(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6409514号
(P6409514)
(45)発行日　平成30年10月24日(2018.10.24)　(24)登録日　平成30年10月5日(2018.10.5)

(51)Int.Cl.　　F I
G06F 9/445 (2018.01)　　G06F 9/445 120
G06F 8/60 (2018.01)　　G06F 8/60

請求項の数 10　(全 28 頁)

(21)出願番号　特願2014-228181(P2014-228181)
(22)出願日　平成26年11月10日(2014.11.10)
(65)公開番号　特開2016-91460(P2016-91460A)
(43)公開日　平成28年5月23日(2016.5.23)
審査請求日　平成29年10月16日(2017.10.16)

(73)特許権者　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(74)代理人　100109313
弁理士　机　昌彦
(74)代理人　100124154
弁理士　下坂　直樹
(72)発明者　梶ヶ谷　圭祐
東京都港区芝五丁目７番１号
日本電気株式会社内

審査官　三坂　敏夫

最終頁に続く

(54)【発明の名称】情報処理装置およびライブラリロード方法、並びにコンピュータ・プログラム

(57)【特許請求の範囲】
【請求項１】
ライブラリファイルごとに、前記ライブラリファイルの内容を識別する識別子情報を生成し、前記ライブラリファイルに含まれるロード要求の対象となりうる部分を表す要求対象情報と前記識別子情報との対応関係を表すロード要求対応情報を生成し、前記識別子情報と前記ロード要求対応情報を記憶装置に保存出力する識別子生成手段と、
実行されているアプリケーションから前記ロード要求を受け付け、前記ロード要求、および前記ロード要求対応情報に基づいて、前記ロード要求の対象を含む前記ライブラリファイルの識別子情報を求め、前記ロード要求の対象を表す要求対象情報と、前記求めた識別子情報を出力するロード要求解釈手段と、
前記ライブラリファイルに含まれる各部のロード状態を表すロード状態情報を管理し、前記ロード状態情報に基づいて、前記ロード要求の対象を表す要求対象情報および前記求めた識別子情報が示すロードを要求された部分が前記記憶装置にロードされていないと判断した場合に、前記要求対象情報、および前記求めた識別子情報が示す前記ライブラリファイルから、少なくとも前記ロードを要求された部分をロードし、前記ロード要求に対する応答を返すロード手段と
を備える情報処理装置。
【請求項２】
前記ロード手段は、前記ロード状態情報に基づいて、前記ロードを要求された部分がロードされていると判断した場合に、前記ロード要求に対して前記ロードを要求された部分

を何もロードせずに、前記応答を返す
請求項１記載の情報処理装置。
【請求項３】
前記アプリケーションを実行可能な状態にする配備を行う際、前記識別子情報と、前記識別子情報に対応する前記ライブラリファイルを参照するアプリケーションの数を表す参照数情報とを含むライブラリ識別子管理情報を生成するアプリケーション配備手段を、
さらに備える請求項１または２記載の情報処理装置。
【請求項４】
前記アプリケーション配備手段は、前記アプリケーションの前記実行可能な状態を解除する配備解除の際、前記ライブラリ識別子管理情報に基づいて、配備解除の結果、参照している前記アプリケーションが無くなる前記ライブラリファイルと、配備を解除される前記アプリケーションとに関する記憶情報を含むリソースを削除する
請求項３に記載の情報処理装置。
【請求項５】
前記識別子生成手段は、前記ライブラリファイルに含まれるパッケージ名を前記要求対象情報として前記ロード要求対応情報を生成し、
前記ロード要求解釈手段は、前記ロード要求の対象であるクラスのパッケージ名と、前記ロード要求対応情報に基づいて、前記ロード要求の対象を含む前記ライブラリファイルの識別子情報を求める
請求項１乃至４のいずれか１つに記載の情報処理装置。
【請求項６】
前記識別子生成手段は、前記ライブラリファイルのハッシュ情報を求め、前記ハッシュ情報に基づいて前記識別子情報を生成する
請求項１乃至５のいずれか１つに記載の情報処理装置。
【請求項７】
ライブラリファイルごとに、前記ライブラリファイルの内容を識別する識別子情報を生成し、
前記ライブラリファイルに含まれるロード要求の対象となりうる部分を表す要求対象情報と前記識別子情報との対応関係を表すロード要求対応情報を生成し、
前記ロード要求対応情報を記憶装置に保存し、
実行されているアプリケーションからロード要求を受け付けた際、
前記ロード要求、および前記ロード要求対応情報に基づいて、前記ロード要求の対象を含む前記ライブラリファイルの識別子情報を求め、
前記ライブラリファイルに含まれる各部のロード状態を表すロード状態情報に基づいて、前記ロード要求の対象を表す要求対象情報、および前記求めた識別子情報が示すロードを要求された部分が前記記憶装置にロードされていないと判断した場合に、前記求めた識別子情報が示す前記ライブラリファイルから、少なくとも前記ロードを要求された部分をロードし、さらに、前記ロード状態情報にロードしたことを登録し、
前記ロード要求に対する応答を返す　ライブラリロード方法。
【請求項８】
前記ロード要求を受け付けた際、前記ロードを要求された部分がロードされていると判断した場合に、
前記ロード要求に対して前記ロードを要求された部分をロードせずに、前記応答を返す請求項７記載のライブラリロード方法。
【請求項９】
前記識別子情報を生成するより後、かつ、前記ロード要求を受け付ける前に、
前記識別子情報と、前記識別子情報に対応する前記ライブラリファイルを参照する参照するアプリケーションの数を表す参照数情報とを含むライブラリ識別子管理情報を生成し、
前記アプリケーションの実行可能な状態を解除する配備解除の際、

前記ライブラリ識別子管理情報に基づいて、前記配備解除の結果、参照している前記アプリケーションが無くなる前記ライブラリファイルと、前記配備解除をされる前記アプリケーションとに関する記憶情報を含むリソースを削除する

請求項７または８記載のライブラリロード方法。

【請求項１０】

ライブラリファイルごとに、前記ライブラリファイルの内容を識別する識別子情報を生成し、前記ライブラリファイルに含まれるロード要求の対象となりうる部分を表す要求対象情報と前記識別子情報との対応関係を表すロード要求対応情報を生成し、前記ロード要求対応情報を記憶装置に保存する識別子生成処理と、

実行されているアプリケーションからロード要求を受け付け、前記ロード要求、および前記ロード要求対応情報に基づいて、前記ロード要求の対象を含む前記ライブラリファイルの識別子情報を求めるロード要求解釈処理と、

前記ライブラリファイルに含まれる各部のロード状態を表すロード状態情報を管理し、前記ロード状態情報に基づいて、前記ロード要求の対応を表す要求対象情報および前記求めた識別子情報が示すロードを要求された部分が前記記憶装置にロードされていないと判断した場合に、前記求めた識別子情報が示す前記ライブラリファイルから、少なくとも前記ロードを要求された部分をロードし、前記ロード要求に対する応答を返すロード処理とをコンピュータに実行させるコンピュータ・プログラム。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、アプリケーションプログラムを実行可能な情報処理装置（コンピュータ）におけるライブラリのロード技術に関する。

【背景技術】

【０００２】

アプリケーションプログラムを実行可能な情報処理装置において、機能を実行する際に、その機能を実現するプログラムを含むライブラリをロードする方法が存在する。例えば、Ｊａｖａ（登録商標）においては、クラスローダと呼ばれるモジュールが、アプリケーションによるクラスのロード要求に応じて、ライブラリからプログラムをロードすることが一般的である。

【０００３】

また、ライブラリには、機能強化または提供元の違いなどによって、同名であっても内容の異なる様々なバージョンが存在することが知られている。例えば、アプリケーションの実行環境を提供するアプリケーションサーバが利用するライブラリと、アプリケーションが利用する同名のライブラリのバージョンが異なる場合がある。このような場合、クラスローダが、あるクラス名に対するロード要求に対して、同じクラス名のプログラムを含む異なるバージョンのライブラリをロードする可能性がある。もし、クラスローダが、アプリケーションのロード要求に対して、アプリケーションサーバのライブラリをロードした場合、アプリケーションが正常動作しないという問題が発生する。すなわち、クラスローダは、アプリケーションに応じて、複数の同名のライブラリから適正なライブラリを選んでロードしなければならない。

【０００４】

このような問題に対する関連技術として、特許文献１には、１つのＪＶＭ（Ｊａｖａ Ｖｉｒｔｕａｌ Ｍａｃｈｉｎｅ）において、複数のアプリケーションプログラムが、同名のクラスを含む異なるライブラリを利用できる情報処理装置が開示されている。この特許文献１に記載された情報処理装置においては、利用者等が、ライブラリおよびプログラムに関する属性を、あらかじめテーブル情報として設定する。そして、設定手段が、テーブル情報に基づいてプログラムの配給元を判別し、判別した配給元に応じて、ライブラリファイルのパスを設定する。このようにして、この情報処理装置は、プログラムの実行中に呼び出されるライブラリファイルのパスを切り替えることができる。

【０００５】
また、特許文献２には、実行するアプリケーションプログラムの配給元に応じて、ライブラリファイルのパスの設定を変更する情報処理装置が開示されている。この特許文献２に記載された情報処理装置においては、利用者が、ライブラリのバージョンに関する情報をあらかじめマニフェストファイルなどに記述する。そして、クラスローダが、要求されたクラスをロードする際に、バージョン情報に基づいて、アプリケーションが提供するクラスライブラリか、同名のシステムライブラリかのどちらかを選択する。
【０００６】
また、特許文献３には、オブジェクト指向プログラミング言語で記述されたプログラムを実行する情報処理装置において、希望するバージョンのクラスをロードする情報処理装置が開示されている。この特許文献３に記載された情報処理装置においては、外部定義記憶部が、クラスローダごとに、ロード対象にするクラスの完全修飾クラス名とロード元とが記録されている。そして、委譲モデル介入手段が、クラスローダを実現するためのクラスが生成されたときに、外部定義記憶部に定義されたロード元からクラスをロードする委譲モデル介入用のバイトコードを、生成された当該クラスに挿入する。このようにして、子の情報処理装置は、実行されるプログラムに応じて、クラスライブラリのロード元を切り替えることができる。
【先行技術文献】
【特許文献】
【０００７】
【特許文献１】特開２０１０－１１３４７４号公報
【特許文献２】特開２００７－２０６９６５号公報
【特許文献３】特開２０１３－１９６４５３号公報
【発明の概要】
【発明が解決しようとする課題】
【０００８】
しかしながら、特許文献１乃至３に開示された情報処理装置においては、複数のアプリケーションを実行する際、同じ内容のライブラリであってもアプリケーションごとにロードしてしまうので、メモリ使用量が増大するという問題がある。例えば、これらの情報処理装置において同じアプリケーションを複数実行した場合、クラスローダが、すでにロードされた同じライブラリを、後から実行されたアプリケーションに対してもロードしてしまう。
【０００９】
本発明の一つの目的は、同じ内容のライブラリを多重にロードすることによるメモリ使用量の増大を抑えつつ、複数のアプリケーションが、同名でロード要求され、かつ内容が異なる複数のライブラリを、適切に利用できる情報処理装置等を提供することにある。
【課題を解決するための手段】
【００１０】
上記の目的を達成すべく、本発明の一態様に係る情報処理装置は、以下の構成を備えることを特徴とする。
【００１１】
すなわち、本発明の一態様に係る情報処理装置は、ライブラリファイルごとに、前記ライブラリファイルの内容を識別する識別子情報を生成し、前記ライブラリファイルに含まれるロード要求の対象となりうる部分を表す要求対象情報と前記識別子情報との対応関係を表すロード要求対応情報を生成し、前記識別子情報と前記ロード要求対応情報を記憶装置に保存出力する識別子生成手段と、実行されているアプリケーションから前記ロード要求を受け付け、前記ロード要求、および前記ロード要求対応情報に基づいて、前記ロード要求の対象を含む前記ライブラリファイルの識別子情報を求め、前記ロード要求の対象を表す要求対象情報と、前記求めた識別子情報を出力するロード要求解釈手段と、前記ライブラリファイルに含まれる各部のロード状態を表すロード状態情報を管理し、前記ロード状

態情報に基づいて、前記ロード要求の対象を表す要求対象情報および前記求めた識別子情報が示すロードを要求された部分が前記記憶装置にロードされていないと判断した場合に、前記要求対象情報、および前記求めた識別子情報が示す前記ライブラリファイルから、少なくとも前記ロードを要求された部分をロードし、前記ロード要求に対する応答を返すロード手段とを備える。

【００１２】

また、上記の同目的を達成すべく、本発明の一態様に係るライブラリロード方法は、情報処理装置によって、ライブラリファイルごとに、前記ライブラリファイルの内容を識別する識別子情報を生成し、前記ライブラリファイルに含まれるロード要求の対象となりうる部分を表す要求対象情報と前記識別子情報との対応関係を表すロード要求対応情報を生成し、前記ロード要求対応情報を記憶装置に保存し、実行されているアプリケーションからロード要求を受け付けた際、前記ロード要求、および前記ロード要求対応情報に基づいて、前記ロード要求の対象を含む前記ライブラリファイルの識別子情報を求め、前記ライブラリファイルに含まれる各部のロード状態を表すロード状態情報に基づいて、前記ロード要求の対象を表す要求対象情報、および前記求めた識別子情報が示すロードを要求された部分が前記記憶装置にロードされていないと判断した場合に、前記求めた識別子情報が示す前記ライブラリファイルから、少なくとも前記ロードを要求された部分をロードし、さらに、前記ロード状態情報にロードしたことを登録し、前記ロード要求に対する応答を返す。

【００１３】

また、同目的は、上記の各構成を有する情報処理装置、並びに対応する方法をコンピュータによって実現するコンピュータ・プログラム、およびそのコンピュータ・プログラムが格納されている、コンピュータ読み取り可能な記憶媒体によっても達成される。

【発明の効果】

【００１４】

本発明には、複数のアプリケーションが、メモリ使用量の増大を抑制しながら、同名かつ内容が異なる複数のライブラリを適切に利用できるという効果がある。

【図面の簡単な説明】

【００１５】

【図１】本発明の第１の実施形態に係る情報処理装置１の構成を示すブロック図である。

【図２】本発明の第２の実施形態に係るアプリケーションサーバの構成を示すブロック図である。

【図３】第２の実施形態におけるクラスロード部１１０の構成、および、クラスロード部１１０の各構成と記憶装置１０５に格納された情報との対応関係を示す図である。

【図４】第２の実施形態において、アプリケーション配備部１０１および固有ＩＤ生成部１０２が行うアプリケーションの配備動作を示すフローチャートである。

【図５】第２の実施形態において、アプリケーション１２０および１３０、並びにクラスロード部１１０が行うクラス要求に対するロード動作を示すフローチャートである。

【図６】第２の実施形態において、アプリケーション配備部１０１が行うアプリケーションの配備を解除する動作を示すフローチャートである。

【図７】第２の実施形態におけるパッケージ名対応テーブル１２７の一例を示す図である。

【図８】第２の実施形態におけるパッケージ名対応テーブル１３７の一例を示す図である。

【図９】第２の実施形態における固有ＩＤテーブル１０６の一例を示す図である。

【図１０】本発明の各実施形態、および、その変形例に係る情報処理装置またはアプリケーションサーバに適用可能なコンピュータ（情報処理装置）の構成を例示する図である。

【発明を実施するための形態】

【００１６】

次に、本発明の実施形態について図面を参照して詳細に説明する。

【００１７】
＜第１の実施形態＞
図１は、本発明の第１の実施形態に係る情報処理装置１の構成を示すブロック図である。図１を参照すると、本実施形態に係る情報処理装置１は、識別子生成部２、ロード部３、ロード要求解釈部４、および記憶装置５を含む。
【００１８】
情報処理装置１は、ＣＰＵ（Ｃｅｎｔｒａｌ　Ｐｒｏｃｅｓｓｉｎｇ　Ｕｎｉｔ：図示せず）を用いて実行されるコンピュータ・プログラム（ソフトウェア・プログラム）の制御により動作する一般的な情報処理装置（コンピュータ）によって構成されても良い。または、情報処理装置１の各部が、専用のハードウェアデバイス、または論理回路によって構成されても良い。なお、この情報処理装置１をコンピュータによって実現したハードウェア構成例については、図１０を参照して後述する。
【００１９】
記憶装置５は、例えば、半導体メモリ装置やディスク装置により実現される。記憶装置５は、ロード要求対応情報７、ロード状態情報８を格納することができる。
【００２０】
識別子生成部２は、ライブラリファイル１０ごとに、ライブラリファイル１０の内容を識別する識別子情報を生成する。すなわち、識別子情報は、ライブラリファイル１０の内容の違いを識別することができる情報である。識別子生成部２は、例えば、同名で異なるバージョン（すなわち、異なる内容）のライブラリファイル１０に対して、異なる識別子情報を生成する。識別子生成部２がライブラリファイル１０ごとの内容の違いを識別する方法としては、例えば、ハッシュの生成、または、ファイルの比較のように、良く知られた一般的な技術を採用することができる。
【００２１】
さらに、識別子生成部２は、ライブラリファイル１０に含まれているロード要求の対象となりうる部分を表す要求対象情報と、生成した識別子情報との対応関係を表すロード要求対応情報７を生成する。ここで、「ロード要求の対象となりうる部分」とは、アプリケーションの実行中に、アプリケーションが発行するロード要求１１において指定されるロードの対象であるライブラリファイル１０の部分（プログラム）である。要求対象情報は、アプリケーションの言語に応じて、例えば、クラス名、関数名、または、それらの名前とライブラリ名の組み合わせなどとして表される。なお、ライブラリファイル１０は、１つ以上のロード要求対象となるプログラムを含むことができる。識別子生成部２は、生成したロード要求対応情報７を記憶装置５に保存する。
【００２２】
ロード要求解釈部４は、実行されているアプリケーションから、ロード要求１１を受け付ける。そして、ロード要求解釈部４は、ロード要求１１の対象を表す要求対象情報およびロード要求対応情報７に基づいて、ロード要求１１の対象を含むライブラリファイル１０の識別子情報を求める。すなわち、ロード要求解釈部４は、ロード要求１１が、どのライブラリファイル１０に対して発行されたかを解釈し、解釈したライブラリファイル１０を識別子情報として表す。
【００２３】
ロード部３は、ライブラリファイル１０に含まれる各部のロード状態を表すロード状態情報８を、識別子情報ごとに管理する。そして、ロード部３は、ロード状態情報８に基づいて、ロード要求１１の対象を表す要求対象情報と、ロード要求解釈部４が求めた識別子情報が示すライブラリファイル１０の部分（以下、「ロードを要求された部分」と言う）がロードされているか否かを判断することができる。
【００２４】
ロード部３は、ロード要求１１の対象を表す要求対象情報、ロード要求解釈部４が求めた識別子情報、およびロード状態情報８に基づいて、ロードを要求された部分がまだロードされていないと判断した場合は、識別子情報が示すライブラリファイル１０から、少な

くとも要求対象情報が示す部分をロードする。そして、ロード部３は、少なくともライブラリファイル１０から要求対象情報が示す部分をロードしたことを、ロード状態情報８に登録する。なお、ロード部３は、ロード状態情報８を、記憶装置５に記憶する。そして、ロード部３は、ロード要求１１に対する応答を返す。

【００２５】

一方、ロード部３は、ロードを要求された部分が既にロードされていると判断した場合は、何もロードせずに、ロード要求１１に対する応答を返す。。すなわち、ロード部３は、識別子情報に基づいて、同じ内容のライブラリファイル１０が多重にロードされることを防ぐ。

【００２６】

このように、本実施形態には、複数のアプリケーションが、同名でロード要求され、かつ内容が異なる複数のライブラリを、適切に利用できるという効果がある。また、本実施形態には、同じ内容のライブラリを多重にロードすることによるメモリ使用量の増大を抑えることができるという効果もある。すなわち、本実施形態によれば、同じ内容のライブラリを多重にロードすることによるメモリ使用量の増大を抑えつつ、複数のアプリケーションが、同名でロード要求され、かつ内容が異なる複数のライブラリを、適切に利用できる情報処理装置等を提供することができる。なお、この効果は、ロード要求における対象の名称、または、ライブラリファイルの名称が異なるが同じ内容である場合にも発揮される。すなわち、本実施形態は、どのようなロード要求がなされても、同じ内容のライブラリを多重にロードしないように制御することが可能である。

【００２７】

その理由は、識別子生成部２が生成する識別子情報によって、同名でロード要求されるライブラリファイル１０の内容の違いを識別することができるからである。また、ロード要求解釈部４が、ロード要求１１の対象を含むライブラリファイル１０を、ライブラリファイルの名前ではなく、ライブラリファイルの内容ごとに割り当てられる識別子情報として解釈するからである。そして、ロード部３が、識別子情報に基づいて、同じ内容のライブラリファイル１０が多重にロードされることを防ぐからである。

【００２８】

また、本実施形態には、ライブラリファイル１０またはアプリケーションに関する事前情報を準備する必要がないという効果がある。「背景技術」欄において上述した特許文献１乃至３は、ライブラリおよびプログラムに関する属性、バージョン情報、またはロード元などの事前に準備した情報を情報処理装置に与える手間が掛かるという問題があった。しかし、本実施形態には、そのような事前情報を準備する手間は掛からない。

【００２９】

その理由は、識別子生成部２が、ライブラリファイル１０に含まれているロード要求の対象を表す要求対象情報と、生成した識別子情報との対応関係を表すロード要求対応情報７を、自動的に生成するからである。

【００３０】

なお、本実施形態の変形例としては、以下のようなものが考えられる。

【００３１】

例えば、識別子生成部２は、ロード要求対象情報７を生成する際に、あらかじめ準備された、ライブラリファイル１０に含まれるロード要求の対象となりうる部分に関する情報を利用してもよい。具体的には、例えば、良く使われるライブラリファイル１０に関して、そのライブラリファイル１０に含まれる関数名の一覧情報を作成しておく。識別子生成部２は、ライブラリファイル１０と共にその一覧情報を取得する。そして、識別子生成部２は、ライブラリファイル１０を調査する代わりに、その一覧情報に基づいて、ロード要求対象情報７を生成してもよい。このようにすれば、識別子生成部２は、ライブラリファイル１０から関数名を取得する処理の時間を短縮することができる。なお、本変形例において準備する上記の情報は、例えば、時間を短縮する効果が高い一部のライブラリファイル１０を選んで作成してもよい。

【００３２】
以上、説明したように本変形例には、アプリケーションからロード要求を受ける前に行う処理の時間が短縮できるという効果がある。
【００３３】
その理由は、識別子生成部２が、ロード要求対象情報７を生成する際に、ライブラリファイル１０に対するロード要求の対象となりうる部分の調査を行わずに済むからである。
【００３４】
＜第２の実施形態＞
次に、上述した第１の実施形態を基本とする第２の実施形態について説明する。以下では、第２の実施形態に係る特徴的な部分を中心に説明し、第１の実施形態と同様な構成を有する第２の実施形態の構成要素には、第１の実施形態で付した参照符号と同一の参照符号を付し、その構成要素について重複する詳細な説明は省略する。
【００３５】
本実施形態は、一例として、Ｊａｖａアプリケーションサーバ（以下、単に「アプリケーションサーバ」と言う）に本発明を適用した場合を説明する。本実施形態では、アプリケーションの配備および配備解除も含めて対応できる点が上述した第１の実施形態と異なる。アプリケーションの配備とは、情報処理装置において、アプリケーションに対するリソースの割り当て、およびメモリへのロードなどを行うことにより、アプリケーションを実行可能な状態にすることである。アプリケーションの配備解除とは、配備とは反対に、実行可能な状態にあるアプリケーションに割り当てられたメモリなどのリソースを解放することにより、実行可能な状態を解除することである。
【００３６】
まず、図２を参照して、以下に本実施形態の構成を説明する。図２は、本発明の第２の実施形態に係るアプリケーションサーバの構成を示すブロック図である。
【００３７】
図２を参照すると、本実施形態に係るアプリケーションサーバは、情報処理装置１００、ライブラリ管理ディレクトリ１４０、および記憶装置１５０を含む。ライブラリ管理ディレクトリ１４０および記憶装置１５０は、情報処理装置１００に接続される半導体記憶装置などの記憶装置であってもよい。ライブラリ管理ディレクトリ１４０および記憶装置１５０は、インターネットや構内ＬＡＮ（ローカルエリアネットワーク）等の通信ネットワークを介して通信可能な外部装置であってもよい。また、ライブラリ管理ディレクトリ１４０と、記憶装置１５０とは、例えば、同じ記憶装置であってもよい。
【００３８】
情報処理装置１００は、ＣＰＵ（図示せず）を用いて実行されるコンピュータ・プログラム（ソフトウェア・プログラム）の制御により動作する一般的な情報処理装置（コンピュータ）によって構成されても良い。または、情報処理装置１００の各部が、専用のハードウェアデバイス、または論理回路によって構成されても良い。なお、この情報処理装置１００をコンピュータによって実現したハードウェア構成例については、図１０を参照して後述する。
【００３９】
記憶装置１５０は、本実施形態に係るアプリケーションサーバにおいて配備、実行、および配備解除されるアプリケーションであるアプリケーション１２０および１３０を格納することができる。アプリケーション１２０（１３０）は、アプリケーションプログラム１２１（１３１）と、１つ以上のライブラリファイル１２２（１３２）を含む。
【００４０】
アプリケーションプログラム１２１（１３１）は、実行可能なプログラムである。アプリケーションプログラム１２１（１３１）は、自身の実行中に、第１の実施形態におけるロード要求１１を発行することにより、ライブラリファイル１２２（１３２）を参照（利用）する。
【００４１】

ライブラリファイル１２２（１３２）は、アプリケーションプログラム１２１（１３１）に参照されるクラス群を含むライブラリファイルである。ライブラリファイル１２２（１３２）は、第１の実施形態におけるライブラリファイル１０を基本とする。

【００４２】

情報処理装置１００は、アプリケーション配備部１０１、固有ＩＤ生成部１０２、クラスロード部１１０、および記憶装置１０５を含む。記憶装置１０５は、例えば、半導体メモリ装置やディスク装置により実現される。記憶装置１０５は、固有ＩＤテーブル（ライブラリ識別子管理情報）１０６、パッケージ対応テーブル１２７および１３７、およびクラス管理領域１０８を格納することができる。図３を参照すると、クラスロード部１１０は、上位クラスローダ群１１１、共通クラスローダ１１３、およびアプリケーションクラスローダ１２４および１３４を含む。図３は、第２の実施形態におけるクラスロード部１１０の構成、および、クラスロード部１１０の各構成と記憶装置１０５に格納された情報との対応関係を示す図である。

【００４３】

情報処理装置１００は、第１の実施形態に係る情報処理装置１を基本とする各要素に加えて、アプリケーション配備部１０１および固有ＩＤテーブル１０６を含む。本実施形態において、第１の実施形態における識別子情報は、一例として、ライブラリファイル１２２（１３２）のハッシュ情報に基づいて生成される「固有ＩＤ」として実現する。以下では、「識別子情報」を「固有ＩＤ」と読み替えて説明する。例えば、第１の実施形態における識別子生成部２を基本とする識別子生成部１０２は、以下では、固有ＩＤ生成部１０２と呼ばれる。また、本実施形態では、第１の実施形態におけるロード要求対応情報７は、アプリケーションごとにパッケージ対応テーブル１２７または１３７に格納される。

【００４４】

アプリケーション配備部１０１は、記憶装置１５０に格納されるアプリケーション１２０および１３０を配備することができる。アプリケーション配備部１０１は、アプリケーション１２０（１３０）を配備する際、アプリケーションごとにアプリケーションクラスローダ１２４（１３４）を生成する。アプリケーションクラスローダ１２４（１３４）の詳細は、クラスロード部１１０の説明において後述する。

【００４５】

また、アプリケーション配備部１０１は、固有ＩＤ生成部１０２に対してライブラリファイル１２２（１３２）を渡すことにより、固有ＩＤおよびパッケージ対応テーブル１２７および１３７の生成を指示する。

【００４６】

また、アプリケーション配備部１０１は、固有ＩＤ生成部１０２によって生成された固有ＩＤ（識別子情報）と、固有ＩＤに対応するライブラリファイル１２２（１３２）を参照するアプリケーションの数を表す参照数情報とを含むライブラリ識別子管理情報１０６を、記憶装置１０５に記憶する。本実施形態では、具体例の一つとして、ライブラリ識別子管理情報１０６を、固有ＩＤテーブル１０６として実現する。図９は、第２の実施形態における固有ＩＤテーブル１０６の一例を示す図である。図９を参照すると、固有ＩＤテーブル１０６は、固有ＩＤをキーとして、固有ＩＤに対応するライブラリファイル１２２（１３２）を参照するアプリケーションの数を表す参照数情報を、そのキーに対する値として格納することができる。

【００４７】

さらに、アプリケーション配備部１０１は、ライブラリファイル１２２（１３２）のコピーを、ライブラリ管理ディレクトリ１４０に保存する。

【００４８】

また、アプリケーション配備部１０１は、配備されたアプリケーション１２０および１３０の配備を解除することができる。すなわち、アプリケーション配備部１０１は、ライブラリ識別子管理情報に基づいて、配備解除の結果、参照しているアプリケーション１２０（１３０）が無くなるライブラリファイル１２２（１３２）に関するリソースの削除を

行う。その場合、さらに、アプリケーション配備部１０１は、配備を解除するアプリケーション１２０（１３０）に関するリソースの削除を行う。削除されるリソースには、記憶情報などが含まれる。削除されるリソースとは、具体的には、アプリケーションローダ１２４（１３４）、パッケージ名対応テーブル１２７（１３７）、および、ライブラリ管理ディレクトリ１４０にコピーされたライブラリファイル１２２（１３２）などが含まれる。

【００４９】

固有ＩＤ生成部１０２は、第１の実施形態における識別子生成部２を基本とする。本実施形態では、固有ＩＤ生成部１０２は、アプリケーション配備部１０１からライブラリファイル１２２（１３２）を受け取ることにより、固有ＩＤ（識別子情報）とロード要求対応情報７を生成する。なお、図９における「項番」の欄は、説明を明確にする目的で付した欄である。すなわち、「項番」の欄は、実際には設けられる必要はない。

【００５０】

固有ＩＤ生成部１０２は、生成したロード要求対応情報７を、アプリケーション１２０（１３０）ごとのパッケージ名対応テーブル１２７（１３７）（図３）に格納する。パッケージ名対応テーブル１２７（１３７）は、アプリケーション１２０（１３０）が利用するライブラリファイル１２２（１３２）を構成する、１つ以上のファイルに対するロード要求対応情報７を含むことができる。

【００５１】

パッケージ名対応テーブル１２７（１３７）は、ライブラリファイル１２２（１３２）が含むすべてのパッケージ名と、固有ＩＤとの対応関係を表すロード要求対応情報１０７を含む。パッケージ名とは、一群のクラスを束ねるグループの名称である。１つのライブラリファイル１２２（１３２）は、複数のパッケージを含むことができる。Ｊａｖａにおいて、ロード要求は、パッケージ名とクラス名とを含む完全修飾クラス名に対するクラスの要求（クラス要求）として発行される。すなわち、本実施形態において、クラス要求は、第１の実施形態におけるロード要求１１に対応する。本実施形態におけるロード要求対応情報７は、ロード要求と、識別子情報（ライブラリファイル）との対応関係を、ロードが要求される単位であるクラスではなく、クラスが含まれるパッケージ名（グループ名）によって管理する事例の一つである。

【００５２】

パッケージ名対応テーブル１２７（１３７）の一例を、図７および図８に示す。図７は、第２の実施形態におけるパッケージ名対応テーブル１２７の一例を示す図である。図８は、第２の実施形態におけるパッケージ名対応テーブル１３７の一例を示す図である。パッケージ名対応テーブル１２７（１３７）に含まれる各ロード要求対応情報１０７は、図７および図８に示すように、「パッケージ名」をキーとして、そのパッケージ（クラスのグループ）を含むライブラリファイル１２２（１３２）に対応する「固有ＩＤ」を、そのキーに対する値として格納することができる。なお、図７および図８における「項番」の欄は、説明を明確にする目的で付した欄である。すなわち、「項番」の欄は、実際には設けられる必要はない。

【００５３】

固有ＩＤ生成部１０２の構造と内容は、上述した点以外は、第１の実施形態における識別子生成部２と同様であるので、重複する詳細な説明は省略する。

【００５４】

クラスロード部１１０は、上述した通り、上位クラスローダ群１１１、共通クラスローダ１１３、およびアプリケーションクラスローダ１２４および１３４を含む。クラスロード部１１０のこの構造は、Ｊａｖａにおけるクラスローダの階層構造に対応している。Ｊａｖａにおける一般的なクラスローダにおける動作と問題点の詳細については、後述する。ここでは、クラスローダ１１０に含まれる各部における、本実施形態の実現に関連する動作に関して説明する。

【００５５】

上位クラスローダ群１１１は、アプリケーション１２０（１３０）が動作する際に使用するクラスのうち、ライブラリファイル１２２（１３２）に含まれないシステムライブラリなどからロードするクラスに関するロード要求に対応する。すなわち、上位クラスローダ群１１１は、Ｊａｖａにおける一般的なクラスローダに含まれる、共通クラスローダより上位のクラスローダと同等である。動作の詳細は、後述するＪａｖａにおける一般的なクラスローダにおける動作と問題点の詳細にて説明する。

【００５６】

共通クラスローダ１１３は、第１の実施形態におけるロード部３を基本とする。また、アプリケーションクラスローダ１２４（１３４）は、第１の実施形態におけるロード要求解釈部４を基本とする。

【００５７】

本実施形態では、アプリケーションクラスローダ１２４（１３４）は、アプリケーション配備部１０１によるアプリケーション１２０（１３０）の配備の後、実行されたアプリケーション１２０（１３０）から、クラス要求を受ける。アプリケーションクラスローダ１２４（１３４）は、クラス要求とパッケージ名対応テーブル１２７（１３７）とに基づいて、クラス要求に対応するライブラリファイル１２２（１３２）の固有ＩＤを求める。アプリケーションクラスローダ１２４（１３４）は、クラス要求の対象である完全修飾クラス名（第１の実施形態における「ロード要求１１の対象を表す要求対象情報」）と、求めた固有ＩＤとを共通クラスローダ１１３に対して出力する。

【００５８】

共通クラスローダ１１３は、クラス管理領域１０８に基づいて、第１の実施形態におけるロード部３と同様に、クラス要求の対象であるクラスのロード状況に応じた処理を行う。

【００５９】

クラス管理領域１０８は、第１の実施形態におけるロード状態情報８を基本とする。本実施形態において、共通クラスローダ１１３は、一例として、クラス管理領域１０８にクラスをロードすることによって、そのクラスがロードされたということを登録する。すなわち、クラス管理領域１０８は、クラスをロード済みであるという情報に加えて、さらにロードされたクラスの実体であるプログラムコードを含む。

【００６０】

さらに、本実施形態では、アプリケーションクラスローダ１２４（１３４）および共通クラスローダ１１３は、ライブラリファイル１２２（１３２）に含まれないクラスに対するクラス要求の処理を含む。すなわち、アプリケーションクラスローダ１２４（１３４）および共通クラスローダ１１３は、そのようなクラス要求への対応を、上位クラスローダ群１１１に委譲する（任せる）ことによって、処理する。

【００６１】

アプリケーションクラスローダ１２４（１３４）、および共通クラスローダ１１３の構造と内容は、上述した点以外は、第１の実施形態におけるロード要求解釈部４およびロード部３と同様であるので、重複する詳細な説明は省略する。

【００６２】

次に、ライブラリ管理ディレクトリ１４０について説明する。ライブラリ管理ディレクトリ１４０は、アプリケーション配備部１０１によってコピーされるライブラリファイル１２２（１３２）を格納することができる。また、ライブラリ管理ディレクトリ１４０は、共通クラスローダ１１３がクラスをロードする際に、格納したライブラリファイル１２２（１３２）を共通クラスローダ１１３に対して出力することができる。

【００６３】

次に、上述した構成を備える本実施形態の動作について詳細に説明する。以下の動作の説明においては、ロード要求１１を「クラス要求」と呼ぶ。本実施形態における動作は、アプリケーションを配備する動作（配備動作）、クラス要求に対するロードの動作（ロード動作）、および、アプリケーションの配備を解除する動作（配備解除動作）がある。

【００６４】
まず、図４を参照して、アプリケーションの配備動作を説明する。図４は、アプリケーション配備部１０１および固有ＩＤ生成部１０２が行うアプリケーションの配備動作を示すフローチャートである。
【００６５】
はじめに、以下において説明する具体例における前提条件を述べる。本実施形態では、具体例として、アプリケーション１２０および１３０は、２つの同名のライブラリファイル１２２および１３２を含むことを前提とする。そして、２つの同名のライブラリファイル１２２および１３２のうちの一方のファイルは、内容が異なることを前提とする。また、２つの同名のライブラリファイル１２２および１３２のうちの他方のファイルは、同じ内容であることを前提とする。
【００６６】
また、アプリケーション配備動作を開始する時点で、クラスロード部１１０における上位クラスローダ群１１１および共通クラスローダ１１３が、図示しない主記憶装置などにおいて常駐していることを前提とする。アプリケーション配備部１０１および固有ＩＤ生成部１０２は、動作を行うときに起動してもよい。また、固有ＩＤテーブル１０６、パッケージ対応テーブル１２７および１３７、および、クラス管理領域１０８には、何も記憶されていないことを前提とする。
【００６７】
こうした前提条件下で、アプリケーション１２０の配備を行う場合の動作を説明する。
【００６８】
まず、アプリケーション配備部１０１が、図示しない入力装置等を介して、利用者から、アプリケーション１２０の配備の指示を受ける。すると、アプリケーション配備部１０１は、クラスロード部１１０の共通クラスローダ１１３の配下として、アプリケーション１２０に対応するアプリケーションローダ１２４を生成する（ステップＳ１０）。
【００６９】
ライブラリファイル１２２が複数ある場合は、ステップＳ１１からＳ１５までの動作は、各ライブラリファイル１２２ごとに実行してもよい。本具体例では、ライブラリファイル１２２は２つあるので、ステップＳ１１からＳ１５までの動作は、各ライブラリファイル１２２に対して、計２回が実行される。以下、ステップＳ１１からステップＳ１５までにおいて、「ライブラリファイル１２２」を「ライブラリファイル１２２の１つ」と読み替えることができる。
【００７０】
次に、アプリケーション配備部１０１は、固有ＩＤ生成部１０２に対して、アプリケーション１２０に含まれるライブラリファイル１２２を渡すとともに、固有ＩＤおよびロード要求対応情報７（パッケージ対応テーブル１２７）の生成を指示する。固有ＩＤ生成部１０２は、まず、ライブラリファイル１２２ごとの固有ＩＤを生成する（ステップＳ１１）。具体的には、固有ＩＤ生成部１０２は、一例として、ハッシュ関数によって、ライブラリファイル１２２に対するハッシュ情報を求める。そして、固有ＩＤ生成部１０２は、求めたハッシュ情報を、そのライブラリファイル１２２に対する固有ＩＤとする。
【００７１】
次に、固有ＩＤ生成部１０２は、ライブラリファイル１２２に含まれるパッケージ名と固有ＩＤとの対応関係をパッケージ名対応テーブル１２７に登録する（ステップＳ１２）。具体的には、固有ＩＤ生成部１０２は、まず、ライブラリファイル１２２に含まれる全てのクラスのパッケージ名を抽出する。そして、固有ＩＤ生成部１０２は、抽出したパッケージ名をキーとして、ステップＳ１１で生成した固有ＩＤをそのキーに対する値として、パッケージ対応テーブル１２７に登録する。パッケージ対応テーブル１２７は、初めて情報が登録される前に、固有ＩＤ生成部１０２よって生成してもよい。
【００７２】
ステップＳ１２の最後として、固有ＩＤ生成部１０２は、固有ＩＤをアプリケーション

配備部１０１に返す。アプリケーション配備部１０１は、生成された固有ＩＤが固有ＩＤテーブル１０６に登録済みであるかどうかを調べる（ステップＳ１３）。この具体例では、ライブラリファイル１２２に対応する固有ＩＤは、固有ＩＤテーブル１０６に登録されていない。

【００７３】

まだ、固有ＩＤが登録されていない場合（ステップＳ１３のＮＯ）、アプリケーション配備部１０１は、参照数情報を「１」として、生成した固有ＩＤを固有ＩＤテーブル１０６に登録する。（ステップＳ２０）。そして、アプリケーション配備部１０１は、ライブラリファイル１２２をライブラリ管理ディレクトリ１４０にコピーする（ステップＳ２１）。ライブラリファイル１２２をコピーする際、アプリケーション配備部１０１は、固有ＩＤに基づいて、ライブラリファイル１２２を検索できるようにコピーする。一例として、アプリケーション配備部１０１は、ライブラリファイル１２２をコピーしたファイルを、固有ＩＤを含む名前のファイルとして保存する。さらに、アプリケーション配備部１０１は、固有ＩＤに対応するクラス管理領域１０８を生成する（ステップＳ２２）。

【００７４】

このような処理の後、まだ処理していないライブラリファイル１２２があれば、アプリケーション配備部１０１は、未処理のライブラリファイル１２２に対して、ステップＳ１１以降の処理を行う（ステップＳ１５）。

【００７５】

図７は、このようにして生成されたアプリケーション１２０に対応するパッケージ対応テーブル１２７の一例である。例えば、図７において「項番」欄が「Ａ１」のロード要求対応情報７（以下、「Ａ１の項」のように言う）は、固有ＩＤが「０ｃ４１ｆｅｅ１ｂ５０ｂ２２ａｃ２３ａｃ９９３８８ｆ６４９ｂ９ａ６３４ｅ８３４６」のライブラリファイル１２２に、「ｏｒｇ．ａｐａｃｈｅ．ｃｏｍｍｏｎｓ．ｂｅａｎｕｔｉｌｓ．ｃｏｎｖｅｒｔｅｒｓ」というパッケージ名のパッケージが含まれていたことを表す。固有ＩＤ欄が同一であるＡ１からＡ４の４つの項は、固有ＩＤが「０ｃ４１ｆｅｅ１ｂ５０ｂ２２ａｃ２３ａｃ９９３８８ｆ６４９ｂ９ａ６３４ｅ８３４６」のライブラリファイル１２２に含まれる４つのパッケージに関する情報である。また、Ａ５およびＡ６の項は、固有ＩＤが「ｆ６ｆ６６ｅ９６６ｃ７０ａ８３ｆｆｂｄｂ６ｆ１７ａ０９１９ｅａｆ７ｃ８ａｃａ７ｆ」のライブラリファイル１２２に含まれる２つのパッケージに対する情報である。

【００７６】

このようにして、アプリケーション１２０は配備される。

【００７７】

アプリケーション１２０の配備に続いて、アプリケーション１３０が配備される場合の動作を説明する。上述した通り、アプリケーション１３０は、ライブラリファイル１２２と同名であるが、内容が異なるライブラリファイル１３２を含む。また、アプリケーション１３０は、ライブラリファイル１２２と同名で内容も同じ、もう１つのライブラリファイル１３２を、さらに含む。

【００７８】

ライブラリファイル１２２と同名で同内容のライブラリファイル１３２に対する配備の動作は、アプリケーション１２０の配備動作と同様である。また、ライブラリファイル１２２と同名で内容が異なるライブラリファイル１３２に対するステップＳ１０からＳ１２までの動作は、上述したアプリケーション１２０の配備と同様である。なお、ステップＳ１１において、固有ＩＤ生成部１０２は、ライブラリファイル１２２と同名で内容が異なるライブラリファイル１３２に対しては、新たな固有ＩＤを生成する。また、固有ＩＤ生成部１０２は、ライブラリファイル１２２と同名同内容のライブラリファイル１３２に対しては、そのライブラリファイル１２２と同じ固有ＩＤを生成する。

【００７９】

ステップＳ１３において、ライブラリファイル１２２と同名で内容が異なるライブラリファイル１３２に対する固有ＩＤは、固有ＩＤテーブル１０６に既に登録されている。固

有ＩＤが登録されていた場合（ステップＳ１３のＹＥＳ）、アプリケーション配備部１０１は、その固有ＩＤに対応する参照数情報に「１」を加えて、固有ＩＤテーブル１０６を更新する（ステップＳ１４）。図９は、アプリケーション１３０の配備後における固有ＩＤテーブル１０６の一例である。図９におけるＣ１の項が、ライブラリファイル１２２およびライブラリファイル１３２の両方に含まれる、同名同内容のライブラリファイル１２２（１３２）に対して、参照数を「２」として更新した情報である。そして、アプリケーション配備部１０１は、まだ処理していないライブラリファイル１３２があれば、ステップＳ１１以降の処理を行う（ステップＳ１５）。

【００８０】

図８は、このようにして生成されたアプリケーション１３０に対応するパッケージ対応テーブル１３７の一例である。図８において、Ｂ１からＢ４の項は、アプリケーション１２０のライブラリファイル１２２と同名同内容のライブラリファイル１３２に対する情報である。また、Ｂ５およびＢ６の項は、ライブラリファイル１２２と同名で内容が異なるライブラリファイル１３２（固有ＩＤ「３７ｃ６５９ｅ５７２９３６５６ｅｂｅｆ１ａ２４７ｆｃ６ｃｅｂ７３８ｅｂｄｆｃ７４」）に含まれる２つのパッケージに対する情報である。

【００８１】

このようにして、アプリケーション１３０は配備される。

【００８２】

なお、アプリケーション配備部１０１は、図４に示す一連のアプリケーション配備動作の前後を含む任意のタイミングにおいて、一般的なＪａｖａのアプリケーションサーバにおいて実行されるアプリケーション配備の動作を行ってもよい。

【００８３】

以上が、アプリケーションの配備動作である。

【００８４】

次に、図５を参照して、クラス要求に対するロード動作を説明する。図５は、第２の実施形態において、アプリケーション１２０および１３０、並びにクラスロード部１１０が行うクラス要求に対するロード動作を示すフローチャートである。クラス要求は、配備されたアプリケーションプログラム１２１および１３１がそれぞれ起動された後に発行される。なお、以下では、「実行されたアプリケーションプログラム１２１（１３１）」を、簡単に「アプリケーション１２０（１３０）」と記述する。

【００８５】

まず、内容が異なるライブラリファイルに対して、複数のアプリケーションから同名でクラス要求された場合における、ロード動作を説明する。始めに、アプリケーション１２０が、「ｏｒｇ．ａｐａｃｈｅ．ｃｏｍｍｏｎｓ．ｌｏｇｇｉｎｇ．ＬｏｇＦａｃｔｏｒｙ」クラスを要求したときのロード動作を説明する。このクラス要求における完全修飾クラス名は、「ｏｒｇ．ａｐａｃｈｅ．ｃｏｍｍｏｎｓ．ｌｏｇｇｉｎｇ．ＬｏｇＦａｃｔｏｒｙ」である。また、このクラス要求におけるパッケージ名は、末尾の「．ＬｏｇＦａｃｔｏｒｙ」を除いた前方の部分である「ｏｒｇ．ａｐａｃｈｅ．ｃｏｍｍｏｎｓ．ｌｏｇｇｉｎｇ」である。

【００８６】

このクラスを含むライブラリファイル１２２および１３２は、同名であるが内容が異なる。したがって、このクラス要求におけるパッケージ名は、パッケージ名対応テーブル１２７（図７）のＡ５の項、およびパッケージ名対応テーブル１３７（図８）のＢ５の項に、異なる固有ＩＤと対応づけて登録されている。以下では、パッケージ名対応テーブル１２７および１３７、または固有ＩＤテーブル１０６から得られた項目を、図７～９における項番によって、簡易的に記述する。例えば、パッケージ名対応テーブル１２７のＡ１の項から得られた「固有ＩＤ」欄の情報を、「Ａ１項の固有ＩＤ」と言う。

【００８７】

アプリケーション１２０が、上記のクラスを要求する（ステップＳ３０）。

【００８８】
クラスロード部１１０においては、まず各アプリケーション１２０（１３０）に対応するアプリケーションクラスローダ１２４（１３４）が、クラスの要求を受ける（ステップＳ４０）。具体的には、アプリケーションクラスローダ１２４が、アプリケーション１２０が発行したクラスの要求を受ける。
【００８９】
アプリケーションクラスローダ１２４（１３４）は、対応するパッケージ名対応テーブル１２７（１３７）に要求されたクラスのパッケージ名があるかどうかを調べる（ステップＳ４１）。具体的には、アプリケーションクラスローダ１２４が、アプリケーション１２０に対応するパッケージ名対応テーブル１２７を、要求されたクラスのパッケージ名「ｏｒｇ．ａｐａｃｈｅ．ｃｏｍｍｏｎｓ．ｌｏｇｇｉｎｇ」をキーとして検索する。アプリケーションクラスローダ１２４は、図７のＡ５の項から、固有ＩＤ（Ａ５項の固有ＩＤ）が得られることにより、パッケージ名対応テーブル１２７に要求されたクラスのパッケージ名が含まれていると判断する。
【００９０】
クラスのパッケージ名がパッケージ名対応テーブル１２７（１３７）にある場合（ステップＳ４１のＹＥＳ）、アプリケーションクラスローダ１２４（１３４）は、クラス要求の対象である完全修飾クラス名と、求めた固有ＩＤとを、共通クラスローダ１１３に対して出力する。共通クラスローダ１１３は、固有ＩＤに対応するクラス管理領域１０８に、クラス要求の対象のクラスがロードされているかどうかを調べる（ステップＳ４２）。具体例では、クラス管理領域１０８には、まだ何もロードされていない。したがって、共通クラスローダ１１３は、要求されたクラスはロードされていないと判断する。
【００９１】
要求されたクラスがロードされていない場合（ステップＳ４２のＮＯ）、共通クラスローダ１１３は、固有ＩＤに対応するライブラリファイル１２２（１３２）から、クラスをロードする（ステップＳ４３）。具体的には、共通クラスローダ１１３は、ライブラリ管理ディレクトリ１４０から、Ａ５項の固有ＩＤをファイル名としてコピーされているライブラリファイル１２２を取得する。そして、共通クラスローダ１１３は、取得したライブラリファイル１２２から、要求されたクラスを含むプログラムを、クラス管理領域１０８にロードする。
【００９２】
最後に、共通クラスローダ１１３は、アプリケーションクラスローダ１２４（１３４）を介して、クラス要求に対する応答を返す（ステップＳ４４）。具体的には、共通クラスローダ１１３は、クラスがロードされた位置の情報を含むオブジェクトを、アプリケーションクラスローダ１２４を介して、アプリケーション１２０に対して返す。
【００９３】
アプリケーション１２０（１３０）は、クラス要求に対する応答を受ける（ステップＳ３１）。
【００９４】
続いて、アプリケーション１３０が、上記と同じ「ｏｒｇ．ａｐａｃｈｅ．ｃｏｍｍｏｎｓ．ｌｏｇｇｉｎｇ．ＬｏｇＦａｃｔｏｒｙ」クラスを要求したときのロード動作を説明する。以下では、上記の動作と異なる点について、簡単に説明する。
【００９５】
まず、アプリケーション１３０が、アプリケーション１２０と同じ完全修飾クラス名に対するクラスを要求する（ステップＳ３０）。
【００９６】
クラスロード部１１０においては、アプリケーションクラスローダ１３４がクラスの要求を受ける（ステップＳ４０）。
【００９７】
アプリケーションクラスローダ１３４は、パッケージ名対応テーブル１３７から、要求

されたクラスのパッケージ名に対応する固有ＩＤ（Ｂ５項の固有ＩＤ）を得ることにより、要求されたクラスのパッケージ名が含まれていると判断する（ステップＳ４１）。そして、アプリケーションクラスローダ１３４は、共通クラスローダ１１３に対して、完全修飾クラス名とＢ５項の固有ＩＤを出力する（ステップＳ４１のＹＥＳ）。なお、上述の通り、Ｂ５項の固有ＩＤは、Ａ５項の固有ＩＤとは異なる。

【００９８】

共通クラスローダ１１３は、Ｂ５項の固有ＩＤに対応するクラス管理領域１０８に、まだ完全修飾クラス名に対応するクラスがロードされていないと判断する（ステップＳ４２）。

【００９９】

要求されたクラスがロードされていない場合（ステップＳ４２のＮＯ）、共通クラスローダ１１３は、上記と同様に、Ｂ５項の固有ＩＤに対応するライブラリファイル１３２から、要求されたクラスを含むプログラムを、クラス管理領域１０８にロードする（ステップＳ４３）。

【０１００】

最後に、共通クラスローダ１１３は、アプリケーションクラスローダ１３４を介して、クラスが新たにロードされた位置の情報を含むオブジェクトを、クラス要求に対する応答として返す（ステップＳ４４）。

【０１０１】

このようにして、クラスロード部１１０は、内容が異なるライブラリファイルに対して、複数のアプリケーションから同名でクラス要求された場合に、それぞれのアプリケーションに対応するライブラリファイルを区別してロードすることができる。

【０１０２】

次に、複数のアプリケーションから同じ内容のライブラリファイルに対するロードが要求された場合における、ロード動作を説明する。一例として、アプリケーション１２０および１３０が、続けて「ｏｒｇ．ａｐａｃｈｅ．ｃｏｍｍｏｎｓ．ｂｅａｎｕｔｉｌｓ．ｃｏｎｖｅｒｔｅｒｓ．ＤａｔｅＣｏｎｖｅｒｔｅｒ」クラスを要求したときのロード動作を説明する。

【０１０３】

このクラスを含むライブラリファイル１２２および１３２は、同名かつ同内容である。したがって、このパッケージ名「ｏｒｇ．ａｐａｃｈｅ．ｃｏｍｍｏｎｓ．ｂｅａｎｕｔｉｌｓ．ｃｏｎｖｅｒｔｅｒｓ」は、パッケージ名対応テーブル１２７（図７）のＡ１の項、およびパッケージ名対応テーブル１３７（図８）のＢ１の項に、同じ固有ＩＤと対応づけて登録されている。

【０１０４】

まず、アプリケーション１２０が、上記のクラスを要求する（ステップＳ３０）。

【０１０５】

クラスロード部１１０において、アプリケーションクラスローダ１２４が、クラスの要求を受け（ステップＳ４０）、さらに、パッケージ名対応テーブル１２７から、要求されたクラスのパッケージ名に対応する固有ＩＤ（Ａ１項の固有ＩＤ）を得る（ステップＳ４１）。そして、アプリケーションクラスローダ１２４は、共通クラスローダ１１３に対して、完全修飾クラス名とＡ１項の固有ＩＤを出力する（ステップＳ４１のＹＥＳ）。

【０１０６】

共通クラスローダ１１３は、クラス管理領域１０８に、まだ当該クラスがロードされていないと判断し（ステップＳ４２）、Ａ１項の固有ＩＤに対応するライブラリファイル１２２から、そのクラスを含むプログラムをロードする（ステップＳ４３）。そして、共通クラスローダ１１３は、アプリケーションクラスローダ１２４を介して、クラス要求に対する応答を返す（ステップＳ４４）。

【０１０７】

続いて、アプリケーション１３０が、アプリケーション１２０と同じ「ｏｒｇ．ａｐａ

ｃｈｅ．ｃｏｍｍｏｎｓ．ｂｅａｎｕｔｉｌｓ．ｃｏｎｖｅｒｔｅｒｓ．ＤａｔｅＣｏｎｖｅｒｔｅｒ」クラスを要求したときのロード動作を説明する。

【０１０８】

まず、アプリケーション１３０が、アプリケーション１２０と同じ完全修飾クラス名に対するクラスを要求する（ステップＳ３０）。

【０１０９】

クラスロード部１１０において、アプリケーションクラスローダ１３４が、クラスの要求を受け（ステップＳ４０）、パッケージ名対応テーブル１３７から、要求されたクラスのパッケージ名に対応する固有ＩＤ（Ｂ１項の固有ＩＤ）を得る（ステップＳ４１）。そして、アプリケーションクラスローダ１３４は、共通クラスローダ１１３に対して、完全修飾クラス名とＢ１項の固有ＩＤを出力する（ステップＳ４１のＹＥＳ）。なお、上述の通り、Ｂ１項の固有ＩＤは、Ａ１項の固有ＩＤと同じである。

【０１１０】

共通クラスローダ１１３は、Ｂ１項の固有ＩＤに対応するクラス管理領域１０８に、すでに完全修飾クラス名に対応するクラスがロードされていると判断する（ステップＳ４２）。

【０１１１】

要求されたクラスがロードされている場合（ステップＳ４２のＹＥＳ）、共通クラスローダ１１３は、アプリケーションクラスローダ１３４を介して、クラスがロードされた位置の情報を含むオブジェクトを、クラス要求に対する応答として返す（ステップＳ４４）。この応答に含まれるクラスがロードされた位置の情報は、先にアプリケーション１２０から受けたクラスをロードした際（ステップＳ４３）に、Ａ１項の固有ＩＤに対応するライブラリファイル１２２からロードされた位置の情報である。

【０１１２】

このようにして、クラスロード部１１０は、同じ内容のライブラリファイル１２２（１３２）を多重にロードすることを回避することができる。

【０１１３】

次に、ライブラリファイル１２２および１３２に含まれないクラスが要求された場合における、ロード動作を説明する。一例として、アプリケーション１２０または１３０が、「ｆｏｏ．Ｂａｒ」クラスを要求したときのロード動作を説明する。なお、「ｆｏｏ」というクラスは、ライブラリファイル１２２にも１３２にも含まれていない。

【０１１４】

アプリケーション１２０が、上記のクラスを要求する（ステップＳ３０）。

【０１１５】

クラスロード部１１０においては、アプリケーションクラスローダ１２４（１３４）が、クラスの要求を受け（ステップＳ４０）、さらに、パッケージ名対応テーブル１２７（１３７）を「ｆｏｏ」というパッケージ名で検索する（ステップＳ４１）。しかし、アプリケーションクラスローダ１２４（１３４）は、「ｆｏｏ」というパッケージ名に対応する固有ＩＤが得られないので（ステップＳ４１のＮＯ）、共通クラスローダ１１３を介して、上位クラスローダ群１１１にロード処理を委譲する（ステップＳ５０）。具体的には、アプリケーションクラスローダ１２４（１３４）は、共通クラスローダ１１３に対して、要求されたクラスの完全修飾クラス名「ｆｏｏ．Ｂａｒ」のみを通知する。共通クラスローダ１１３は、固有ＩＤが通知されない場合、「ｆｏｏ．Ｂａｒ」クラスのロードを、さらに上位にある上位クラスローダ群１１１に委譲する。

【０１１６】

これ以降の動作は、一般的なＪａｖａクラスローダの動作に従う。すなわち、その後、共通クラスローダ１１３は、上位クラスローダ群１１１からクラス要求に対する応答を受ける（ステップＳ５１）。そして、共通クラスローダ１１３は、その応答に基づき、アプリケーションクラスローダ１２４（１３４）を介して、アプリケーション１２０（１３０）に対する応答を返す（ステップＳ４４）。

【０１１７】
このようにして、クラスロード部１１０は、一般的なＪａｖａのクラスロードの仕組みと、本実施形態の動作を併存させることができる。
【０１１８】
以上が、クラス要求に対するロード動作である。
【０１１９】
次に、図６を参照して、アプリケーションの配備解除動作を説明する。図６は、第２の実施形態において、アプリケーション配備部１０１が行うアプリケーションの配備を解除する動作を示すフローチャートである。以下では、上述したアプリケーション１２０および１３０が配備された状態から、アプリケーション１２０または１３０の配備が解除される場合の動作である。
【０１２０】
まず、アプリケーション配備部１０１が、図示しない入力装置等を介して、利用者から、アプリケーション１２０または１３０の配備を解除する指示を受ける。すると、アプリケーション配備部１０１は、配備を解除するアプリケーション１２０または１３０に対応するパッケージ名対応テーブル１２７または１３７に対して、そこに含まれる未処理の固有ＩＤを選択する（ステップＳ６０）。すなわち、アプリケーション配備部１０１は、パッケージ名対応テーブル１２７または１３７に含まれる固有ＩＤごとに、ステップＳ６１からＳ６３までの動作を行う。
【０１２１】
アプリケーション配備部１０１は、固有ＩＤ管理テーブル１０６において、選択した固有ＩＤに対応する参照数情報から「１」を減らす（ステップＳ６１）。
【０１２２】
次に、アプリケーション配備部１０１は、ステップＳ６１で減算した結果、その参照数情報が「１」以上であるかどうかを判断する（ステップＳ６２）。
【０１２３】
参照数情報が「１」以上の場合（ステップＳ６２のＹＥＳ）、アプリケーション配備部１０１は、当該固有ＩＤに対応するライブラリファイル１２２または１３２を含むアプリケーション１２０または１３０があるので、その固有ＩＤに関連する配備処理を終了する。そして、まだ処理していない固有ＩＤがあれば、アプリケーション配備部１０１は、未処理の固有ＩＤに対して、ステップＳ６１に戻る（ステップＳ６３）。
【０１２４】
一方、参照数情報が「１」以上でない（すなわち「０」である）場合（ステップＳ６２のＮＯ）、アプリケーション配備部１０１は、当該固有ＩＤに対応するライブラリファイル１２２または１３２を含むアプリケーションがなくなるので、そのライブラリファイル１２２または１３２に関する情報やリソースを削除する。
【０１２５】
すなわち、アプリケーション配備部１０１は、固有ＩＤ管理テーブル１０６から、当該固有ＩＤのエントリを削除する（ステップＳ７０）。また、アプリケーション配備部１０１は、当該固有ＩＤ管理テーブルに対応するクラス管理領域１０８を破棄する（ステップＳ７１）。これにより、ロードされていたライブラリファイル１２２または１３２が占有するメモリが解放される。さらに、アプリケーション配備部１０１は、ライブラリ管理ディレクトリ１４０から、当該固有ＩＤに対応するライブライファイル１２２または１３２を削除する（ステップＳ７２）。そして、まだ処理していない固有ＩＤがあれば、アプリケーション配備部１０１は、未処理の固有ＩＤに対して、ステップＳ６１に戻る（ステップＳ６３）。
【０１２６】
具体的には、アプリケーション１２０または１３０の一方の配備を解除する際、図９に示す固有ＩＤテーブル１０６におけるＣ１の項のエントリと、Ｃ１の固有ＩＤに対応するライブラリファイル１２２（１３２）およびクラス管理領域１０８とは、削除されない。

そして、固有ＩＤテーブル１０６におけるＣ２またはＣ３の項のどちらかのエントリと、Ｃ２またはＣ３の固有ＩＤに対応するライブラリファイル１２２または１３２、およびクラス管理領域１０８のそれぞれ一方とが、削除される。

【０１２７】

パッケージ名対応テーブル１２７または１３７に含まれるすべての固有ＩＤの処理の後（ステップＳ６３のＮＯ）、アプリケーション配備部１０１は、配備を解除されるアプリケーション１２０または１３０に対応するアプリケーションクラスローダ１２４または１３４、およびパッケージ名対応テーブル１２７または１３７を削除する（ステップＳ６４）。

【０１２８】

このようにして、アプリケーション１２０または１３０の配備が解除される。アプリケーション配備部１０１は、アプリケーションに対する配備の解除の際、固有ＩＤテーブル１０６に基づいて、ライブラリファイル１２２および１３２に関連するリソースの残留と削除を制御することができる。

【０１２９】

なお、アプリケーション配備部１０１は、図６に示す一連のアプリケーション配備解除動作の前後を含む任意のタイミングにおいて、一般的なＪａｖａのアプリケーションサーバにおいて実行されるアプリケーションの配備解除の動作を行ってもよい。

【０１３０】

以上が、アプリケーションの配備解除動作である。

【０１３１】

動作の説明の最後として、Ｊａｖａにおける一般的なクラスローダにおける動作と問題点の詳細について説明する。

【０１３２】

Ｊａｖａアプリケーションサーバにおいては、一般に、複数のクラスローダが、階層構造を形成している。クラスをロードする際には、以下のような委譲モデルが用いられる。

【０１３３】

委譲モデルにおいては、ロード対象とするライブラリファイルが、クラスローダごとに定義（関連付け）される。そして、アプリケーションから、あるクラスローダに対してクラスの要求が発行されたとき、そのクラスローダは、要求されたクラスがロード済みであれば対応するクラス（オブジェクト）を返す。

【０１３４】

しかし、要求されたクラスがロードされていなければ、そのクラスローダは、上位のクラスローダに対してロードを委譲する。上位のクラスローダも、同様の動作を再帰的に行う。階層の最上位のクラスローダにおいても、要求されたクラスがロード済みでない場合、最上位のクラスローダは、そのクラスが定義されたライブラリファイルに含まれているならば、そのクラスをロードする。そして、最上位のクラスローダは、ロードしたクラスを、１階層下の子クラスローダに返す。要求されたクラスが定義されたライブラリファイルに含まれない場合、最上位のクラスローダは、ロードできなかったことを示す応答を子クラスローダに返す。以下、子クラスローダにおいても、同様の動作を再帰的に行う。

【０１３５】

同名のクラスを含むライブラリが複数ある場合、このようにして、階層構造における上位のクラスローダのロード対象のライブラリファイルに含まれるクラスが優先的にロードされる。

【０１３６】

また、クラスローダの階層構造の末端部においては、アプリケーションのクラスローダが、アプリケーションサーバに配備された複数のアプリケーションごとに生成される。アプリケーションのクラスローダは、アプリケーションモジュールに含まれるプログラムおよびライブラリファイルをロード対象とする。アプリケーションごとのクラスローダは、アプリケーションごとにクラスの名前空間を分けることを可能とする。例えば、アプリケ

ーションＡのライブラリファイルとアプリケーションＢのライブラリファイルとに同じ名前で異なる内容のクラスが存在する場合にも、２つのアプリケーションのクラスローダが、それぞれ同じ名前のクラスを別々にロードすることができる。

【０１３７】

この一般的なクラスローダの構造には、「背景技術」欄において記述したような問題がある。問題の１つは、下位のクラスローダのロード対象のライブラリファイルに含まれる同名で異なる内容のクラスがロードできないことである。すなわち、一般的なクラスローダは、ある場合にはアプリケーションごとのライブラリファイルからロードし、その他の場合には、上位のクラスローダのロード対象であるシステムのライブラリファイルからロードするという切り替えができない。これは、常に上位のクラスローダのロード対象のライブラリファイルが優先されることが原因である。もう１つの問題は、各アプリケーションのクラスローダが、まったく同じ内容のライブラリファイルを、アプリケーションごとにロードしてしまうことである。その結果、メモリ使用量が増大する。これは、アプリケーションのクラスローダが、ロードするライブラリファイルを統一的に管理する仕組みがないからである。

【０１３８】

本実施形態は、このような一般的なクラスローダの構造における問題を解決することができる。すなわち、本実施形態におけるクラスロード部１１０は、アプリケーション１２０（１３０）に含まれるライブラリファイル１２２（１３２）を優先的にロードすることができる。すなわち、本実施形態に係る情報処理装置１００は、アプリケーション１２０（１３０）にライブラリファイル１２２（１３２）を含めるかどうかによって、優先的にロードするライブラリファイルを切り替えることができる。同時に、クラスロード部１１０は、同じ内容のライブラリファイルを多重にロードすることを回避できる。

【０１３９】

以上が、Ｊａｖａにおける一般的なクラスローダにおける動作と問題点の詳細である。

【０１４０】

以上、説明したように、本実施形態には、上述した第１の実施形態と同様の効果に加えて、さらに、アプリケーションの配備および配備の解除において、ライブラリファイルとメモリ等のリソースを適切に管理できるという効果もある。

【０１４１】

その理由は、アプリケーション配備部１０１が、固有ＩＤテーブル１０６に基づいて、固有ＩＤごとのライブラリファイルを参照しているアプリケーションの数を管理するからである。

【０１４２】

（第２の実施形態の変形例）

なお、本実施形態の変形例としては以下のようなものが考えられる。

【０１４３】

例えば、上述したアプリケーション配備部１０１の機能の一部は、他の機能部が実行してもよい。例えば、Ｊａｖａの一般的な慣習に合わせて、クラスロード部１１０が、記憶装置１０５に格納される情報類を以下のように管理してもよい。すなわち、アプリケーション配備部１０１は、固有ＩＤが生成された（図４のステップＳ１１）後、生成された固有ＩＤとライブラリファイル１２２（１３２）のパスを、共通クラスローダ１１３に通知する。そして、共通クラスローダ１１３が、固有ＩＤ管理テーブル１０６およびクラス管理領域１０８の生成を含む管理と、ライブラリ管理ディレクトリ１４０へのライブラリファイル１２２（１３２）のコピーとを行ってもよい。同様に、アプリケーションの配備の解除においても、共通クラスローダ１１３が、固有ＩＤ管理テーブル１０６、クラス管理領域１０８、および、ライブラリ管理ディレクトリ１４０に対する処理を行ってもよい。

【０１４４】

＜ハードウェア構成例の説明＞

なお、上述した各実施形態において図１乃至図３に示した各部は、それぞれ独立したハ

ードウェア回路で構成されていてもよいし、ソフトウェアプログラムの機能（処理）単位（ソフトウェアモジュール）と捕らえることができる。ただし、これらの図面に示した各部の区分けは、説明の便宜上の構成であり、実装に際しては、様々な構成が想定され得る。このような場合のハードウェア環境の一例を、図１０を参照して説明する。

【０１４５】

図１０は、本発明の各実施形態、および、その変形例に係る情報処理装置またはアプリケーションサーバに適用可能なコンピュータ（情報処理装置）の構成を例示する図である。すなわち、図１０は、上述した各実施形態における情報処理装置１および１００の少なくとも何れかを実現可能なコンピュータの構成であって、上述した各実施形態における各機能を実現可能なハードウェア環境を示す。

【０１４６】

図１０に示したコンピュータ９００は、ＣＰＵ（Ｃｅｎｔｒａｌ Ｐｒｏｃｅｓｓｉｎｇ Ｕｎｉｔ）９０１、ＲＯＭ（Ｒｅａｄ Ｏｎｌｙ Ｍｅｍｏｒｙ）９０２、ＲＡＭ（Ｒａｎｄｏｍ Ａｃｃｅｓｓ Ｍｅｍｏｒｙ）９０３、通信インタフェース（Ｉ／Ｆ）９０４、ディスプレイ９０５、およびハードディスク装置（ＨＤＤ）９０６を備え、これらがバス９０７を介して接続された構成を有する。なお、図１０に示したコンピュータが情報処理装置１および１００として機能する場合、ディスプレイ９０５は常時設けられる必要はない。

【０１４７】

また、通信インタフェース９０４は、上述した各実施形態において、当該各コンピュータ間における通信を実現する一般的な通信手段である。ハードディスク装置９０６には、プログラム群９０６Ａと、各種の記憶情報９０６Ｂとが格納されている。プログラム群９０６Ａは、例えば、上述した図１乃至図３に示した各ブロック（各部）に対応する機能を実現するためのコンピュータ・プログラムである。各種の記憶情報９０６Ｂは、例えば、図１乃至図３に示したロード要求対応情報７、ロード状態情報８、ライブラリ１０、１２２、および１３２、固有ＩＤテーブル１０６、クラス管理領域１０８、および、パッケージ名対応テーブル１２７および１３７などである。このようなハードウェア構成において、ＣＰＵ９０１は、コンピュータ９００の全体の動作を司る。

【０１４８】

そして、上述した各実施形態を例に説明した本発明は、各実施形態の説明において参照したブロック構成図（図１乃至図３）あるいはフローチャート（図４乃至図６）の機能を実現可能なコンピュータ・プログラムを供給した後、そのコンピュータ・プログラムを、当該ハードウェアのＣＰＵ９０１に読み出して実行することによって達成される。また、このコンピュータ内に供給されたコンピュータ・プログラムは、読み書き可能な一時記憶メモリ９０３またはハードディスク装置９０６などの不揮発性の記憶デバイス（記憶媒体）に格納すれば良い。

【０１４９】

また、前記の場合において、当該各装置内へのコンピュータ・プログラムの供給方法は、フロッピーディスク（登録商標）やＣＤ－ＲＯＭ等の各種記録媒体を介して当該装置内にインストールする方法や、インターネット等の通信ネットワーク１０００を介して外部よりダウンロードする方法等のように、現在では一般的な手順を採用することができる。そして、このような場合において、本発明は、係るコンピュータ・プログラムを構成するコード、或いは係るコードが記録されたところの、コンピュータ読み取り可能な記憶媒体によって構成されると捉えることができる。

【０１５０】

なお、上述した実施形態の一部または全部は、以下の付記のようにも記載されうるが、以下の付記に限定されるものではない。

【０１５１】

（付記１）

ライブラリファイルごとに、前記ライブラリファイルの内容を識別する識別子情報を生

成し、前記ライブラリファイルに含まれるロード要求の対象となりうる部分を表す要求対象情報と前記識別子情報との対応関係を表すロード要求対応情報を生成し、前記ロード要求対応情報を記憶装置に保存する識別子生成手段と、
実行されているアプリケーションからロード要求を受け付け、前記ロード要求、および前記ロード要求対応情報に基づいて、前記ロード要求の対象を含む前記ライブラリファイルの識別子情報を求めるロード要求解釈手段と、
前記ライブラリファイルに含まれる各部のロード状態を表すロード状態情報を管理し、前記ロード状態情報に基づいて、前記ロード要求の対象を表す要求対象情報および前記求めた識別子情報が示すロードを要求された部分がロードされていないと判断した場合に、前記求めた識別子情報が示す前記ライブラリファイルから、少なくとも前記ロードを要求された部分をロードし、前記ロード要求に対する応答を返すロード手段と
を備える情報処理装置。

【０１５２】

（付記２）
前記ロード手段は、前記ロード状態情報に基づいて、前記ロードを要求された部分がロードされていると判断した場合に、前記ロード要求に対して前記ロードを要求された部分をロードせずに、前記応答を返す
付記１記載の情報処理装置。

【０１５３】

（付記３）
前記アプリケーションを実行可能な状態にする配備を行う際、前記識別子情報と、前記識別子情報に対応する前記ライブラリファイルを参照するアプリケーションの数を表す参照数情報とを含むライブラリ識別子管理情報を生成する、アプリケーション配備手段を、
さらに備える付記１または２記載の情報処理装置。

【０１５４】

（付記４）
前記アプリケーション配備手段は、前記アプリケーションの前記実行可能な状態を解除する配備解除の際、前記ライブラリ識別子管理情報に基づいて、配備解除の結果、参照している前記アプリケーションが無くなる前記ライブラリファイルと、配備を解除される前記アプリケーションとに関する記憶情報を含むリソースを削除する
付記１乃至３のいずれか１つに記載の情報処理装置。

【０１５５】

（付記５）
前記識別子生成手段は、前記ライブラリファイルに含まれるパッケージ名を前記要求対象情報として前記ロード要求対応情報を生成し、
前記ロード要求解釈手段は、前記ロード要求の対象であるクラスのパッケージ名と、前記ロード要求対応情報に基づいて、前記ロード要求の対象を含む前記ライブラリファイルの識別子情報を求める
付記１乃至４のいずれか１つに記載の情報処理装置。

【０１５６】

（付記６）
前記識別子生成手段は、前記ライブラリファイルのハッシュ情報を求め、前記ハッシュ情報に基づいて前記識別義情報を生成する
付記１乃至５のいずれか１つに記載の情報処理装置。

【０１５７】

（付記７）
ライブラリファイルごとに、前記ライブラリファイルの内容を識別する識別子情報を生成し、
前記ライブラリファイルに含まれるロード要求の対象となりうる部分を表す要求対象情報と前記識別子情報との対応関係を表すロード要求対応情報を生成し、

前記ロード要求対応情報を記憶装置に保存し、
実行されているアプリケーションからロード要求を受け付けた際、
前記ロード要求、および前記ロード要求対応情報に基づいて、前記ロード要求の対象を含む前記ライブラリファイルの識別子情報を求め、
前記ライブラリファイルに含まれる各部のロード状態を表すロード状態情報に基づいて、前記ロード要求の対象を表す要求対象情報、および前記求めた識別子情報が示すロードを要求された部分がロードされていないと判断した場合に、前記求めた識別子情報が示す前記ライブラリファイルから、少なくとも前記ロードを要求された部分をロードし、さらに、前記ロード状態情報にロードしたことを登録し、
前記ロード要求に対する応答を返す
ライブラリロード方法。

【０１５８】

（付記８）
前記ロード要求を受け付けた際、前記ロードを要求された部分がロードされていないと判断した場合に、
前記ロード要求に対して前記ロードを要求された部分をロードせずに、前記応答を返す
付記７記載のライブラリロード方法。

【０１５９】

（付記９）
前記識別子情報を生成するより後、かつ、前記ロード要求を受け付ける前に、
前記識別子情報と、前記識別子情報に対応する前記ライブラリファイルを参照する参照するアプリケーションの数を表す参照数情報とを含むライブラリ識別子管理情報を生成し、
前記アプリケーションの実行可能な状態を解除する配備解除の際、
前記ライブラリ識別子管理情報に基づいて、前記配備解除の結果、参照している前記アプリケーションが無くなる前記ライブラリファイルと、前記配備解除をされる前記アプリケーションとに関する記憶情報を含むリソースを削除する
付記７または８記載のライブラリロード方法。

【０１６０】

（付記１０）
前記ロード要求対応情報を生成する際に、前記ライブラリファイルに含まれるパッケージ名を前記要求対象情報とし、
前記ロード要求の対象を含む前記ライブラリファイルの識別子情報を求める際に、前記ロード要求の対象であるクラスのパッケージ名と、前記ロード要求対応情報に基づいて、前記ロード要求の対象を含む前記ライブラリファイルの識別子情報を求める
付記７乃至９のいずれか１つに記載のライブラリロード方法。

【０１６１】

（付記１１）
前記識別子情報を生成する前に、前記ライブラリファイルのハッシュ情報を求め、
前記ハッシュ情報に基づいて前記識別子情報を生成する
付記７乃至１０のいずれか１つに記載のライブラリロード方法。

【０１６２】

（付記１２）
ライブラリファイルごとに、前記ライブラリファイルの内容を識別する識別子情報を生成し、前記ライブラリファイルに含まれるロード要求の対象となりうる部分を表す要求対象情報と前記識別子情報との対応関係を表すロード要求対応情報を生成し、前記ロード要求対応情報を記憶装置に保存する識別子生成処理と、
実行されているアプリケーションからロード要求を受け付け、前記ロード要求、および前記ロード要求対応情報に基づいて、前記ロード要求の対象を含む前記ライブラリファイルの識別子情報を求めるロード要求解釈処理と、

前記ライブラリファイルに含まれる各部のロード状態を表すロード状態情報を管理し、前記ロード状態情報に基づいて、前記ロード要求の対応を表す要求対象情報および前記求めた識別子情報が示すロードを要求された部分がロードされていないと判断した場合に、前記求めた識別子情報が示す前記ライブラリファイルから、少なくとも前記ロードを要求された部分をロードし、前記ロード要求に対する応答を返すロード処理と
をコンピュータに実行させるコンピュータ・プログラム。

【０１６３】

（付記１３）

前記ロード処理は、前記ロード状態情報に基づいて、前記ロードを要求された部分がロードされていると判断した場合に、前記ロード要求に対して前記ロードを要求された部分をロードせずに、前記応答を返す
付記１２記載のコンピュータ・プログラム。

【０１６４】

（付記１４）

前記識別子生成処理より後に、
前記識別子情報と、前記識別子情報に対応する前記ライブラリファイルを参照するアプリケーションの数を表す参照数情報とを含むライブラリ識別子管理情報を生成するアプリケーション配備処理を、
前記アプリケーションの実行可能な状態を解除する配備解除の際に、
前記ライブラリ識別子管理情報に基づいて、配備解除の結果、参照している前記アプリケーションが無くなる前記ライブラリファイルと、配備を解除される前記アプリケーションとに関する記憶情報を含むリソースを削除するアプリケーション配備解除処理を、
さらに実行する付記１２または１３記載のコンピュータ・プログラム。

【０１６５】

（付記１５）

前記識別子生成処理は、前記ライブラリファイルに含まれるパッケージ名を前記要求対象情報として前記ロード要求対応情報を生成し、
前記ロード要求解釈処理は、前記ロード要求の対象であるクラスのパッケージ名と、前記ロード要求対応情報に基づいて、前記ロード要求の対象を含む前記ライブラリファイルの識別子情報を求める
付記１２乃至１４のいずれか１つに記載のコンピュータ・プログラム。

【０１６６】

（付記１６）

前記識別子生成処理は、前記ライブラリファイルのハッシュ情報を求め、前記ハッシュ情報に基づいて前記識別義情報を生成する
付記１２乃至１５のいずれか１つに記載のコンピュータ・プログラム。

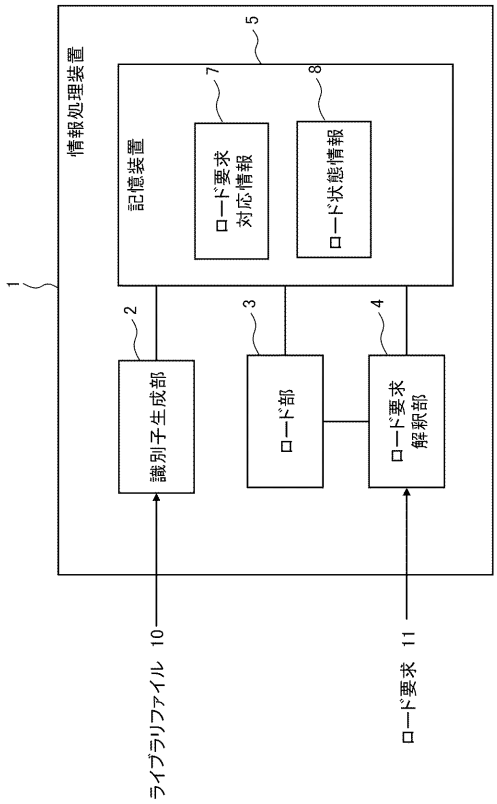
【符号の説明】

【０１６７】

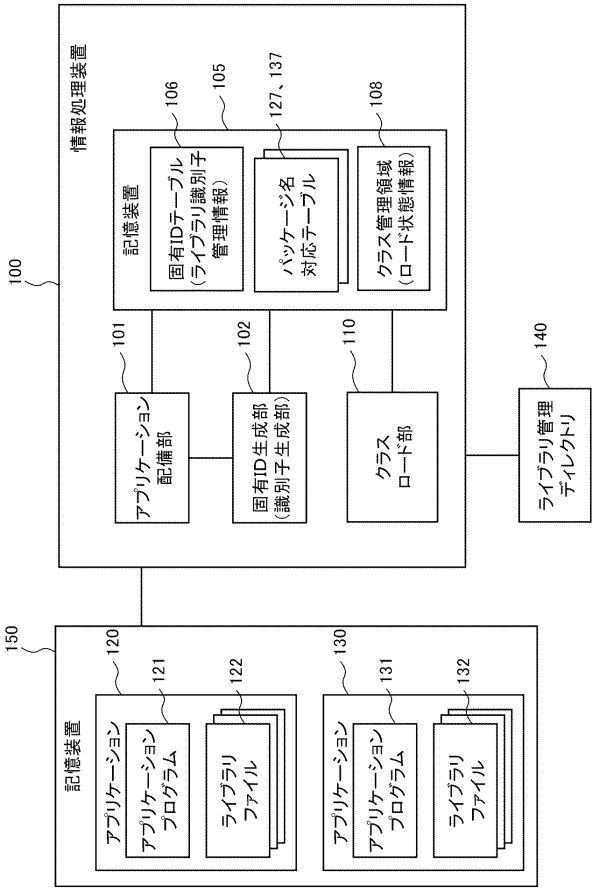
１、１００　情報処理装置
２　識別子生成部
３　ロード部
４　ロード要求解釈部
５、１０５、１５０　記憶装置
７　ロード要求対応情報
８　ロード状態情報
１０、１２２、１３２　ライブラリファイル
１１　ロード要求
１０１　アプリケーション配備部
１０２　固有ＩＤ生成部（識別子生成部）
１０６　固有ＩＤテーブル（ライブラリ識別子管理情報）

１０８　　クラス管理領域（ロード状態情報）
１１０　　クラスロード部
１１１　　上位クラスローダ群
１１３　　共通クラスローダ（ロード部）
１２０、１３０　　アプリケーション
１２１、１３１　　アプリケーションプログラム
１２４、１３４　　アプリケーションクラスローダ（ロード要求解釈部）
１２７、１３７　　パッケージ名対応テーブル
１４０　　ライブラリ管理ディレクトリ
９００　　情報処理装置（コンピュータ）
９０１　　ＣＰＵ
９０２　　ＲＯＭ
９０３　　ＲＡＭ
９０４　　通信インタフェース（Ｉ／Ｆ）
９０５　　ディスプレイ
９０６　　ハードディスク装置（ＨＤＤ）
９０６Ａ　　プログラム群
９０６Ｂ　　各種の記憶情報
９０７　　バス
１０００　　ネットワーク（通信ネットワーク）

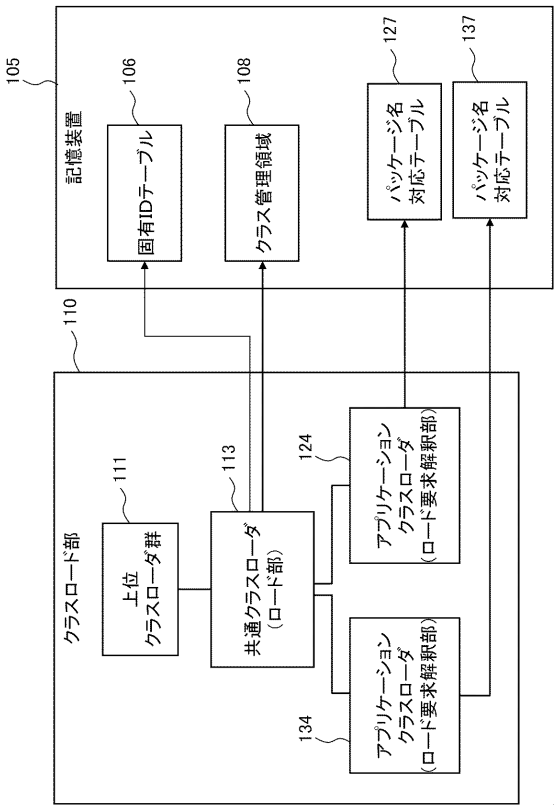
【図１】



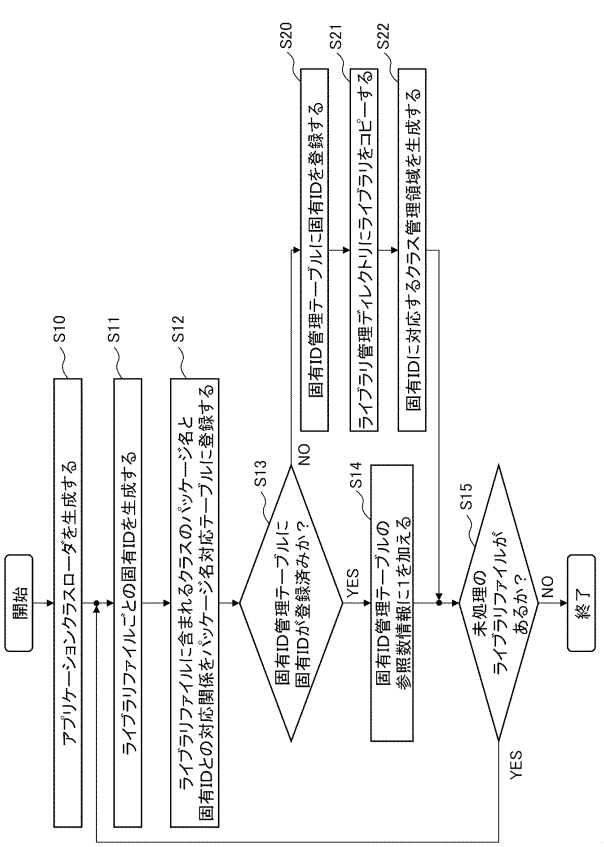
【図２】



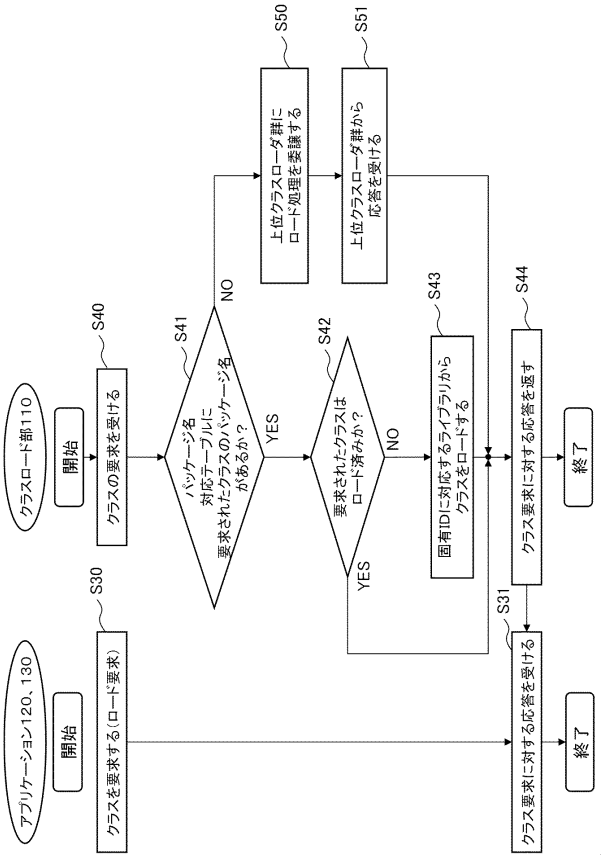
【図３】


105
記憶装置
106
固有IDテーブル
108
クラス管理領域
127
パッケージ名
対応テーブル
137
パッケージ名
対応テーブル
110
クラスロード部
111
上位
クラスローダ群
113
共通クラスローダ
（ロード部）
124
アプリケーション
クラスローダ
（ロード要求解釈部）
134
アプリケーション
クラスローダ
（ロード要求解釈部）

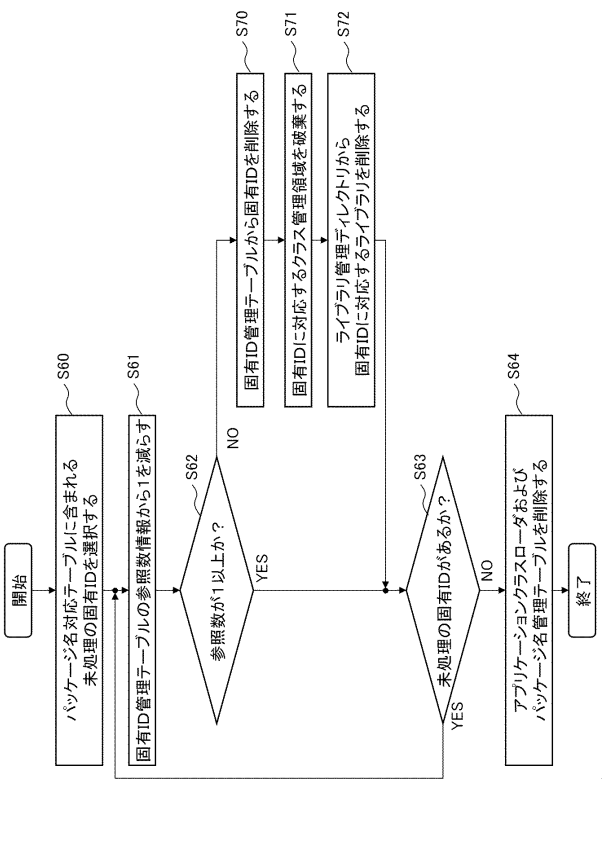
【図４】


開始
S10
アプリケーションクラスローダを生成する
S11
ライブラリファイルごとの固有IDを生成する
S12
ライブラリファイルに含まれるクラスのパッケージ名と
固有IDとの対応関係をパッケージ名対応テーブルに登録する
S13
固有ID管理テーブルに
固有IDが登録済みか？
NO
S20
固有ID管理テーブルに固有IDを登録する
S21
ライブラリ管理ディレクトリにライブラリをコピーする
S22
固有IDに対応するクラス管理領域を生成する
YES
S14
固有ID管理テーブルの
参照数情報に1を加える
S15
未処理の
ライブラリファイルが
あるか？
YES
NO
終了

【図５】


アプリケーション120、130
開始
S30
クラスを要求する（ロード要求）
クラスロード部110
開始
S40
クラスの要求を受ける
S41
パッケージ名
対応テーブルに
要求されたクラスのパッケージ名
があるか？
NO
YES
S50
上位クラスローダ群に
ロード処理を委譲する
S51
上位クラスローダ群から
応答を受ける
S42
要求されたクラスは
ロード済みか？
NO
YES
S43
固有IDに対応するライブラリから
クラスをロードする
S44
クラス要求に対する応答を返す
終了
S31
クラス要求に対する応答を受ける
終了

【図６】


開始
S60
パッケージ名対応テーブルに含まれる
未処理の固有IDを選択する
S61
固有ID管理テーブルの参照数情報から1を減らす
S62
参照数が1以上か？
NO
YES
S70
固有ID管理テーブルから固有IDを削除する
S71
固有IDに対応するクラス管理領域を破棄する
S72
ライブラリ管理ディレクトリから
固有IDに対応するライブラリを削除する
S63
未処理の固有IDがあるか？
NO
YES
S64
アプリケーションクラスローダおよび
パッケージ名管理テーブルを削除する
終了

【図７】

127

項番	パッケージ名	固有ID
A1	org.apache.commons.beanutils.converters	0c41fee1b50b22ac23ac99388f649b9a634e8346
A2	org.apache.commons.beanutils.expression	0c41fee1b50b22ac23ac99388f649b9a634e8346
A3	org.apache.commons.beanutils.locale.converters	0c41fee1b50b22ac23ac99388f649b9a634e8346
A4	org.apache.commons.beanutils.locale	0c41fee1b50b22ac23ac99388f649b9a634e8346
A5	org.apache.commons.logging	f6f66e966c70a83ffbdb6f17a0919eaf7c8aca7f
A6	org.apache.commons.logging.impl	f6f66e966c70a83ffbdb6f17a0919eaf7c8aca7f

【図８】

137

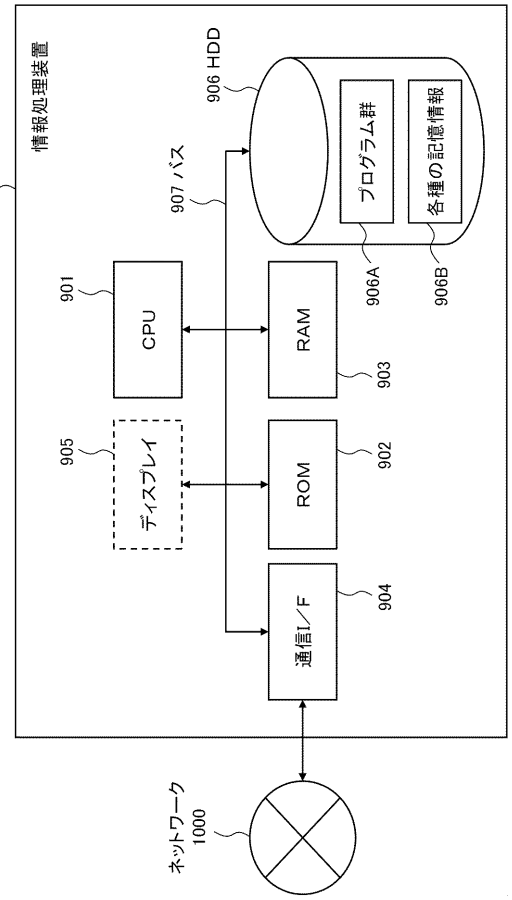
項番	パッケージ名	固有ID
B1	org.apache.commons.beanutils.converters	0c41fee1b50b22ac23ac99388f649b9a634e8346
B2	org.apache.commons.beanutils.expression	0c41fee1b50b22ac23ac99388f649b9a634e8346
B3	org.apache.commons.beanutils.locale.converters	0c41fee1b50b22ac23ac99388f649b9a634e8346
B4	org.apache.commons.beanutils.locale	0c41fee1b50b22ac23ac99388f649b9a634e8346
B5	org.apache.commons.logging	37c659e57293656ebef1a247fc6ceb738ebdfc74
B6	org.apache.commons.logging.impl	37c659e57293656ebef1a247fc6ceb738ebdfc74

【図９】

106

項番	固有ID	参照数
C1	0c41fee1b50b22ac23ac99388f649b9a634e8346	2
C2	f6f66e966c70a83ffbdb6f17a0919eaf7c8aca7f	1
C3	37c659e57293656ebef1a247fc6ceb738ebdfc74	1

【図１０】



フロントページの続き

(56)参考文献 特開２００１－１５４８３１（ＪＰ，Ａ）
特開２０１２－１４１９７３（ＪＰ，Ａ）
米国特許出願公開第２００８／０２４３９６５（ＵＳ，Ａ１）
特開２０１０－１１３４７４（ＪＰ，Ａ）
特開２００７－２０６９６５（ＪＰ，Ａ）
特開２０１３－１９６４５３（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　８／００－　８／７７
９／４４－　９／４５１