできます。http://java.sun.com/j2se/1.5.0/ja/docs/ja/api/java/util/Scanner.html にあるドキュメントを参照してください。

リフレクション

汎用型、メタデータ、列挙、および簡易メソッドのサポートが追加されました。また、`java.lang.Class` が総称化されました。
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/reflection/enhancements.html にあるドキュメントを参照してください。

JavaBeans コンポーネントのアーキテクチャ

変更された bean 部分を特定するためにインデックスを使用するバウンドプロパティをサポートするため、`PropertyChangeEvent` のサブクラスとして `IndexedPropertyChangeEvent` が追加されました。また、インデックス化されたプロパティ変更イベントの起動をサポートするために、`PropertyChangeSupport` クラスにメソッドがいくつか追加されました。
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/beans/index.html にあるドキュメントを参照してください。

コレクションフレームワーク

コレクションフレームワークは、以下のように強化されました。

- コレクションを対象とする汎用型、拡張ループ、およびオートボクシングの 3 つの新しい言語機能
- 3 つの新しいインタフェース、`Queue`、`BlockingQueue`、および `ConcurrentMap` がフレームワークに 追加されました (このうちの 2 つは `java.util.concurrent` の一部)。
- `Queue` の 2 つの実装と 1 つのスケルトン実装が 追加されました。
- 5 つのブロックキュー実装と 1 つの `ConcurrentMap` 実装が追加されました。

- 型保証された列挙で使用できるように、特殊用途の `Map` 実装と `Set` 実装が提供されました。
- 特殊用途の書き込み時コピー `List` 実装と `Set` 実装が追加されました。
- ほとんどのコレクションインタフェースに動的なタイプセーフを加えるラッパー実装が提供されました。
- コレクション操作に使用できるいくつかの新しいアルゴリズムが提供されました。
- 配列のハッシュコードと文字表現を計算するメソッドが提供されました。

http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/collections/index.html にあるドキュメントを参照してください。

XML 処理用の Java API (JAXP)

詳細については、JSR 206、または
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/xml/jaxp/index.html にあるドキュメントを参照してください。

ビット操作処理

ラッパークラス (`Integer`、`Long`、`Short`、`Byte`、および `Char`) は、`highestOneBit`、`lowestOneBit`、`numberOfLeadingZeros`、`numberOfTrailingZeros`、`bitCount`、`rotateLeft`、`rotateRight`、`reverse`、`signum`、`reverseBytes` などの共通のビット操作処理をサポートするようになりました。

数値演算

ライブラリが提供する数値機能は、いくつかの次の方法で拡張されました。

- `BigDecimal` クラスは、固定精度の浮動小数点演算のサポートを追加しました。JSR 13 を参照してください。
- `Math` ライブラリと `StrictMath` ライブラリには、双曲線超越関数 (sinh、cosh、tanh)、cube root、base 10 logarithm などが含まれます。
- 16 進浮動小数点サポート - 特定の浮動小数点値を正確でかつ予測可能なものにするため、浮動小数点リテラルと、`Float` および `Double` における浮動小数点変換メソッドを指定する文字列に16 進表記を使用できます。

http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/math/index.html に挙げられたドキュメントを参照してください。

インストゥルメンテーション

新しい `java.lang.instrument` パッケージには、Java プログラミングエージェントが Java 仮想マシン上で稼動しているプログラムを計測できるサービスが含まれます。このインストゥルメンテーションメカニズムは、メソッドのバイトコードを修正するものです。

直列化

バージョン 1.5.0 で追加された列挙型を処理できるようにサポートが追加されました。列挙インスタンスを直列化する規則は、直列化が可能な通常のオブジェクトを直列化する規則とは異なります。直列化された列挙インスタンスフォームには、その列挙定数名と、そのベースの列挙型を特定する情報しか含まれません。非直列化の動作も異なります。適切な列挙クラスの特定には、クラス情報が使用されます。返す列挙定数の取得には、このクラスおよび受け取った定数名によって Enum.valueOf メソッドが呼び出されます。

http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/serialization/index.html にあるドキュメントを参照してください。

ユーティリティの並行性サポート

`java.util.concurrent`、`java.util.concurrent.atomic`、および `java.util.concurrent.locks` パッケージは、同時実行されるクラスおよびアプリケーション (スレッドプール、スレッドに対して安全なコレクション、セマフォ、タスクスケジューリングフレームワーク、タスク同期ユーティリティ、自動変数、ロックなど) を開発するための、パフォーマンスとスケーラビリティに優れた、スレッドに対して安全な構築ブロックを実現する、拡張性の高い強力なフレームワークを提供します。これらのパッケージをコアクラスライブラリに追加することで、手動でこれらのユーティリティを作成する手間が省けます。これは、データ構造に対するコレクションフレームワークの効果によく似ています。また、これらのパッケージは、プロセッサによって提供される並行性のサポートを利用する高度な並行プログラミングを対象とした低レベルのプリミティブも提供します。これによりプログラマは、以前にはネイティブコードを使用せずには不可能であった、高パフォーマンスで高スケーラビリティの並行アルゴリズムを Java 言語に実装できます。

JSR 166 と、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/concurrency/index.html にあるドキュメントを参照してください。

スレッド

`java.lang.Thread` クラスは、以下のように強化されました。

- 「Thread Priority」 処理は変更されました。詳細については上記のリンクを参照してください。
- スレッドの実行状態を照会できるように、`Thread.State` 列挙クラスと新しい `getState()` API が提供されました。
- 新しいスレッドダンプ API (`Thread` クラス内の `getStackTrace` メソッドと `getAllStackTraces` メソッド) を使用すると、プログラム化した方法で 1 つのスレッドまたはすべてのスレッドのスタックトレースを取得できます。
- `uncaughtExceptionHandler` メカニズムは以前には ThreadGroup クラスを通してしか使用できませんでしたが、Thread クラスを通して直接使用できるようになりました。
- `sleep()` メソッドに、1 ミリ秒よりも短いスリープタイムを設定できる新しいフォームが追加されました。

監視と管理

今回の J2SE リリースでは、Java プラットフォームの監視と管理の機能が大幅に強化されました。

- Java 仮想マシン用の監視と管理 API。新しい `java.lang.management` パッケージは、Java 仮想マシンの監視と管理のためのインタフェースを提供します。
- ロギング機能用の監視と管理 API。新しい `java.util.logging.LoggingMXBean` インタフェースは、ロギング機能用の管理インタフェースです。
- Java 仮想マシンの JMX インストゥルメンテーション。 Java 仮想マシン (JVM) は、JMX を使用してその監視と管理が行える組み込み型のインストゥルメンテーションです。リモートまたはローカルの JVM インストゥルメンテーションを監視・管理する JMX エージェント、または JMX インストゥルメンテーションを使用した任意のアプリケーションの JMX エージェントを簡単に起動できます。詳細については、http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/management/agent.html にある「JMX を使用する監視と管理」を参照してください。
- SNMP エージェントは、JSR 163 に定められている方法で、JVM インストゥルメンテーションの標準の MIB を公開します。詳細については、「SNMP の監視と管理」を参照してください。
- J2SE 5 リリースには、Java™ Management Extensions JMX™ API バージョン 1.2 と、JMX Remote API バージョン 1.0 の RMI コネクタが含まれています。JMX API を使用すると、監視・管理用のライブラリとアプリケーションを計測できます。RMI コネクタは、このインストゥルメンテーションをリモートでアクセスするためのものです。詳細については、http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/management/index.html にある JMX の文書を参照してください。

統合ライブラリ

Remote Method Invocation (RMI)

RMI は、以下のように強化されました。

- スタブクラスの動的な生成 - このリリースでは、実行時にスタブクラスを動的に生成できるようになりました。このため、Java Remote Method Invocation (Java RMI) スタブコンパイラ、rmic を使用してリモートオブジェクト用のスタブクラスをあらかじめ生成する必要がなくなりました。ただし、以前のバージョンで稼動しているクライアントをサポートする必要があるリモートオブジェクト用のスタブクラスを生成するために、rmic を使用する必要があります。
- 標準の SSL/TLS ソケットファクトリクラス - このリリースでは、標準の Java RMI ソケットファクトリクラス、`javax.rmi.ssl.SslRMIClientSocketFactory` と `javax.rmi.ssl.SslRMIServer SocketFactory` が追加されました。これらのクラスは、Java Secure Socket Extension (JSSE) を使用して Secure Sockets Layer (SSL) または Transport Layer Security (TLS) プロトコルを介して通信します。
- `inetd/xinetd` からの rmid または Java RMI サーバーの起動 - `System.inheritedChannel` メソッドが提供するこの新しい機能を使用すると、アプリケーションは仮想マシン (VM) を起動したプロセスから継承されたチャネル (例: `java.nio.channels.SocketChannel`、`java.nio.channels.ServerSocketChannel` など) を取得できます。継承されたこのようなチャネルは、単一の着信接続にサービスを提供するように使用することも (`SocketChannel` と同様)、あるいは複数の着信接続を受け入れるように使用することも (`ServerSocketChannel` と同様) できます。このため、`inetd` (Solaris) または `xinetd` (Linux) によって起動された Java ネットワーキングアプリケーションは、`inetd/xinetd` から継承された `SocketChannel` または `ServerSocketChannel` を取得できます。

http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/rmi/index.html にあるドキュメントを参照してください。

Java Database Connectivity (JDBC)

`javax.sql` パッケージの一部として J2SE バージョン 1.4 で導入された `RowSet` インタフェースは、コンポーネント間でデータを受け渡す軽量手法を提供します。

このリリースでは、開発者に便利なように、`RowSet` オブジェクトを使用できる 5 つの一般的な方法で (JSR 114 として) `RowSet` インタフェースが実装されました。これらの実装により、現状のままあるいは拡張して自由に使用できる標準が提供されます。以下に、これらの 5 つの標準実装を示します。

- `JdbcRowSet` - JDBC テクノロジを使用するために実装される結果セットまたはドライバをカプセル化するために使用されます。
- `CachedRowSet` - データソースからデータを取得する場合や変更されたデータをデータソースに書き戻す場合を除き、そのデータソースからは切り離されて独自に稼動します。このため、メモリー内にできるだけ多くのデータを保存する軽量コンテナとして機能できます。
- `FilteredRowSet` - `CachedRowSet` を拡張したもので、データのサブセットを取得するのに使用されます。
- `JoinRowSet` - `CachedRowSet` を拡張したもので、複数の `RowSet` オブジェクトからデータの SQL `JOIN` を取得するために使用されます。
- `WebRowSet` - `CachedRowSet` を拡張したもので、XML データに使用されます。これは、標準化された XML スキーマを使用する XML 内で表形式コンポーネントを表現します。

http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/jdbc/index.html にあるドキュメントを参照してください。

CORBA、Java IDL、および RMI-IIOP

CORBA、Java IDL、および Java RMI-IIOP の機能強化については、「J2SE 1.4.x と 5.0 の間での CORBA 機能の変更点」で説明しています。
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/idl/index.html にある Java IDL ドキュメントと、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/rmi-iiop/index.html にある Java RMI-IIOP ドキュメントを参照してください。

Java Naming and Directory Interface™ (JNDI)

JNDI の新機能は次のとおりです。

- ディレクトリ / ネーミングサービスから完全名にアクセスする javax.naming.NameClassPair の機能強化
- 標準の LDAP 制御である Manage Referral Control (参照コントロール)、Paged Results Control (ページング結果コントロール)、および Sort Control (ソートコントロール) のサポート
- LDAP 名の操作のサポート

http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/jndi/index.html にあるドキュメントを参照してください。

ユーザーインタフェース

国際化

- 論理フォントを使用して多言語使用のテキストを描画するために、2D は、サポートされるすべての書記法に対してインストール済みのホスト OS フォントを利用するようになりました。たとえば、タイ語ロケール環境で実行しているが韓国語フォントがインストールされている場合、タイ語と韓国語の両方が描画されます。J2RE はその lib/fonts/fallback ディレクトリにインストールされている物理フォントを自動的に検出し、それらの物理フォントを 2D 描画用のすべての論理フォントに加えるようになりました。
- AWT は、Windows 2000/XP 上の Unicode API を使用するようになりました。この結果、一部のコンポーネントは Windows ロケール設定の制限を受けることなくテキストを処理できます。たとえば、AWT テキストコンポーネントは、Windows ロケール設定に関係なくデーバナーガリー (Devanagari) 文字入力システムでのテキストを受け入れ、表示できます。

http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/intl/index.html にあるドキュメントを参照してください。

Java Sound テクノロジ

- このリリースでは、すべてのプラットフォームでポートが使用できるようになりました (RFE 4782900)。
- すべてのプラットフォームで MIDI デバイス入出力が使用できるようになりました (RFE 4812168 と 4782924)。
- 最適化されたダイレクトオーディオアクセスが実装されました (RFE 4908240 と 4908879)。ネイティブミキシング (ハードウェアミキシング機能が付いた Linux ALSA、Solaris Mixer、Windows DirectSound) を提供するシステムでは、これがデフォルトで有効になります。
- 新しいリアルタイムのシーケンサはすべての MIDI デバイスに利用でき、トランスミッタは無制限となりました (RFE 4773012)。
- ユーザーは、`sound.properties` 構成ファイルを使用してデフォルトのデバイスを選択できます (RFE 4776511)。詳細については、`MidiSystem` と `AudioSystem` を参照してください。
- Midi デバイスは、接続されたレシーバとトランスミッタを照会できます (RFE 4931387、`MidiDevice.getReceiver` メソッドと `MidiDevice.getTransmitter` メソッド)。
- `AudioFormat`、`AudioFileFormat`、および `MidiFileFormat` には、詳細にフォーマットを記述し修飾するプロパティが加えられました (RFE 4925767 と 4666845)。

- 簡単に使用できるメソッドによって、`AudioSystem` からラインの取得を容易に行えるようになりました (RFE 4896221)。
- シーケンサインタフェースは、MIDI シーケンスの特定の部分を継続的にループできるように、`loop` メソッドによって拡張されました (RFE 4204105)。
- Java Sound は、VM の終了を妨げることがなくなりました (バグ 4735740)。

http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/sound/index.html にあるドキュメントを参照してください。

Java 2D™ テクノロジ

2D 機能には、広範な Solaris または Linux プリンタサポート、ファイルとストリームからフォントを作成する新しい方法、`VolatileImage` とイメージ描画のハードウェアベースの高速化に関連する新しい方法などが追加されました。テキスト描画コードに多数の内部変更が加えられたことで、その堅牢性、パフォーマンス、およびスケーラビリティが大幅に向上しました。他のパフォーマンス機能としては、OpenGL (デフォルトでは無効) を使用するハードウェアにより高速化された描画などがあります。

http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/2d/index.html にあるドキュメントを参照してください。

Image I/O

Image I/O システムに、BMP 形式と WBMP 形式用の読み込みと書き込みが追加されました。

http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/imageio/index.html にあるドキュメントを参照してください。

AWT

1.5.0 リリースでは、顧客から要求の多かったものも含め、多数の AWT 強化とバグ修正を実現しました。顕著な例として、新しい `MouseInfo` クラスはデスクトップ上のマウスの位置を検出できるようになりました。新しい `Window` メソッドは、新しく作成されるウィンドウ (またはフレーム) のデフォルトの位置をプラットフォームに合わせて指定できます。`Window` の他の強化によって、ウィンドウ (またはフレーム) を常に最上位に配置することも可能となりました (この機能は Solaris または Linux 上の一部のウィンドウマネージャでは機能しません)。データ転送の領域では、新しい `DropTargetDragEvent` API を使用することで、ドラッグ操作の際にドロップターゲットが転送データにアクセスできるようになりました。

詳細は、http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/awt/index.html にあるAWTを参照してください。

Swing

1.4.2 リリースでは、Swing に 2 つの新しい見た目と使い心地、XP と GTK が追加されました。それに続けて、1.5.0 でもさらに 2 つの見た目と使い心地、Synth (スキンを変更可能) と Ocean (Metal の新しいテーマ) が提供されます。見た目と使い心地だけでなく、`JTable` に印刷サポートも追加しました。これにより、`JTable` のきれいな印刷コピーを簡単に出力できるようになりました。7 年をかけて、`jFrame.add` を `jFrame.getContentPane().add()` と同等にしました。

http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/swing/index.html にある Swing のドキュメントを参照してください。

配備

一般的な配備

Pack200 (JSR 200 で定義されている、JAR ファイルの新しい超高圧縮形式) を使用すると、Java Web Start アプリケーションおよび Java Plug-in アプレットで使用される JAR ファイルのダウンロードサイズを大幅に縮小できます。

一般的な配備の機能と機能強化についての概要は、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/deployment/enhancements-1.5.0.html
を参照してください。

Java Web Start の配備

Java Web Start の配備機能と機能強化についての概要は、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/javaws/enhancements-1.5.0.html
を参照してください。

ツールとツールアーキテクチャ

Java Virtual Machine Tool Interface (JVMTI)

JVMTI は、開発ツールと監視ツールで使用する新しいネイティブプログラミングインタフェースです。JVMTI は、Java 仮想マシン (VM) で稼動しているアプリケーションの状態の検査と、その実行制御の両方に使用できます。JVMTI は、VM の状態にアク

セスする必要があるさまざまなツール (プロファイリングツール、デバッグツール、監視ツール、スレッド分析ツール、カバレージ解析ツールなど) を対象に VM インタフェースを提供することを目的としています。

JVMTI は、 J2SE の次のメジャーリリースで、現在推奨されていない JVMPI と JVMDI に置き換わります。

http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/jvmti/index.html にあるドキュメントを参照してください。

Java Platform Debugger Architecture (JPDA)

以下 URL のドキュメントの「JPDA の拡張機能」で説明されているように、JPDA 自体、多数の新機能が追加されています。

- 読み取り専用の JDI サブセットが定義されました。このサブセットは、デバッグコードを実行できない debuggee に使用できます (コアファイルや、ハングしているプロセス、デバッグモードで開始されていないプロセスなど)。このサブセットを使用して、このような debuggee のデバッグに使用できる JDI コネクタを作成できます。
- コネクタとトランスポート用のサービスプロバイダインタフェースを使用すると、デバッガベンダーや一般ユーザーは、独自の JDI コネクタとトランスポートを作成し、それらを JPDA 参照実装に組み込むことができます。たとえば、デバッガと debuggee 間の通信を行うための SSL を使用するためにコネクタを提供できます。
- JDI は、新しい言語機能 (汎用型、列挙、および変数引数) をサポートします。
- JPDA の最下層である Java Virtual Machine Debugger Interface (JVMDI) は推奨されていませんでしたが、J2SE の次のメジャーリリースで削除される予定です。これに置き換わるものは、Java Virtual Machine Tool Interface (JVMTI) です。これは、プロファイリング、デバッグとも可能にする、より一般的なインタフェースです。現在のプロファイリングインタフェースである Java Virtual Machine Profiling Interface(JVMPI) も推奨されておらず、次のメジャーリリースで削除される予定です。
- JPDA 参照実装には、コアファイルやハングしたプロセスのデバッグを可能にする新しい JDI コネクタが含まれます。

http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/jpda/index.html にあるドキュメントを参照してください。

Java プログラミング言語コンパイラ (javac)

コンパイラには、次のオプションがあります。

- `-source 1.5` - ソースファイル内で 1.5 固有の言語機能を使用できます (`-target 1.5` は暗黙的に設定)。

- `-target 1.5` - ライブラリと仮想マシン内で `javac` が 1.5 固有の機能を使用できます。
- `-Xlint` - 正当であるが疑わしく、問題を引き起こしがちであるというプログラム構造について、`javac` が警告メッセージを生成できます。この例として、`Serializable` を実装するが `serialVersionUID` を定義しないクラスの宣言があります。
- `-d32` - 32 ビットの Solaris または Linux プラットフォームを示します。
- `-d64` - 64 ビットの Solaris または Linux プラットフォームを示します。

http://java.sun.com/j2se/1.5.0/ja/docs/ja/tooldocs/solaris/javac.html にあるマニュアルページドキュメントを参照してください。

Javadoc ツール

http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/javadoc/whatsnew-1.5.0.html にある 「Javadoc 5.0 の新機能」 を参照してください。

注釈処理ツール (apt)

`apt` は、注釈処理用の新しいコマンド行ユーティリティです。これには、プログラムの注釈を処理するためのリフレクト API セットとサポート用インフラも含まれます。

http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/apt/index.html にあるドキュメントを参照してください。

OS とハードウェアプラットフォーム

サポートされているシステム構成

詳細については、http://java.sun.com/j2se/1.5.0/system-configurations.html を参照してください。

64 ビット AMD Opteron プロセッサ

J2SE 5 では、Suse Linux または Windows 2003 を使用したサーバー VM で AMD Opteron プロセッサがサポートされます。

第 2 章

以前のリリースとの互換性

このマニュアルでは、次のトピックについて説明します。

互換性についてのドキュメントは、前後するバージョン間だけの非互換性を示すように分割されています。たとえば、この 1.5.0 互換性ページは、1.4.2 と 1.5.0 の間の非互換性についてのみ説明しており、以前のバージョンとの非互換性については説明していません。このため、1.5.0 がすべてのバージョンとどのような非互換性があるかを確認するには、すべての互換性ページに目を通す必要があります。

前後するリリース間の非互換性については、以下のドキュメントで説明されています。

- http://java.sun.com/j2se/1.4.2/ja/compatibility.html にある「バージョン 1.4.2 における非互換性 (1.4.1 以降)」
- http://java.sun.com/j2se/1.4.1/ja/compatibility.html にある「バージョン 1.4.1 における非互換性 (1.4.0 以降)」
- http://java.sun.com/j2se/1.4/ja/compatibility.html にある「バージョン 1.4.0 における非互換性 (1.3 以降)」
- http://java.sun.com/j2se/1.3/ja/compatibility.html にある「バージョン 1.3 における非互換性 (1.2 以降)」

Java Language Specification, Second Edition (http://java.sun.com/docs/books/jls/index.html) の公開以降に Java プログラミング言語の仕様に加えられた変更の概要については、『Java Language Specification Maintenance Page』 (http://java.sun.com/docs/books/jls/jls-maintenance.html) を参照してください。

バイナリ互換性

Java 2 Platform Standard Edition 5 のバージョン 1.5.0 は、以下に示す非互換性を除き、バージョン 1.4.2 とバイナリレベルの上位互換性があります。これは、下記の非互換性を除き、バージョン 1.4.2 コンパイラで構築されたクラスファイルはバージョン 1.5.0 上で正しく稼動することを意味します。

初期のバイトコード難読化ツールの中には、仮想マシンの仕様とは異なる形式のクラスファイルを生成するものがありました。このような不適切な形式のクラスファイルは Java 2 JDK の仮想マシン上では動作しませんが、初期バージョンの仮想マシン上では動作するものもあります。この問題を修復するには、適切な形式のクラスファイルを生成する新しい難読化ツールでクラスファイルを生成し直します。

ソース互換性

ソースの下位互換性はサポートされません。新しい言語機能や Java 2 Platform API を使用するソースファイルは、初期バージョンの Java 2 Platform では使用できません。

後の方に挙げられた非互換性を除き、一般にポリシーは次のとおりです。

- 保守リリース (1.4.1、1.4.2 など) は新しい言語機能や API を取り入れていないため、互いにソースレベルの互換性があります。
- 機能リリースとメジャーリリース (1.3.0、1.4.0、1.5.0 など) は上位互換性はありますが、ソースレベルの下位互換性はありません。

推奨されない API は、下位互換性だけを考慮してサポートされているインタフェースです。これらのどれかを使用すると、`-nowarn` コマンド行オプションを指定していないかぎり、`javac` コンパイラは警告メッセージを生成します。推奨されない API を使用しないようにプログラムを変更することを推奨しますが、JVMDI と JVMPI がシステムから完全に削除される以外、現在のところこのような API を削除する計画はありません (バグ 4639363 を参照)。

`sun.*` パッケージ内のいくつかの API は変更されました。これらの API は開発者向けではありません。`sun.*` パッケージからインポートする場合、開発者は自分の責任で行うようにしてください。詳細については、
http://java.sun.com/products/jdk/faq/faq-sun-packages.html にある 「*Why Developers Should Not Write Programs That Call sun.* Packages*」を参照してください。

Java 2 Platform Standard Edition 5 (1.4.2 以降) における非互換性

J2SE 5 は以前のバージョンの Java 2 Platform との強力な互換性を持っています。既存のプログラムは変更しなくても、ほとんどすべて、J2SE 5 上で動作するはずです。しかし、JRE と JDK にはまれな状況や「コーナーケース」で発生するソースレベルまたはバイナリレベルのマイナーな潜在的非互換性も存在するため、その状況に対処できるように補足情報をここで説明します。

1. 総称化 - 総称化は既存のクラスとメソッドに総称 (generic) 型のパラメータと引数を追加するプロセスであり、それらのクラスの仕様に一致した方法で行われます。JSR 14 では、多くの core ライブラリについて総称化の仕様が定められました (代表的なものはコレクションクラスと `Class` クラス)。1.5 Beta 2 リリースでは、core の総称化の効果が、可能なかぎりプラットフォームの残りの部分全体に伝播されました。

   総称化されたクラス、コンストラクタ、メソッド、フィールドを使用するほとんどのソースコードは 1.5 でもコンパイルを継続しますが、一部はコンパイルを継続しません。総称化の変更が原因となってコードがコンパイルに失敗する場合は、最も簡単な方法として `javac` コマンド行で `-source 1.4` と指定できます。

   汎用型および core の総称化については、JSR 14 と、http://java.sun.com/j2se/1.5/pdf/generics-tutorial.pdf にある「Generics in the Java Programming Language」を参照してください。

2. 仮想マシン - 以前には、Solaris (SPARC 版) のデフォルト仮想マシン (VM) はクライアント VM でした。しかし、パフォーマンスを考えた場合、サーバー VM の方が適しているため、多くの Solaris (SPARC 版) はサーバーとして使用されます。このため、1.5 ではサーバークラスの Solaris (SPARC 版) マシンはデフォルトでサーバー VM を実行します。一般に、サーバー VM のスループットはクライアント VM よりもはるかに優れていますが、起動時間は少し劣ります。サーバークラスマシン は、現在、2 個以上のプロセッサと 2GB 以上のメモリーを持つマシンと定義されています。

   詳細については、「サーバークラスマシンの検出」(http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/vm/server-class.html) と「ガーベージコレクタのエルゴノミクス」(http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/vm/gc-ergonomics.html) を参照してください。

3. 仮想マシン - 1.5 で導入されたクラス共有機能を反映するため、このリリースでは `java.vm.info` プロパティ (`java -version` によって表示されるテキストに示される) は共有モードを指定するようになりました。`java.vm.info` プロパティ値の最後まで解析を行うコード、または `java -version` の出力を解析するコードは必要に応じて変更してください。

   詳細については、バグ 4964160 と、http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/vm/class-data-sharing.html にある「クラスデータの共有」を参照してください。

4. クラスローダ - 以前には、`String` クラス名引数を受け付ける `ClassLoader` メソッドにバイナリでないクラス名を指定できました。この意図的でない動作は、長年にわたるクラス名の仕様に準拠していませんでした。1.5 では、これらの `ClassLoader` メソッドをチェックするパラメータは仕様に準拠するように変更され、バイナリ名でないクラス名は認識されないその他のクラス名と同様に扱われます。クラス名を明示的に要求するか、あるいはクラス名を明示的に返す API (`Class.forName`、`Class.getName` など) は、参照型にバイナリ名を使用します。このため、`Class` を返すクラス名を通常のユーザーが生成することはありません。

   詳細については、『Java Language Specification, Second Edition』(http://java.sun.com/docs/books/jls/) 内の 「binary name」の定義 (http://java.sun.com/docs/books/jls/second_edition/html/binaryComp.doc.html#59876) を参照してください。また、バグ 4986512 の評価も参照してください。

5. 直列化 - コンパイラによって生成された全体的な合成結果に変更を加えると、デフォルトのシリアルバージョン UID に影響を与えます。したがって、その UID が明示的に上書きされない場合には直列化の互換性が保たれなくなる可能性があります。

   詳細については、バグ 4786115 を参照してください。

6. ロギング - 以前には、`java.util.logging.Level(String name, int value, String resourceBundleName)` コンストラクタは null の `name` 引数を許可しましたが、`parse` メソッドは許可しませんでした。1.5 では、コンストラクタは名前が null の場合、`NullPointerException` をスローするようになりました。このコンストラクタを使用するには `Level` をサブクラス化する必要があり、後続の呼び出し (`toString` のようなシンプルな呼び出しを除く) で null の `Level` 名を使用する場合には `NullPointerException` を受け取るため、互換性上のリスクは軽減されます。

   詳細については、バグ 4625722 を参照してください。

7. Apache - `org.apache` クラス (J2SE API ではサポートされなかったが `javax.xml` パッケージでは使用されている) は、開発者によってダウンロードされる最近のクラスバージョンと重複しないように、1.5 で `com.sun.org.apache.package.internal` に移されました。J2SE リリースに含まれる `org.apache` クラスに依存するアプリケーションを 1.5 で稼動させるには、以下の作業のいずれかを実施する必要があります。

   - 「JAXP」に含まれるサポート対象のインタフェースだけを使用するようにアプリケーションを作成する
   - Apache から `org.apache.xalan` クラスをダウンロードする

   詳細については、バグ 4740355 を参照してください。

8. JAXP - J2SE 1.4 プラットフォームには JAXP 1.1 (Crimson) が含まれていました。J2SE 1.5 プラットフォームには JAXP 1.3 (Xerces) が含まれます。Crimson と Xerces は、同一のコードベースを持つ個別のバージョンではありません。これらは、まったく異なる JAXP 標準実装です。このため、どちらも JAXP 標準に準拠していますが、微妙な違いがあります。

Crimson はサイズが小さく高速でしたが、結局のところ機能的には Xerces (Apache でホスティングされるオープンソース実装) に及びませんでした。この上、JAXP 標準が 1.1 から 1.3 へ進みました。これらの 2 つの要因が相まって互換性問題が出現しました。

詳細については、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/xml/jaxp/JAXP-Compatibility_150.html にある「JAXP 互換性ガイド J2SE5 Platform 用」を参照してください。

9. JAXP - J2SE 1.4 プラットフォームは、DOM Level 2 API をサポートしました。J2SE 1.5 プラットフォームは、DOM Level 3 ファミリの API をサポートします。DOM Level 3 インタフェースに新しいメソッドがいくつか追加されました。このため、DOM Level 2 を使用する既存のアプリケーションの中には新しいインタフェースではコンパイルできないものがあります。

   クラスパス内で DOM Level 2 の代わりに DOM Level 3 が使用されていると、ほとんどの DOM Level 2 アプリケーションは稼動します。しかし、中には `NoSuchMethodException` が発生するものもあります。したがって、一部のアプリケーションはバイナリレベルの互換性を維持できません。

   詳細については、
   http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/xml/jaxp/JAXP-Compatibility_150.html にある「JAXP 互換性ガイド J2SE5 Platform 用」を参照してください。

10. JAXP - J2SE 1.4 プラットフォームは、SAX 2.0 API をサポートしました。J2SE 1.5 プラットフォームは、SAX 2.0.2 をサポートします。SAX 2.0.2 はバグ修正リリースであり、API の変更はありません。しかし、SAX 2.0.2 リリースの一部として行われたいくつかの修正作業が互換性問題を引き起こしている可能性があります。

    - `ErrorHandler`、`EntityResolver`、`ContentHandler`、および `DTDHandler` は、現在ではアプリケーションによって null に設定される可能性があります。この場合、SAX 2.0 は `java.lang.NullPointerException` をスローするために XML プロセッサを必要とします。ほとんどの構文解析部はデフォルト設定に戻すことにより null に対応するため、この変更が XML プロセッサに関係します。
    - `DefaultHandler` は、各種のハンドラ (`EntityResolver` など) のデフォルトの実装クラスです。`DefaultHandler` における `resolveEntity` メソッド実装は、現在では `throws IOException, SAXException` として宣言されます。以前には、この実装は `SAXException` しかスローできませんでした。
    - `resolveEntity` メソッドによってスローされる例外リストへの `java.io.IOException` の追加は、ソースレベルでは互換性のない変更です。具体的には、`resolveEntity` を呼び出すコードは SAX 2.0 で正常にコンパイルできる可能性がありますが、SAX 2.0.2 ではコンパイルに失敗する可能性があります。これは、`SAXException` と共に `IOException` を処理する必要があるためです。

    詳細については、
    http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/xml/jaxp/JAXP-Compatibility_150.html にある「JAXP 互換性ガイド J2SE5 Platform 用」を参照してください。

11. JAXP - 以前には、デフォルトのトランスフォーマは Xalan でした。Apache コミュニティは XSLT 2.0 の開発用として XSLTC をデフォルトのプロセッサとすることに同意したため、1.5 では XSLTC がデフォルトのトランスフォーマです。これには以下の互換性のリスクがあります。

    - Xalan には XSLTC にはないバグがあり、XSLTC には Xalan にないバグがあります。Xalan のバグを考慮したアプリケーションコードは失敗する可能性があります。
    - Xalan がサポートしている拡張子の中には XSLTC でサポートされないものがあります。これらの拡張子については、JAXP 仕様および XSLT 仕様の範囲外です。このことに影響を受けるユーザーは、現在でも Apache から Xalan クラスをダウンロードできます。また、将来的には XSLTC でより多くの拡張子がサポートされると予測されます。

    詳細については、
    http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/xml/jaxp/JAXP-Compatibility_150.html
    にある「JAXP 互換性ガイド J2SE5 Platform 用」を参照してください。

12. 2D - 以前には、null の `Image` パラメータを `Graphics.drawImage` メソッドに渡すと `NullPointerException` が発生しました。1.5 ではこの例外は発生しません。このように動作が変わったため、Microsoft VM で稼動したアプリケーションは標準の VM でも稼動できます。`NullPointerException` に依存しているアプリケーションは、1.5 で稼動するように変更する必要があります。

13. AWT - 以前には、フォーカストラバーサルポリシーを提供できたのはフォーカスサイクルルートであるコンテナだけでした。1.5 では、どのコンテナもフォーカストラバーサルポリシーを提供できます。`Container` の新しい `FocusTraversalPolicyProvider` プロパティは、ポリシーを提供するかどうかを示します。

    Java プラットフォームで提供されるフォーカストラバーサルポリシーは、フォーカストラバーサルポリシープロバイダに対応するように変更されました。具体的には、フォワード (バックワード) トラバーサル中にポリシーがフォーカストラバーサルポリシープロバイダに遭遇すると、ポリシーは提供されたフォーカスサイクルルートに属するものとしてそのコンポーネントを扱わず、フォーカストラバーサルポリシプロバイダのフォーカストラバーサルポリシーを使用して次の (前の) コンポーネントを取得する必要があります。返されたコンポーネントがフォーカストラバーサルポリシプロバイダのフォーカストラバーサルポリシーによって返された最初の (最後の) コンポーネントと同じである場合、フォーカストラバーサルポリシープロバイダの後 (前) のサイクル内における次の (前の) コンポーネントをポリシーの呼び出しによって取得する必要があります。フォーカスサイクルルート内の最初と最後のコンポーネントの計算には、必要に応じて (最初または最後のコンポーネント自体が `Container` でかつフォーカストラバーサルポリシープロバイダである場合) フォーカストラバーサルポリシープロバイダのフォーカストラバーサルポリシーを使用する必要があります。

    この変更はフォーカストラバーサルポリシー内に新しいメソッドを使用することを要求しないため、サン以外のフォーカストラバーサルポリシーは引き続き稼動します (ただしプロバイダという概念はサポートしない)。

    フォーカストラバーサルポリシーを作成してあり、プロバイダをサポートしたいという場合は、1.5 でプラットフォームによって提供されているポリシーに加えられている変更に類似した変更を加える必要があります。

詳細については、フォーカス仕様の
http://java.sun.com/j2se/1.5.0/ja/docs/ja/api/java/awt/doc-files/FocusSpec.html#FocusTraversa
にある「Focus Traversal Policy Providers」セクションと、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/api/java/awt/doc-files/FocusSpec.html
にある「AWT Focus Subsystem」を参照してください。

14. ドラッグ＆ドロップ - 以前には、X11 でサポートされたドラッグ＆ドロップ (DnD) プロトコルは Motif DnD プロトコルだけでした。1.5 では XDND プロトコルもサポートされ、Motif DnD プロトコルは Motif ライブラリに依存しないように実装し直されました。新しい Motif DnD プロトコル実装と Motif ライブラリで提供されるプロトコル実装の違いのためにリグレッションが起きる可能性があります。ただし、Motif ライブラリの実装はバグが発生するため、新しい実装は少なくとも質において高く、サポートも比較的良いと考えられています。

    詳細については、バグ 4638443 を参照してください。

15. Swing - カスタマイズされた背景色を持つボタンは、1.5 Java の見た目と使い心地がテーマである Ocean で意図されたように描画するためにはコード変更が必要となる場合があります。これは、デフォルトの設定では Ocean がボタンに明暗を付けるためです。明暗を付けたくない場合は、`contentAreaFilled` プロパティを true に設定するか、あるいは背景を `UIResource` でない `Color` に設定してください。ほとんどの場合、これは以下のように簡単です。`button.setBackground(Color.RED);` 何らかの理由で `UIResource` を選択している場合には、次のように指定して `UIResource` 以外の新しい `Color` を作成できます。

    ```
    button.setBackground(new Color(oldColor));
    ```

    詳細については、バグ 4908404 を参照してください。

16. Swing - `JTree` と `JList` では、ユーザーは常にキーボードを使用してリードインデックスを操作する必要がありました。たとえば、リードが `JList` 内で 4 行目 に存在する場合、上向きの矢印キーを押して 3 行目 にリードを移動させ、この行で項目を選択します。以上の操作で、これらのコンポーネントはリードをフォーカスされたインデックスと見なします。これらのコンポーネントは、その描画機能 (renderer) に情報を渡して特定のインデックスのフォーカスインジケータを描画するかどうかを示します。これは、そのインデックスがリードであるかどうかに基づいて行われます。

    1.5 よりも前のリリースでは、`JTable` は逆の処理を行っていました。そして、`JTree` と `JList` がリードを使用するのと同じ方法でアンカーインデックスを使用していました。これを正す要求が RFE 番号 4759422 としてなされ、最終的に 4303294 の一環として修正されました。現在 `JTable` は、`JList` と `JTree` に対応したものとなっています。以前の動作を想定していた開発者は、この影響を受ける可能性があります。たとえば、フォーカスされたセルとして `JTable` で何が表示されているかという情報を必要とするアプリケーションの場合、アンカーが想定されます。1.5 以前ではこれは正しかったかもしれませんが、現在では、実際には他のインデックスがフォーカスを示す四角形を表示している時点で 1 つのインデックスがフォーカスされていると判断される可能性があります。

    詳細については、バグ 4759422 と 4303294 を参照してください。

17. JVMDI - 1.5 では、Java Virtual Machine Debug Interface (JVMDI) は推奨されていません。*JVMDI* は、次のメジャーリリースよりサポートされなくなります。新しい開発では JVMTI を使用してください。既存のツールは JVMTI への移行を始めることをお勧めします。

詳細については、
http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/jvmti/index.html にあるJVMTIの文書を参照してください。

18. JVMPI - 1.5 では、Java Virtual Machine Profiling Interface (JVMPI) は推奨されていません。JVMPI は次のメジャーリリースよりサポートされなくなります。新しい開発では JVMTI を使用してください。既存のツールは JVMTI への移行を始めることをお勧めします。

    詳細については、
    http://java.sun.com/j2se/1.5.0/ja/docs/ja/guide/jvmti/index.html にあるJVMTIの文書を参照してください。