(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4913302号
(P4913302)

(45)発行日　平成24年4月11日(2012.4.11)　(24)登録日　平成24年1月27日(2012.1.27)

(51)Int.Cl.		F I		
G06F 11/36	(2006.01)	G06F	9/06	620M
G06F 9/44	(2006.01)	G06F	9/44	530P
G06F 11/28	(2006.01)	G06F	11/28	A

請求項の数 18　(全 25 頁)

(21)出願番号	特願2001-542302 (P2001-542302)
(86)(22)出願日	平成12年11月10日 (2000.11.10)
(65)公表番号	特表2003-515857 (P2003-515857A)
(43)公表日	平成15年5月7日 (2003.5.7)
(86)国際出願番号	PCT/US2000/042092
(87)国際公開番号	WO2001/040900
(87)国際公開日	平成13年6月7日 (2001.6.7)
審査請求日	平成19年10月2日 (2007.10.2)
(31)優先権主張番号	09/439,645
(32)優先日	平成11年11月12日 (1999.11.12)
(33)優先権主張国	米国(US)

前置審査

(73)特許権者　597004720
オラクル・アメリカ・インコーポレイテッド
アメリカ合衆国、９４０６５　カリフォルニア州、レッドウッド・ショアーズ、オラクル・パークウェイ、５００
(74)代理人　110001195
特許業務法人深見特許事務所
(72)発明者　チェン　ジクン
アメリカ合衆国　カリフォルニア州　９４３０３－４９００　パロ　アルト　エムエス　パル　０１－５２１　サン　アントニオ　ロード　９０１

審査官　林　毅

最終頁に続く

(54)【発明の名称】言語サブセットの妥当性検査

(57)【特許請求の範囲】

【請求項１】

言語サブセットの妥当性検査方法であって、

プロセッサ上でメモリに記憶された命令を実行する段階を備え、上記実行する段階は、言語サブセットの妥当性検査を含み、

上記妥当性検査は、上記プロセッサ上で命令を実行することによって、複数のオブジェクト指向プログラムモジュールを含むコンピュータ実行可能プログラムのスタティックな検査を行う段階を含み、

上記複数のオブジェクト指向プログラムモジュールは、クラスファイルを含み、

上記クラスファイルは、バイトコードを有するメソッド、及び、フィールドを含み、

上記バイトコードは、第１コンピュータ言語として定義され、

上記第１コンピュータ言語は、第２コンピュータ言語のハードウェア依存性サブセットであり、上記妥当性検査方法は、更に、

上記プロセッサ上で上記命令を実行することによって、複数のオブジェクト指向プログラムモジュールを含む上記コンピュータ実行可能プログラムのカードベースの妥当性検査を実行する段階を備え、

上記第１コンピュータ言語は、第１実行環境において実行されるものであり、

上記第２コンピュータ言語は、第２実行環境において実行されるものであり、上記第２実行環境は、上記第１実行環境とは異なるものであり、

上記スタティックな検査を行う前記段階は、

上記クラスファイルの１つを受け取り、
上記クラスファイルの上記１つの中のフィールドを読み出し、
上記フィールドのフィールド宣言が上記第１コンピュータ言語によってサポートされていないとき、上記フィールドに対してエラー状態である旨を表示し、
上記第１コンピュータ言語の上記第１実行環境によってサポートされていない上記フィールド宣言において使用される各任意の特徴に対してエラー状態である旨を表示し、
上記クラスファイルの上記１つにおいてメソッドを受け取り、
上記メソッドのメソッド宣言が上記第１コンピュータ言語によってサポートされていない任意の特徴を含むとき、上記メソッドに対してエラー状態である旨を表示し、そして、
上記メソッド宣言が、上記第１コンピュータ言語の上記第１実行環境にサポートされていない任意の特徴を含まないとき、上記メソッドのメソッド宣言以降に記述されたバイトコードを検査するという段階を含み、
上記カードベースの妥当性検査を実行する上記段階は、更に、
全ての上記プログラムモジュールに対するコード及びデータの使用量を加算することによりメモリ使用量を決定する段階、及び
上記メモリ使用量が最大メモリ使用量を越えたときにエラー状態である旨を表示する段階を含む、方法。

【請求項２】
上記バイトコードを検査する上記段階は、更に、
上記バイトコードを読み取り、
上記バイトコードが、上記第１コンピュータ言語によりサポートされていないデータ形式を使用するときにエラー状態である旨を表示し、
上記バイトコードが、上記第１コンピュータ言語によりサポートされていないデータ形式の値に基づいて動作するときにエラー状態である旨を表示し、
上記第１実行環境が上記第１コンピュータ言語の任意の特徴をサポートせずそして上記命令が上記任意の特徴を使用するときにエラー状態である旨を表示し、そして
上記第１実行環境が上記第１コンピュータ言語の任意の特徴をサポートせずそして上記命令が上記任意の特徴の少なくとも１つの値に基づいて動作するときにエラー状態である旨を表示するという段階を含む、請求項１に記載の方法。

【請求項３】
上記妥当性検査方法は、更に、上記プロセッサ上で上記命令を実行することによって、複数のオブジェクト指向プログラムモジュールを含む上記コンピュータ実行可能プログラムのメソッドベースの妥当性検査を実行する段階を備え、上記メソッドベースの妥当性検査を実行する上記段階は、更に、
上記クラスファイルの１つを受け取り、
上記クラスファイルの上記１つの内のメソッドに対するデータ流を発生し、そして
上記データ流を使用してローカル変数の形式を決定するとともに、決定したローカル変数の形式に基づいて上記メソッド内の演算中間結果にオーバーフローの可能性が存在するかどうか決定するという段階を含み、このオーバーフローの可能性は、上記メソッドの第２の最終結果とは意義的に異なる上記メソッドの第１の最終結果を形成し、この第１の最終結果は、上記メソッドが上記第１実行環境において上記第１コンピュータ言語に対して実行されるときに得られ、そして上記第２の最終結果は、上記メソッドが上記第２実行環境において上記第２コンピュータ言語に対して実行されるときに得られる、請求項１に記載の方法。

【請求項４】
上記妥当性検査方法は、更に、
上記プロセッサ上で命令を実行することによって、複数のオブジェクト指向プログラムモジュールを含むコンピュータ実行可能プログラムのメソッドのパッケージベースの妥当性検査を実行する段階を備え、上記パッケージベースの妥当性検査を実行する上記段階は

、
クラスファイルを含むパッケージを受け取り、
上記パッケージにおけるメソッドの妥当性検査を行う段階を含み、上記パッケージにおけるメソッドの妥当性検査を行う上記段階は、
上記パッケージにおける上記クラスファイルの１つにおいてメソッドを読み出し、
上記メソッド内のローカル変数及びパラメータの全数が変数及びパラメータの最大数を越えるときにエラー状態である旨を表示し、
上記メソッド内のローカル変数及びパラメータの上記全数が変数及びパラメータの上記最大数を越えないときに上記メソッド内の全てのバイトコードを上記第１コンピュータ言語に基づいて変換して、変換されたバイトコードを形成し、
変換されたバイトコードの全数がバイトコードの最大数を越えたときにエラー状態である旨を表示するという段階を含む、請求項１に記載の方法。
【請求項５】
上記パッケージベースの妥当性検査は、更に、パッケージの制限の妥当性検査を行う段階を含み、前記パッケージの制限の妥当性検査を行う上記段階は、ＣＡＰファイルコンポーネントサイズの妥当性検査を行う段階を含み、前記ＣＡＰファイルコンポーネントサイズの妥当性検査を行う上記段階は、
上記ＣＡＰファイルのメソッドコンポーネントによって使用されるバイトの全数を決定し、そして
上記バイトの全数が最大メモリ使用量を越えたときにエラー状態である旨を表示する段階を含む、請求項４に記載の方法。
【請求項６】
パッケージベースの妥当性検査を実行する上記段階は、更に、
上記パッケージ中のクラスの妥当性検査を行う段階を含み、上記パッケージ中のクラスの妥当性検査を行う前記段階は、
上記パッケージ内の上記クラスファイルの少なくとも１つを読み出し、
上記少なくとも１つのクラスファイルのクラス内のインスタンスフィールドの全数がインスタンスフィールドの最大数を越えたときにエラー状態である旨を表示し、
上記少なくとも１つのクラスファイルのクラス内のスタティックフィールドの全数がスタティックフィールドの最大数を越えたときにエラー状態である旨を表示し、
上記少なくとも１つのクラスファイルのインスタンスメソッドを含む上記パッケージ内のインスタンスメソッドの全数が上記インスタンスメソッドの最大数を越えたときにエラー状態である旨を表示し、
上記パッケージ内のスタティックメソッドの全数がスタティックメソッドの最大数を越えたときにエラー状態である旨を表示し、そして
上記パッケージの各クラス又は各インターフェイスのスーパーインターフェイスの全数がスーパーインターフェイスの最大数を越えたときにエラー状態である旨を表示するという段階を含む、請求項４に記載の方法。
【請求項７】
クラスの妥当性検査を行う上記段階は、更に、アクセスの妥当性検査を行う段階を含み、アクセスの妥当性検査を行う前記段階は、
パブリックインターフェイスが上記パッケージ内のパッケージ可視インターフェイスを拡張するときにエラー状態である旨を表示し、そして
スーパークラスにおけるパッケージデフォールトメソッドのアクセス可視性が上記パッケージ内のサブセットにおいてパブリック又は保護状態に変更されるときにエラー状態である旨を表示するという段階を含む、請求項６に記載の方法。
【請求項８】
上記第１コンピュータ言語は、Ｊａｖａ　Ｃａｒｄ（登録商標）バーチャルマシンに対して定義され、
上記第２コンピュータ言語は、Ｊａｖａ（登録商標）言語よりなる、請求項１に記載の

方法。
【請求項９】
上記第１コンピュータ言語は、Ｊａｖａ　Ｃａｒｄ（登録商標）バーチャルマシンに対して定義され、
上記第２コンピュータ言語は、Ｊａｖａ（登録商標）クラスファイルよりなる、請求項１に記載の方法。
【請求項１０】
コンピュータに言語サブセットの妥当性検査方法を実行させるためのプログラムを記憶した記憶装置であって、前記プログラムは、前記コンピュータに、
複数のオブジェクト指向プログラムモジュールを含むコンピュータ実行可能プログラムのスタティックな検査を行うステップを実行させ、
上記複数のオブジェクト指向プログラムモジュールは、クラスファイルを含み、
上記クラスファイルは、バイトコードを有するメソッド、及び、フィールドを含み、
上記バイトコードは、第１コンピュータ言語として定義され、
上記第１コンピュータ言語は、第２コンピュータ言語のハードウェア依存性サブセットであり、前記プログラムは、前記コンピュータに更に、
複数のオブジェクト指向プログラムモジュールを含む上記コンピュータ実行可能プログラムのカードベースの妥当性検査を実行するステップを実行させ、
上記第１コンピュータ言語は、第１実行環境において実行されるものであり、
上記第２コンピュータ言語は、第２実行環境において実行されるものであり、上記第２実行環境は上記第１実行環境とは異なるものであり、
上記のスタティックな検査を実行する前記ステップは、
上記クラスファイルの１つを受け取り、
上記クラスファイルの上記１つの中のフィールドを読み出し、
上記フィールドのフィールド宣言が上記第１コンピュータ言語によってサポートされていないとき上記フィールドに対してエラー状態である旨を表示し、
上記第１コンピュータ言語の上記第１実行環境によってサポートされていないフィールド宣言において使用される各任意の特徴に対してエラー状態である旨を表示し、そして
上記クラスファイルの上記１つにおいてメソッドを受け取り、
上記メソッドのメソッド宣言が上記第１コンピュータ言語によってサポートされていない任意の特徴を含むとき、上記メソッドに対してエラー状態である旨を表示し、そして、
上記メソッド宣言が、上記第１コンピュータ言語の上記第１実行環境にサポートされていない任意の特徴を含まないとき、上記メソッドのメソッド宣言以降に記述されたバイトコードを検査するというステップを含み、
上記カードベースの妥当性検査を実行する上記ステップは、更に、
全ての上記プログラムモジュールに対するコード及びデータの使用量を加算することによりメモリ使用量を決定するステップ、及び
上記メモリ使用量が最大メモリ使用量を越えたときにエラー状態である旨を表示するステップを含む、プログラム記憶装置。
【請求項１１】
上記バイトコードを検査する前記ステップは、更に、
上記バイトコードを読み取り、
上記バイトコードが、上記第１コンピュータ言語によりサポートされていないデータ形式を使用するときにエラー状態である旨を表示し、
上記バイトコードが、上記第１コンピュータ言語によりサポートされていないデータ形式の値に基づいて動作するときにエラー状態である旨を表示し、
上記第１実行環境が上記第１コンピュータ言語の任意の特徴をサポートせずそして上記命令が上記任意の特徴を使用するときにエラー状態である旨を表示し、そして
上記第１実行環境が上記第１コンピュータ言語の任意の特徴をサポートせずそして上記

命令が上記任意の特徴の少なくとも１つの値に基づいて動作するときにエラー状態である旨を表示するというステップを含む、請求項１０に記載のプログラム記憶装置。

【請求項１２】

前記プログラムは、前記コンピュータに更に、複数のオブジェクト指向プログラムモジュールを含む上記コンピュータ実行可能プログラムのメソッドベースの妥当性検査を実行するステップを実行させ、上記メソッドベースの妥当性検査を実行する前記ステップは、更に、

上記クラスファイルの１つを受け取り、

上記クラスファイルの上記１つの内のメソッドに対するデータ流を発生し、そして

上記データ流を使用してローカル変数の形式を決定するとともに、決定したローカル変数の形式に基づいて上記メソッド内の演算中間結果にオーバーフローの可能性が存在するかどうか決定するというステップを含み、このオーバーフローの可能性は、上記メソッドの第２の最終結果とは意義的に異なる第１の最終結果を形成し、この第１の最終結果は、上記メソッドが上記第１実行環境において上記第１コンピュータ言語に対して実行されるときに得られ、そして上記第２の最終結果は、上記メソッドが上記第２実行環境において上記第２コンピュータ言語に対して実行されるときに得られる、請求項１０に記載のプログラム記憶装置。

【請求項１３】

前記プログラムは、前記コンピュータに更に、複数のオブジェクト指向プログラムモジュールを含むコンピュータ実行可能プログラムのメソッドのパッケージベースの妥当性検査を実行するステップを実行させ、上記パッケージベースの妥当性検査を実行する前記ステップは、

クラスファイルを含むパッケージを受け取り、

上記パッケージにおけるメソッドの妥当性検査を行うステップを含み、上記パッケージにおけるメソッドの妥当性検査を行う上記ステップは、

上記パッケージにおける上記クラスファイルの１つにおいてメソッドを読み出し、

上記メソッド内のローカル変数及びパラメータの全数が変数及びパラメータの最大数を越えるときにエラー状態である旨を表示し、

上記メソッド内のローカル変数及びパラメータの上記全数が変数及びパラメータの上記最大数を越えないときに上記メソッド内の全てのバイトコードを上記第１コンピュータ言語に基づいて変換して、変換されたバイトコードを形成し、

変換されたバイトコードの全数がバイトコードの最大数を越えたときにエラー状態である旨を表示するというステップを含む、請求項１０に記載のプログラム記憶装置。

【請求項１４】

上記パッケージベースの妥当性検査は、更に、パッケージの制限の妥当性検査を行う段階を含み、前記パッケージの制限の妥当性検査を行う上記ステップは、ＣＡＰファイルコンポーネントサイズの妥当性検査を行うステップを含み、前記ＣＡＰファイルコンポーネントサイズの妥当性検査を行う上記ステップは、

上記ＣＡＰファイルのメソッドコンポーネントによって使用されるバイトの全数を決定し、そして

上記バイトの全数が最大メモリ使用量を越えたときにエラー状態である旨を表示するステップを含む、請求項１３に記載のプログラム記憶装置。

【請求項１５】

パッケージベースの妥当性検査を実行する上記ステップは、更に、

上記パッケージ中のクラスの妥当性検査を行うステップを含み、上記パッケージ中のクラスの妥当性検査を行う前記ステップは、

上記パッケージ内の上記クラスファイルの少なくとも１つを読み出し、

上記少なくとも１つのクラスファイルのクラス内のインスタンスフィールドの全数がインスタンスフィールドの最大数を越えたときにエラー状態である旨を表示し、

上記少なくとも１つのクラスファイルのクラス内のスタティックフィールドの全数が

スタティックフィールドの最大数を越えたときにエラー状態である旨を表示し、
上記少なくとも１つのクラスファイルのインスタンスメソッドを含む上記パッケージ内のインスタンスメソッドの全数が上記インスタンスメソッドの最大数を越えたときにエラー状態である旨を表示し、
上記パッケージ内のスタティックメソッドの全数がスタティックメソッドの最大数を越えたときにエラー状態である旨を表示し、そして
上記パッケージの各クラス又は各インターフェイスのスーパーインターフェイスの全数がスーパーインターフェイスの最大数を越えたときにエラー状態である旨を表示するというステップを含む、請求項１３に記載のプログラム記憶装置。
【請求項１６】
上記クラスの妥当性検査を行う上記ステップは、更に、アクセスの妥当性検査を行うステップを含み、アクセスの妥当性検査を行う前記ステップは、
パブリックインターフェイスが上記パッケージ内のパッケージ可視インターフェイスを拡張するときにエラー状態である旨を表示し、そして
スーパークラスにおけるパッケージデフォールトメソッドのアクセス可視性が上記パッケージ内のサブセットにおいてパブリック又は保護状態に変更されるときにエラー状態である旨を表示するというステップを含む、請求項１５に記載のプログラム記憶装置。
【請求項１７】
上記第１コンピュータ言語は、Ｊａｖａ　Ｃａｒｄ（登録商標）バーチャルマシンに対して定義され、
上記第２コンピュータ言語は、Ｊａｖａ（登録商標）言語よりなる請求項１０に記載のプログラム記憶装置。
【請求項１８】
上記第１コンピュータ言語は、Ｊａｖａ　Ｃａｒｄ（登録商標）バーチャルマシンに対して定義され、
上記第２コンピュータ言語は、Ｊａｖａ（登録商標）クラスファイルよりなる請求項１０に記載のプログラム記憶装置。
【発明の詳細な説明】
【０００１】
【関連出願に対するクロスレファレンス】
本出願は、次のものに関連している。
「OBJECT-ORIENTED INSTRUCTION SET FOR RESOURCE-CONSTRAINED DEVICES」と題する１９９９年２月２日出願のスサー及びシュワベ氏の米国特許出願；
「VIRTUAL MACHINE WITH SECURELY DISTRIBUTED BYTECODE VERIFICATION」と題する１９９７年４月１５日出願のレビー及びシュワベ氏の米国特許出願；及び
「OPTIMIZATION OF N-BASE TYPED ARITHMETIC EXPRESSIONS」と題する１９９９年１１月１２日出願のチェン及びシュワベ氏の米国特許出願。
【技術分野】
本発明は、コンピュータシステムに係る。より詳細には、本発明は、言語サブセットの妥当性検査に係る。
【０００２】
【背景技術】
Ｊａｖａ（登録商標）プラットホームによって使用されるようなオブジェクト指向のプログラミング技術が広範囲に利用されている。オブジェクト指向のプログラムの基本的なユニットは、ここでメンバーと称されるメソッド（手順）及びフィールド（データ）を有するオブジェクトである。メンバーを共用するオブジェクトは、クラスにグループ分けされる。クラスとは、そのクラスにおけるオブジェクトの共用メンバーを定義する。従って、各オブジェクトは、それが属するクラスの特定のインスタンスである。実際に、あるクラスは、同様の特徴をもつ多数のオブジェクト（多数のインスタンス）を形成するためのテンプレートとしてしばしば使用される。

【０００３】
クラスの１つの特性は、そのクラス内のメンバーの実際のインプレメンテーションが、外部のユーザ、及びインターフェイスにより露出されるもの以外の他のクラスから隠されるという特性を示すカプセル化である。これは、クラスを、例えば、ネットワークの異なるサイトの異なる開発者による分散型開発に適したものにする。必要とされるクラスを組み立て、それらを一緒にリンクし、そしてそれにより得られたプログラムを実行することにより完全なプログラムを形成することができる。
クラスは、継承という特性も有する。継承とは、１つのクラスが別のクラスの全てのメンバーを継承できるようにするメカニズムである。別のクラスから継承するクラスをサブクラスと称し、属性を与えるクラスはスーパークラスである。記号的には、これは、サブクラス←スーパークラス、又はスーパークラス→サブクラスと書き表すことができる。サブクラスは、付加的なメンバーを追加することによりスーパークラスの能力を拡張することができる。サブクラスは、同じ名称及び形式をもつ交換メソッドを与えることによりスーパークラスのバーチャルメソッドをオーバーライドすることができる。
【０００４】
クラス形式のメンバーは、フィールド及びメソッドであり、これらは、スーパークラスから継承されたメンバーを含む。クラスファイルも、スーパークラスを命名する。メンバーは、パブリックでもよく、これは、その宣言を含むクラスのメンバーによりこれにアクセスできることを意味する。メンバーは、プライベートであってもよい。クラスのプライベートフィールドは、そのクラス内に定義されたメソッドにおいてのみ見ることができる。同様に、プライベートメソッドは、クラス内のメソッドによってしか呼び出せない。プライベートメンバーは、サブクラス内では見ることができず、他のメンバーと同様にサブクラスによって継承されない。メンバーは、保護することもできる。
【０００５】
インターフェイスの形式は、そのメンバーが定数及び抽象的メソッドであるような形式である。この形式は、インプレメンテーションをもたないが、その他の非関連クラスは、その抽象的メソッドに対するインプレメンテーションを与えることによりそれを実施することができる。インターフェイスは、クラスがサブクラスを有するのと同様に、サブインターフェイスを有することができる。サブインターフェイスは、そのスーパーインターフェイスから継承し、そして新たなメソッド及び定数も定義することができる。更に、あるインターフェイスは、一度に２つ以上のインターフェイスを拡張することができる。２つ以上のインターフェイスを拡張するインターフェイスは、それらインターフェイスの各々から全ての抽象的メソッド及び定数を継承し、そしてそれ自身の付加的なメソッド及び定数を定義することができる。
【０００６】
Ｊａｖａ（登録商標）プログラミング言語では、クラスをグループ分けすることができると共に、グループに命名することができ、クラスの命名されたグループがパッケージとなる。クラスメンバーが、パブリック、プライベート又は保護されたキーワードのいずれでも宣言されない場合には、そのクラスメンバーは、それを定義するクラス内及び同じパッケージの一部分であるクラス内でしか見ることができない。保護されたメンバーは、クラスを宣言するメンバーによりアクセスすることもできるし、又はそれが宣言されたパッケージ内のどこからでもアクセスすることができる。Ｊａｖａ（登録商標）プログラミング言語は、１９９６年８月のアジソンーウェズリー・ロングマン・インク出版のゴースリング氏等の「The Java™ Language Specification」に詳細に説明されている。
【０００７】
バーチャルマシンは、プロセッサにより実行される命令のシーケンス又はソフトウェアアプリケーションにより発生される抽象的計算マシンである。「アーキテクチャー・ニュートラル」という語は、Ｊａｖａ（登録商標）プログラミング言語で書かれたようなプログラムであって、種々の異なるコンピュータアーキテクチャーを有する種々のコンピュータプラットホームにおいてバーチャルマシンにより実行できるプログラムを指す。従って、

例えば、Ｗｉｎｄｏｗｓ（登録商標）ベースのパーソナルコンピュータシステムで実施されるバーチャルマシンは、ＵＮＩＸ（登録商標）ベースのコンピュータシステムで実施されるバーチャルマシンと同じ命令セットを使用してアプリケーションを実行する。バーチャルマシンの命令シーケンスのプラットホーム独立コード化の結果として、１つ以上のバイトコードのストリームが生じ、その各々は、例えば、１バイト長さの数値コードである。

【０００８】

Ｊａｖａ（登録商標）バーチャルマシンは、バーチャルマシンの一例である。このＪａｖａ（登録商標）バーチャルマシンにより実行されるべきコンパイルされたコードは、クラスファイルフォーマットとして知られている通常ファイルに記憶されるハードウェア及びオペレーティングシステム独立の２進フォーマットを使用して表わされる。クラスファイルは、Ｊａｖａ（登録商標）プログラミング言語で書かれたプログラムを表わすことができるが他の多数のプログラミング言語もサポートできるオブジェクト指向の構造を取り扱うように設計される。クラスファイルフォーマットは、プラットホーム特有のオブジェクトファイルフォーマットにおいて許可されるとして取り上げられるバイト順序のような詳細を含むクラス又はインターフェイスの表示を正確に定義する。セキュリティのために、Ｊａｖａ（登録商標）バーチャルマシンは、クラスファイル内の命令に強力なフォーマット及び構造上の制約を課する。有効なクラスファイルに関して表現できる機能をもついかなる言語も、Ｊａｖａ（登録商標）バーチャルマシンをホストとすることができる。クラスファイルは、Ｊａｖａ（登録商標）プログラミング言語で書かれたプログラムを表わすことができるが多数の他のプログラム言語もサポートできるオブジェクト指向の構造を取り扱うように設計される。Ｊａｖａ（登録商標）バーチャルマシンは、１９９９年４月、アジソンーウェスリー・ロングマン・インク、第２版、リンドホルム氏等の「The Java™ Virtual Machine Specification」に詳細に説明されている。

【０００９】

リソースに制約のある装置とは、一般に、典型的なデスクトップコンピュータ等に比して、メモリ及び／又は計算能力又は速度に比較的制約がある装置であると考えられる。リソースに制約のある他の装置は、例えば、セルラー電話、境界走査装置、現場でプログラム可能な装置、パーソナルデジタルアシスタンス（ＰＤＡ）及びページャー、並びに他のミニアチュア又は小型フットプリント装置を含む。本発明は、リソースに制約のない装置にも使用できる。

説明上、「プロセッサ」という語は、物理的なコンピュータ又はバーチャルマシンを指すのに使用される。

【００１０】

インテリジェントポータブルデータ搬送カードとしても知られているスマートカードは、リソースに制約のある装置の一形式である。スマートカードは、プラスチック又は金属で作られ、そしてプログラムを実行するためのマイクロプロセッサ又はマイクロコントローラと、プログラム及びデータを記憶するためのメモリとが埋め込まれた電子チップを有している。ほぼクレジットカードのサイズであるこのような装置は、８ビット又は１６ビットアーキテクチャーのコンピュータチップを有する。更に、これら装置は、通常、メモリ容量が限定されている。例えば、あるスマートカードは、１キロバイト（１Ｋ）未満のランダムアクセスメモリ（ＲＡＭ）、制限のあるリードオンリメモリ（ＲＯＭ）、及び／又は電気的に消去可能なプログラマブルリードオンリメモリ（ＥＥＰＲＯＭ）のような不揮発性メモリを有する。

【００１１】

Ｊａｖａ（登録商標）バーチャルマシンは、Ｊａｖａ（登録商標）プログラミング言語で書かれたプログラムを実行し、そしてメモリが比較的豊富なデスクトップコンピュータにおいて使用するように設計される。スマートカードのようなリソースに制約のある装置で実行するためにＪａｖａ（登録商標）バーチャルマシンの完全インプレメンテーションを使用するプログラムを書くことが要望される。しかしながら、スマートカードのようなリ

ソースに制約のある装置の限定されたアーキテクチャー及びメモリのために、このような装置では完全なＪａｖａ（登録商標）バーチャルマシンプラットホームを実施することができない。従って、個別のＪａｖａ　Ｃａｒｄ（登録商標）（Ｊａｖａ（登録商標）プログラミング言語をサポートするスマートカード）技術が、リソースに制約のある装置に対してＪａｖａ（登録商標）プログラミング言語のサブセットをサポートする。

【００１２】

Ｊａｖａ（登録商標）技術でサポートされる幾つかのアイテムは、Ｊａｖａ　Ｃａｒｄ（登録商標）技術においてサポートされない。例えば、Ｊａｖａ（登録商標）技術は、形式チャー(char)、ダブル、フロート及びロングをサポートするが、Ｊａｖａ　Ｃａｒｄ（登録商標）技術は、それらをサポートしない。更に、あるＪａｖａ（登録商標）言語特徴は、限定された形態でサポートされ、Ｊａｖａ　Ｃａｒｄ（登録商標）技術は、これら特徴のオペレーションの範囲を、Ｊａｖａ（登録商標）技術の範囲未満に制限する。例えば、Ｊａｖａ（登録商標）技術は、二次元以上のアレーを許すが、Ｊａｖａ　Ｃａｒｄ（登録商標）技術は、一次元のアレーしか許さない。

【００１３】

Ｊａｖａ　Ｃａｒｄ（登録商標）技術において、Ｊａｖａ　Ｃａｒｄ（登録商標）コンバータは、通常のクラスファイルを入力として取り上げそしてそれらをＣＡＰ（変換されたアプレット）ファイルに変換する。ＣＡＰフォーマットは、クラスファイル情報のサブセットをサポートする。各ＣＡＰファイルは、１つのＪａｖａ（登録商標）パッケージに定義された全てのクラス及びインターフェイスを含む。ＣＡＰファイルは、コンパクトな最適化されたフォーマットを有し、従って、Ｊａｖａ（登録商標）パッケージを、リソースに制約のある装置に効率的に記憶しそして実行することができる。変換の後に、Ｊａｖａ　Ｃａｒｄ（登録商標）技術でイネーブルされる装置にＣＡＰファイルがインストールされる。

Ｊａｖａ　Ｃａｒｄ（登録商標）技術に対してＪａｖａ（登録商標）言語のサブセットが存在することは、Ｊａｖａ　Ｃａｒｄ（登録商標）サブセットによりサポートされないアイテムを含む有効なＪａｖａ（登録商標）プログラムモジュールを形成できることを意味する。リソースに制約のある装置においてこれらプログラムモジュールを実行すると、エラーのある結果を招く。従って、公知技術では、リソースに制約のある装置においてアプリケーションの正しい実行を容易にする言語サブセット妥当性検査方法及び装置が要望されている。

【００１４】

【発明の開示】

言語サブセット妥当性検査方法は、プログラムを構成する多数のプログラムモジュールを妥当性検査する段階を含む。各プログラムモジュールは、第１コンピュータ言語として定義された多数のバイトコードを含み、その第１コンピュータ言語は、第２コンピュータ言語のハードウェア依存性サブセットである。上記妥当性検査段階は、第１コンピュータ言語として定義されない多数のプログラムモジュール内の各アイテムに対してエラー状態である旨を表示し、第１コンピュータ言語の実行環境によりサポートされない多数のプログラムモジュール内の各アイテムに対してエラー状態である旨を表示し、そして第１コンピュータ言語として定義されるが、第１コンピュータ言語に一致しないやり方で使用される複数のプログラムモジュール内の各アイテムに対してエラー状態である旨を表示することを含む。言語サブセットの妥当性検査装置は、プログラム命令を有する少なくとも１つのメモリと、そのプログラム命令を使用して、プログラムを一緒に形成する多数のプログラムモジュールを妥当性検査するように構成された少なくとも１つのプロセッサとを備えている。このプロセッサは、更に、上記プログラム命令を使用して、第１コンピュータ言語として定義されない多数のプログラムモジュール内の各アイテムに対してエラー状態である旨を表示し、第１コンピュータ言語の実行環境によってサポートされない多数のプログラムモジュール内の各アイテムに対してエラー状態である旨を表示し、そして第１コンピュータ言語として定義されるが第１コンピュータ言語に一致しないやり方で使用される

多数のプログラムモジュール内の各アイテムに対してエラー状態である旨を表示するように構成される。

【００１５】

【発明を実施するための最良の形態】

当業者であれば、本発明の以下の説明は、例示に過ぎないことが明らかであろう。この開示の利益を得る当業者であれば、本発明の他の実施形態も明らかとなろう。

本発明は、コンピュータシステムに係る。より詳細には、本発明は、言語サブセットの妥当性検査に係る。更に、本発明は、（１）本発明のレイアウトパラメータ、及び／又は（２）本発明を使用してコンピュータにおけるオペレーションを遂行するためのプログラム命令が記憶されたマシン読み取り可能な媒体にも係る。このような媒体は、例えば、磁気テープ、磁気ディスク、光学的に読み取り可能な媒体、例えば、ＣＤ　ＲＯＭ、及び半導体メモリ、例えば、ＰＣＭＣＩＡカードを含む。又、媒体は、小型ディスク、ディスケット又はカセットのようなポータブル装置の形態もとり得る。又、媒体は、ハードディスクドライブ、又はコンピュータＲＡＭのような大型即ち不動装置の形態をとってもよい。

【００１６】

図１Ａ及び１Ｂは、高レベル言語ソース及びクラスファイルレベルの両方においてサポートされた言語と言語サブセット特徴との間の関係を示す。図１Ａを参照すれば、これは、言語ソースレベルにおける言語と言語サブセットとの間の関係を示すブロック図である。Ｊａｖａ　Ｃａｒｄ（登録商標）技術は、参照番号１００で示すように、Ｊａｖａ（登録商標）言語のサブセットをサポートする。又、Ｊａｖａ　Ｃａｒｄ（登録商標）技術は、参照番号１０２で示すように、幾つかのサポートされたＪａｖａ（登録商標）言語特徴に対してオペレーションの範囲を制限する。更に、あるＪａｖａ（登録商標）特徴は、特定のＪａｖａ　Ｃａｒｄ（登録商標）バーチャルマシンにより任意にサポートされる。これら任意の特徴は、参照番号１０４で示されている。Ｊａｖａ（登録商標）技術によってサポートされるが、Ｊａｖａ　Ｃａｒｄ（登録商標）技術によってサポートされない特徴は、参照番号１０６で示されている。

【００１７】

図１Ｂを参照すれば、これは、クラスファイルレベルにおいて言語と言語サブセットとの間の関係を示すブロック図である。Ｊａｖａ　Ｃａｒｄ（登録商標）技術は、参照番号１１０で示すように、Ｊａｖａ（登録商標）クラスファイルに含まれた情報のサブセットをサポートする。又、Ｊａｖａ　Ｃａｒｄ（登録商標）技術は、幾つかのサポートされたＪａｖａ（登録商標）特徴に対するオペレーションの範囲を制限し、これらの限界は、参照番号１１２で示すように、クラスファイルレベルにおいてチェックすることができる。更に、特定のＪａｖａ　Ｃａｒｄ（登録商標）バーチャルマシンにより任意にサポートされるＪａｖａ（登録商標）特徴も、クラスファイルレベルにおいてチェックすることができる。これら任意の特徴は、参照番号１１４によって表わされる。Ｊａｖａ（登録商標）技術によりサポートされるが、Ｊａｖａ　Ｃａｒｄ（登録商標）技術によりサポートされない特徴も、クラスファイルレベルにおいてチェックされ、参照番号１１６で表わされる。従って、図１Ｂは、ソースレベルにおいてサポートされる言語特徴と、クラスファイルレベルにおいてサポートされる言語特徴との間に直接的な関係があることを示している。更に、クラスファイルレベルにおいてチェックすることにより、高レベル言語ソースレベルで書かれたプログラムが、Ｊａｖａ　Ｃａｒｄ（登録商標）によって定義されない特徴を使用したかどうか決定することができる。

【００１８】

本発明によれば、言語サブセットを使用するシステムにおける命令レベルプログラムファイルは、言語サブセットに対して妥当性検査される。サポートされないアイテムが検出され、そして制限された形態でサポートされたアイテムは、それらが適切に使用されるかどうか決定するためにチェックされる。更に、特定の実行環境がチェックされて、それが任意の特徴をサポートするかどうか決定する。実行環境が任意の特徴をサポートする場合には、命令レベルプログラムファイルがチェックされ、任意の特徴が適切に使用されるかど

うか決定される。実行環境が任意の特徴をサポートしない場合には、命令レベルプログラムファイルがチェックされ、サポートされない任意の特徴が使用されるかどうか決定される。プロセスは、多数の段階で進行し、比較的完全な妥当性検査を与える。

【００１９】

この説明全体を通して、本発明は、Ｊａｖａ（登録商標）技術及びＪａｖａ　Ｃａｒｄ（登録商標）技術に関して説明する。しかしながら、当業者であれば、本発明は、他のプラットホームにも適用できることが明らかであろう。更に、当業者であれば、本発明は、言語サブセットに対する異なる形式の妥当性検査にも適用できることが明らかであろう。

図２を参照すれば、本発明の１つの実施形態に基づき言語サブセットの妥当性検査を遂行する方法が示されている。参照番号１３０において、スタティックな妥当性検査が実行される。説明上、「スタティックな妥当性検査」とは、他のプログラムユニットやプログラム実行状態を参照せずにクラスファイルのみを検査することにより実行される妥当性検査を指す。スタティックな妥当性検査では、サポートされない特徴が検出される。更に、任意の特徴が妥当性検査される。参照番号１３５では、メソッドをベースとする妥当性検査が実行される。メソッドをベースとする妥当性検査では、メソッドの命令に基づいてデータ流分析が実行され、演算オペレーションの中間値がチェックされる。この段階を使用して、考えられるオーバーフロー状態が検出される。参照番号１４０では、パッケージをベースとする妥当性検査が実行される。パッケージをベースとする妥当性検査では、限定された範囲をもつサポートされた特徴が検出される。参照番号１４５では、カードをベースとする妥当性検査が実行される。カードをベースとする妥当性検査では、全てのプログラムにより使用されるメモリの合計量がチェックされて、プログラムがターゲット装置のメモリ制約を越えるかどうか決定される。

【００２０】

図３を参照すれば、これは、本発明の１つの実施形態に基づきスタティックな妥当性検査を実行するところを示すフローチャートである。参照番号１６０において、クラスファイルが受け取られる。参照番号１６５において、フィールドに対しスタティックな妥当性検査が実行される。参照番号１７０において、メソッドに対しスタティックな妥当性検査が実行される。参照番号１７５において、妥当性検査されるべき別のクラスファイルが残っているかどうか決定するためのチェックが行われる。別のクラスファイルがある場合には、参照番号１６０において実行が続けられる。このプロセスは、全てのクラスファイルが妥当性検査されるまで続けられる。

【００２１】

図４を参照すれば、これは、本発明の１つの実施形態に基づきメソッドに対しスタティックな妥当性検査を遂行するところを示すフローチャートである。参照番号１９０において、メソッドが受け取られる。参照番号１９２において、メソッドの宣言が言語サブセットによりサポートされるかどうか決定するためのチェックが行われる。Ｊａｖａ　Ｃａｒｄ（登録商標）技術では、パラメータ形式、返送形式及びアクセスフラグをチェックして、それらが妥当かどうか決定される。例えば、キャラクターストリング、ロング、ダブル及びフロート形式は、Ｊａｖａ（登録商標）技術によりサポートされるが、これら形式は、Ｊａｖａ　Ｃａｒｄ（登録商標）ではサポートされない。更に、Ｊａｖａ（登録商標）技術は、「同期された」及び「ネーティブな」アクセスフラグをサポートするが、Ｊａｖａ　Ｃａｒｄ（登録商標）技術は、サポートしない。形式又はアクセスフラグがサポートされない場合には、参照番号１９４においてエラー状態である旨が表示される。

【００２２】

メソッドの宣言がサポートされる場合には、参照番号１９６において、特定の実行環境が任意の特徴をサポートするかどうか決定するためのチェックがなされる。Ｊａｖａ　Ｃａｒｄ（登録商標）技術では、形式ｉｎｔが任意の特徴の一例である。特定の実行環境（Ｊａｖａ　Ｃａｒｄ（登録商標）バーチャルマシン）は、その任意の特徴がサポートされるかどうか決定する。実行環境が任意の特徴をサポートしない場合には、参照番号１９８において、メソッドの宣言が任意の特徴を含むかどうか決定するためのチェックが行われ

る。メソッドの宣言が任意の特徴を含む場合には、参照番号１９４においてエラー状態である旨が表示される。参照番号２００において、メソッドのバイトコードが妥当性検査される。参照番号２０２において、妥当性検査されるべき別のメソッドが残されているかどうかチェックされる。別のメソッドがあれば、参照番号１９０において実行が続けられる。このプロセスは、クラスファイルの全てのメソッドが妥当性検査されるまで続けられる。

【００２３】

図４Ｂを参照すれば、これは、本発明の１つの実施形態に基づきメソッドのバイトコードを妥当性検査するところを示すフローチャートである。参照番号２１０において、バイトコードが受け取られる。参照番号２１２において、バイトコードが言語サブセットによりサポートされるかどうかについて決定がなされる。バイトコードは、サポートされないデータ形式が使用されるかどうかそしてサポートされないデータ形式の値に対するオペレーションが使用されるかどうか決定するためにチェックされる。又、バイトコードは、サポートされない特徴が使用されるかどうか決定するためにチェックされる。例えば、Ｊａｖａ　Ｃａｒｄ（登録商標）技術では、スレッドはサポートされない。キーワード「同期された」は、スレッドを同期するためにモニタを使用することを指示し、このキーワードは、特定のバイトコード表示を有する。バイトコードにおけるその存在は、モニタの使用を指示する。

【００２４】

バイトコードがサポートされないか、又はこれを使用して、サポートされない形式のデータに対してオペレーションする場合には、参照番号２１４においてエラー状態である旨が表示される。バイトコードがサポートされる場合には、参照番号２１６において、特定の実行環境が任意の特徴をサポートするかどうか決定するためのチェックがなされる。実行環境が任意の特徴をサポートしない場合には、参照番号２１８において、そのチェックされたバイトコードが任意の特徴を表わすかどうか決定するためのチェックがなされる。そのチェックされたバイトコードが任意の特徴を表わす場合には、参照番号２１４においてエラー状態である旨が表示される。参照番号２２０において、更なるバイトコードがメソッドに残っているかどうか決定するためのチェックがなされる。更なるバイトコードが残っている場合には、参照番号２１０において実行が続けられる。このプロセスは、メソッドの全バイトコードがチェックされるまで続けられる。

【００２５】

図５を参照すれば、これは、本発明の１つの実施形態に基づきフィールドに対するスタティックな妥当性検査を遂行するところを示すフローチャートである。フィールドに対するスタティックな妥当性検査は、メソッドに対するスタティックな妥当性検査と同様に実行される。Ｊａｖａ　Ｃａｒｄ（登録商標）技術では、無効のアクセスフラグ又はフィールドは、例えば、「揮発性」及び「過渡」を含む。フィールドの宣言が、サポートされない形式及びアクセスフラグを使用する場合には、エラー状態である旨が表示される。フィールドの宣言が、実行環境によってサポートされないＪａｖａ　Ｃａｒｄ（登録商標）技術の形式ｉｎｔのような任意の特徴を使用する場合にも、エラー状態である旨が表示される。

【００２６】

Ｊａｖａ（登録商標）バーチャルマシン命令セットは、積分形式バイト、ショート及びｉｎｔの値を取り扱うように命令セットを定義する。形式バイト及びショートの変数は、コンパイル中に積分形式ｉｎｔへと広げられる。それ故、計算された値も、３２ビットのｉｎｔ値である。しかしながら、Ｊａｖａ　Ｃａｒｄ（登録商標）バーチャルマシンは、積分形式ｉｎｔの変数を取り扱うための命令セットに加えて、形式バイト及びショートの変数を取り扱うように個別の命令セットを定義する。

【００２７】

Ｊａｖａ　Ｃａｒｄ（登録商標）プラットホームに対して３２ビットのｉｎｔ形式をサポートすることは、任意である。３２ビットｉｎｔ形式をサポートしないターゲットプラ

ットホームは、３２ビットｉｎｔ形式の変数を使用することができない。又、Ｊａｖａ（登録商標）クラスファイルに使用される３２ビットの演算バイトコードは、Ｊａｖａ Ｃａｒｄ（登録商標）命令セットに使用される１６ビット命令に変換されねばならない。従って、１６ビット命令によって計算された値は、１６ビットｉｎｔ値である。この変換は、１６ビット表示を越えて拡張するオーバーフローの可能性を形成する。オーバーフローの可能性をもつ値が、オーバーフローに敏感な命令へ供給された場合、又はそれがアレーインデックスとして使用された場合には、エラー性の結果が生じる。オーバーフローの可能性が存在するかどうかの決定は、データ流を発生することを含む。オーバーフローの可能性が存在するときを決定するこのプロセスの詳細は、１９９９年１１月１５日に出願された「OPTIMIZATION OF N-BASE TYPED ARITHMETIC EXPRESSIONS」と題するチェン及びシュワベ氏の米国特許出願に開示されており、本発明の混乱を防止するためにここでは詳細に説明しない。

【００２８】

図６を参照すれば、これは、本発明の１つの実施形態に基づきメソッドベースの妥当性検査を遂行するところを示すフローチャートである。メソッドベースの妥当性検査では、潜在的なオーバーフロー状態が存在するかどうか決定するためにメソッドが分析される。参照番号２７０において、メソッドが受け取られる。参照番号２７５において、潜在的なオーバーフロー状態に対してメソッドがチェックされる。

参照番号２８５において、ローカル変数形式が妥当性検査される。Ｊａｖａ Ｃａｒｄ（登録商標）技術では、コンパイルディレクティブは、コンパイラーがコード発生プロセスの一部分としてローカル変数属性テーブルを発生するかどうか決定する。ローカル変数属性テーブルは、ローカル変数に対する形式情報を含む。ローカル変数属性テーブルが存在する場合には、有効形式に対する妥当性検査ローカル変数が、上記「スタティックな妥当性検査」段階で行われる。ローカル変数属性テーブルが存在しない場合には、データ流がこの段階で使用されて、ローカル変数の形式を決定する。ローカル変数が、サポートされない形式を有するか、又はターゲット装置によってサポートされない任意の形式を有する場合には、エラー状態である旨が表示される。

【００２９】

参照番号２９０において、妥当性検査されるべき別のメソッドが残っているかどうか決定するためにチェックがなされる。別のメソッドが残っている場合には、参照番号２７０において実行が続けられる。このプロセスは、クラスファイルの全てのメソッドが妥当性検査されるまで続けられる。

図７を参照すれば、これは、本発明の１つの実施形態に基づきパッケージベースの妥当性検査を遂行するところを示すフローチャートである。パッケージベースの妥当性検査では、制限された範囲をもつサポートされた特徴が検出される。参照番号３００において、パッケージを構成するクラスファイルが受け取られる。参照番号３０５において、パッケージ内のメソッドが妥当性検査される。参照番号３１０において、パッケージ内のクラスが妥当性検査される。参照番号３１５において、パッケージの制限が妥当性検査される。

【００３０】

図８を参照すれば、これは、本発明の１つの実施形態に基づきパッケージにおけるメソッドを妥当性検査するところを示すフローチャートである。参照番号３３０において、メソッドが受け取られる。参照番号３３５において、変数及びパラメータの全数が所定数を越えるかどうか決定するためにチェックがなされる。Ｊａｖａ Ｃａｒｄ（登録商標）技術では、変数及びパラメータの全数が２５５を越えることがない。全数が所定数を越える場合には、参照番号３５０においてエラー状態である旨が表示される。参照番号３３７において、ＪａｖａバイトコードがＪａｖａ Ｃａｒｄ（登録商標）バイトコードに変換される。参照番号３４０において、メソッド内のＪａｖａ Ｃａｒｄ（登録商標）バイトコードの全数が所定数を越えるかどうか決定するためのチェックがなされる。Ｊａｖａ Ｃａｒｄ（登録商標）技術では、Ｊａｖａ Ｃａｒｄ（登録商標）バイトコードの全数が３２，７６７を越えることがない。バイトコードの数がこの限界を越える場合には、参照番

号３５５においてエラー状態である旨が表示される。参照番号３６５において、パッケージの全てのメソッドが妥当性検査されたかどうか決定するためのチェックがなされる。別のメソッドが残っている場合には、参照番号３３０において実行が続けられる。このプロセスは、全てのメソッドが妥当性検査されるまで続けられる。

【００３１】

図９Ａを参照すれば、これは、本発明の１つの実施形態に基づきパッケージにおけるクラス及びインターフェイスを妥当性検査するところを示すフローチャートである。参照番号３８０において、クラスファイルが受け取られる。参照番号３８２において、クラス内のインスタンスフィールドの数が妥当性検査される。Ｊａｖａ　Ｃａｒｄ（登録商標）技術では、クラス当りのインスタンスフィールドの全数が２５５を越えてはならない。この数は、クラスにおいて宣言されたフィールドと、クラスの各スーパークラスにおいて宣言された全インスタンスフィールドとを含む。典型的に、スーパークラス内のフィールドに関する情報は、サブクラスに対するクラスファイル内に含まれない。それ故、スーパークラスに対するクラスファイルを読み取って、スーパークラス内のインスタンスフィールドの数を決定しなければならない。このプロセスは、ルートスーパークラスが検査されるまで繰り返し形態で続けられる。このようにして得られたインスタンスフィールドの全数は、インスタンスフィールドの最大数と比較される。

【００３２】

参照番号３８４において、スタティックフィールドの数がチェックされる。Ｊａｖａ　Ｃａｒｄ（登録商標）技術では、クラス当りのパブリック及び保護されたスタティックフィールドの全数が２５６を越えることがない。インスタンスフィールドとは異なり、スタティックフィールドのインカーネーションは、１つしか存在しない。従って、スタティックフィールドに対する最大数は、クラスの各スーパークラスではスタティックフィールドに適用されない。

参照番号３８６において、インスタンスメソッドの数がチェックされる。Ｊａｖａ　Ｃａｒｄ（登録商標）技術では、パッケージ可視インスタンスメソッドの全数が１２８を越えてはならず、そしてパブリック及び保護されたインスタンスメソッドの全数が１２８を越えてはならない。これらの数は、クラスにおいて宣言されたメソッドと、クラスの各スーパークラスにおいて宣言された全メソッドとを含む。典型的に、スーパークラス内のメソッドに関する情報は、サブクラスに対するクラスファイル内には含まれない。それ故、スーパークラスに対するクラスファイルを読み取って、スーパークラスにおけるインスタンスメソッドの数を決定しなければならない。パブリック及び保護されたインスタンスメソッドについては、このプロセスは、ルートスーパークラスが検査されるまで繰り返し形態で続けられる。パッケージ可視メソッドについては、この繰り返しは、チェックされたパッケージ内の全スーパークラスが検査されるまで続けられる。最後に、複写が除去される。

【００３３】

参照番号３８８において、スタティックなメソッドの数がチェックされる。Ｊａｖａ　Ｃａｒｄ（登録商標）技術では、パブリック及び保護されたスタティックなメソッドの全数が２５６を越えてはならない。インスタンスメソッドとは異なり、クラス内にスタティックメソッドのインカーネーションは、１つしか存在しない。従って、スタティックメソッドの最大数は、クラス形式の各スーパークラスにおいてスタティックなメソッドには適用されない。

参照番号３９０において、クラスアクセス制御が妥当性検査される。この妥当性検査は、パブリックインターフェイスがパッケージ可視インターフェイスを不適切に拡張するかどうか決定し、そしてスーパークラスにおけるパッケージデフォールトメソッドのアクセス可視性がサブクラスにおけるパブリック又は保護へと不適切に変更されたかどうか決定する。

【００３４】

参照番号３９２において、クラス又はインターフェイスが所定数より多いスーパーインタ

ーフェイスを有するかどうか決定するためのチェックが行われる。Ｊａｖａ　Ｃａｒｄ（登録商標）技術では、スーパーインターフェイスの最大数が１５である。クラスファイルがクラスを表わす場合には、このクラスのスーパーインターフェイスは、このクラスによって直接実施されるインターフェイスと、各々の直接実施されるインターフェイスのスーパーインターフェイスと、いずれかのスーパークラスによって実施されるインターフェイスとを含む。従って、スーパーインターフェイスの全数は、全てのインターフェイスがカウントされるまで各スーパーインターフェイス及びスーパークラスに繰り返し訪問することにより決定される。複写が除去された後に残っているインターフェイスの数は、実施されるインターフェイスの最大数と比較される。

【００３５】

クラスファイルがインターフェイスを表わす場合には、このチェックされたインターフェイスのスーパーインターフェイスは、このインターフェイスによって直接継承されるインターフェイスと、各直接的スーパーインターフェイスのスーパーインターフェイスとを含む。従って、スーパーインターフェイスの全数は、各インターフェイス及びそのスーパーインターフェイスがどのインターフェイスから継承されるかを繰り返し決定することにより決定される。複写が除去された後に残っているインターフェイスの数は、継承可能なインターフェイスの最大数と比較される。

参照番号３９６において、妥当性検査されるべきクラスファイルが更に残っているかどうかに関して決定がなされる。更にクラスが残っている場合には、参照番号３８０において実行が続けられる。このプロセスは、全てのクラスが妥当性検査されるまで続けられる。

【００３６】

図９Ｂを参照すれば、これは、本発明の１つの実施形態に基づきクラスのスーパーインターフェイスの数をカウントするところを示すブロック図である。３つのクラスが、C_1　４００、C_2　４０２及びC_3　４０４で表わされる。３つのインターフェイスは、I_1　４０６、I_2　４０８及びI_3　４１０で表わされる。クラスC_3及びインターフェイスI_3は、パッケージＢ４１４にあり、一方、他のメンバーは、パッケージＡ４１２にある。クラスC_1でスタートして、そのクラスにより直接実施されるインターフェイス及びそれら全てのスーパーインターフェイスの数は、そのクラスのスーパークラスにより実施されるインターフェイスの数に追加され、そして複写が除去される。ここで、C_1により実施された第１のインターフェイスが検査される。クラスC_1は、I_1を実施する。I_1は、I_2を継承する。従って、I_1及びI_2がカウントされる。I_2もいずれかのインターフェイスを継承する場合には、これらインターフェイスもカウントされる。次いで、C_1のスーパークラスが検査される。クラスC_1は、C_2から延びる。クラスC_2は、いずれのインターフェイスも実施せず、C_3を拡張する。クラスC_3は、（I_2、I_3）を実施する。従って、クラスC_2は、（I_2、I_3）も実施する。クラスC_1へ戻ると、クラスC_1は、インターフェイスI_1及びインターフェイスI_1のスーパーインターフェイスI_2と、そのスーパークラスC_2により実施されるインターフェイスを実施する。それ故、C_1は、（I_1、I_2）＋（I_2、I_3）を実施する。複写を除去した後に、C_1は、３つのインターフェイス（I_1、I_2）＋（I_2、I_3）＝（I_1、I_2、I_3）を実施する。この数は、Ｊａｖａ　Ｃａｒｄ（登録商標）技術に対する最大数より小さい。

【００３７】

図９Ｃを参照すれば、これは、本発明の１つの実施形態に基づきインターフェイスのスーパーインターフェイスの数をカウントするところを示すブロック図である。５つのインターフェイスが、I_1　４１６、I_2　４１８、I_3　４２０、I_4　４２２及びI_5　４２４により表わされる。インターフェイスI_4及びI_5は、パッケージＤ４２８に配置され、そして他のインターフェイスは、個別のパッケージＣ４２６に配置される。インターフェイスI_1は、I_2及びI_3から延び、従って、インターフェイスI_1は、スーパーインターフェイスI_2及びI_3を有する。更に、インターフェイスI_1は、I_2が継承するもの及びI_3が継承するものから継承する。図９Ｃに示すように、I_3は、I_2及びI_4から延び、I_4は、I_5から延び、そしてI_5は、I_2から延びる。従って、I_4は（I_2）＋（I_5）＝（I_2、

I_5）から継承する。I_3は、（I_4）＋（I_2、I_5）＋（I_2）＝（I_2、I_4、I_5）から継承する。従って、I_1は、（I_2）＋（I_3）＋（I_2、I_4、I_5）から継承する。複写が除去された後に、I_1は、４つのインターフェイス（I_2、I_3、I_4、I_5）から継承する。この数は、Ｊａｖａ　Ｃａｒｄ（登録商標）技術に対する最大数より小さい。

【００３８】

図１０Ａを参照すれば、これは、本発明の１つの実施形態に基づいてアクセス制御を妥当性検査するところを示すフローチャートである。クラスファイルは、クラス又はインターフェイスのいずれも表わすことができるので、検査されたクラスファイルがインターフェイスを含むかどうかが参照番号４３０においてチェックされる。クラスファイルがインターフェイスを表わす場合には、インターフェイスがパブリックであるかどうかが参照番号４３１においてチェックされる。Ｊａｖａ　Ｃａｒｄ（登録商標）技術では、パブリックインターフェイスは、パッケージ可視インターフェイスを拡張してはならない。インターフェイスは、少なくとも１つのスーパーインターフェイスを拡張できるので、この決定は、パッケージ内の各スーパーインターフェイスを検査して、チェーン内のいずれかのインターフェイスがパッケージ可視インターフェイスであるかどうか決定することを必要とする。

【００３９】

インターフェイスがパブリックである場合には、参照番号４３２において、現在インターフェイスがいずれかのインターフェイスを拡張するかどうか決定するためのチェックがなされる。現在インターフェイスがインターフェイスを直接継承する場合には、そのインターフェイスが参照番号４３４において受け取られる。参照番号４３６において、直接的なスーパーインターフェイスがパッケージデフォールト可視性を有するかどうかに関して決定がなされる。直接的なスーパーインターフェイスは、現在インターフェイスによって直ちに継承されるインターフェイスである。直接的なスーパーインターフェイスがパッケージデフォールト可視性を有する場合には、参照番号４３８においてエラー状態である旨が表示される。直接的なスーパーインターフェイスがパッケージデフォールト可視性をもたない場合には、参照番号４４０において、別のスーパーインターフェイスが残っているかどうか決定するためのチェックが行われる。別のスーパーインターフェイスがある場合には、参照番号４３４において実行が続けられる。

【００４０】

又、Ｊａｖａ　Ｃａｒｄ（登録商標）技術では、スーパークラスにおけるパッケージデフォールトメソッドのアクセス可視性がサブクラスにおけるパブリック及び保護状態へと変更されてはならない。これは、参照番号４４２から始めてチェックされる。現在クラスファイルがクラスを表わす場合には、参照番号４４２において、現在クラスがスーパークラスを有するかどうかに関して決定がなされる。現在クラスがスーパークラスを有する場合には、現在クラスのパブリック又は保護されたメソッドが参照番号４４４において受け取られる。参照番号４４６では、そのメソッドが、パッケージデフォールト可視性を有するスーパークラスにおいても定義されたかどうか決定するために、メソッドがチェックされる。参照番号４４８では、更にパブリック又は保護されたメソッドが現在クラスにあるかどうかに関する決定がなされる。更にあれば、参照番号４４４において実行が続けられる。このプロセスは、現在クラスにおける全てのパブリック及び保護されたメソッドが検査されるまで続けられる。

【００４１】

図１０Ｂを参照すれば、これは、本発明によりパッケージデフォールトメソッドがパブリック及び保護されたメソッドとされたかどうか決定するところを示すフローチャートである。参照番号４６０において、現在クラスが同じパッケージにスーパークラスを有するかどうか決定するためのチェックがなされる。現在クラスが同じパッケージにスーパークラスを有する場合には、参照番号４６２において、メソッドがスーパークラスにおいても定義されるがパッケージデフォールト可視性を有するかどうかに関する決定がなされる。対応するメソッドがスーパークラスにおいてパッケージデフォールト可視性を有する場合

には、参照番号４６４においてエラー状態である旨が表示される。対応するメソッドがスーパークラスにおいて定義されないか又はパッケージデフォールト可視性をもたない場合には、参照番号４６０でスタートして、現在クラスが現在クラスのスーパークラスにセットされ、そしてプロセスが繰り返される。メソッドをチェックするこのプロセスは、スーパークラスのルート又はパッケージの境界に到達するまで繰り返し形態で続けられる。

【００４２】

図１１を参照すれば、これは、本発明の１つの実施形態に基づきパッケージの限界を妥当性検査するところを示すフローチャートである。参照番号４７６において、パッケージデ定義されたクラス及びインターフェイスの数がチェックされる。Ｊａｖａ　Ｃａｒｄ（登録商標）技術では、パッケージにおけるクラス及びインターフェイスの最大数は２５５である。この数が最大値を越えた場合には、参照番号４７８においてエラー状態である旨が表示される。参照番号４８０において、このチェックされたパッケージによりインポートされるパッケージの数が所定数より大きいかどうかの決定がなされる。Ｊａｖａ　Ｃａｒｄ（登録商標）技術では、パッケージは、１２８より多数のパッケージをインポートすることができない。

【００４３】

参照番号４９０では、パッケージにおけるクラスファイルがＣＡＰファイルに変換された後に、ＣＡＰファイルコンポーネントサイズが妥当性検査される。Ｊａｖａ　Ｃａｒｄ（登録商標）技術では、ＣＡＰファイルは、１組のコンポーネントより成る。各コンポーネントは、定義されたＪａｖａ（登録商標）パッケージにおける１組のエレメント、又はＣＡＰファイルのアスペクトを記述する。全てのコンポーネントは、次の一般的フォーマットを有する。

```
Component {
u1 tag
u2 size
u1 info[]
}
```

【００４４】

「size」アイテムは、「info」アレーコンポーネントにおけるバイトの数を指示する。サイズコンポーネントによって表わされる最大サイズは、６４Ｋバイトである。「info」アレーの内容及びフォーマットは、コンポーネントの形式と共に変化する。コンポーネントの１つの形式は、メソッドコンポーネントである。メソッドコンポーネントは、次のような一般的フォーマットを有する。

```
 Method_component  {
     u1  tag
     u2  size
        .
        .
 method_info methods[]
 }
```

メソッドアレーは、多数のメソッドを含むことができ、そして各メソッドは、３２Ｋバイト未満である。しかしながら、全てのメソッドに使用されるバイトの合計数が６４Ｋ限界を越えることが考えられる。従って、各コンポーネント内のバイトの数を加算し、そしてその合計を最大サイズ（６４Ｋ）と比較する。合計がこの限界より大きい場合には、参照番号４９５においてエラー状態である旨が表示される。

【００４５】

Ｊａｖａ　Ｃａｒｄ技術では、「エクスポート」ファイルは、Ｊａｖａパッケージの全ての一般的にアクセスできる情報を含む２進ファイルである。本発明の別の実施形態によれば、パッケージベースの妥当性検査を遂行する段階において、スーパークラス又はスーパ

ーインターフェイスに対するチェックが、スーパークラス又はスーパーインターフェイスを定義するパッケージを表わす「エクスポート」ファイルをチェックすることにより実行される。このエクスポートファイルは、スーパークラス又はスーパーインターフェイスのクラスファイルに代わって使用される。しかしながら、エクスポートファイルは、一般にアクセスできる情報しか含まないので、パッケージデフォールトクラス及びインターフェイス、パッケージデフォールト及びプライベートメソッド並びにフィールドは含まれない。それ故、パッケージベースの妥当性検査における幾つかのチェックは、充分な情報が得られないためにカードベースの妥当性検査まで延期する必要がある。

【００４６】

カードベースの妥当性検査までチェックを延期する例として、スーパーインターフェイスの数を正確にカウントするときには、パッケージ及びパブリック可視インターフェイスの両方が含まれることを必要とする。エクスポートファイルは、一般にアクセスできる情報しか含まないので、パッケージ可視インターフェイスは含まれない。従って、パブリックスーパーインターフェイスの数が、パッケージベースの妥当性検査中に、最大量より少ないと決定された場合には、パッケージ可視インターフェイスに関する情報が得られるときに、パブリック及びパッケージ可視インターフェイスの数がカードベースの妥当性検査において後で繰り返されねばならない。しかしながら、パブリックスーパーインターフェイスの数のカウントが最大量より多い場合には、スーパーインターフェイスの最大量を既に越えているので、それ以上の評価は不必要となる。

【００４７】

カードベースの妥当性検査が必要となるまでチェックを延期する別のケースは、クラスにおけるインスタンスフィールドの数をカウントすることである。スーパークラスにおけるプライベート及びパッケージデフォールトインスタンスフィールドは、エクスポートファイルに得られないので、正確なカウントを得ることができない。従って、クラスにおけるインスタンスフィールドの数を、カードベースの妥当性検査において、全てのクラスの情報が得られるときにカウントし直さねばならない。

本発明の別の実施形態によれば、パッケージベースの妥当性検査において実行される妥当性検査は、以下に述べるカードベースの妥当性検査まで延期される。この場合に、カードに対する全てのクラスの情報が得られるときに妥当性検査が実行される。

【００４８】

図１２を参照すれば、これは、本発明の１つの実施形態に基づきカードベースの妥当性検査を遂行するところを示すフローチャートである。参照番号５００において、実行可能なランタイムイメージに必要なクラスファイル又はＣＡＰファイルが受け取られる。参照番号５０５において、実行可能なイメージ内の全てのプログラムユニットに対するメモリ使用量が加算される。参照番号５１０において、計算されたメモリ使用量が最大量と比較される。合計メモリ使用量が、実行可能なイメージに対して定義された最大値より大きい場合には、参照番号５１５においてエラー状態である旨が表示される。Ｊａｖａ　Ｃａｒｄ（登録商標）技術では、全てのプログラムユニットが、Ｊａｖａ　Ｃａｒｄ（登録商標）バーチャルマシンにより参照されるメモリスペースである６４Ｋ境界内に適合しなければならない。

【００４９】

本発明は、Ｊａｖａ　Ｃａｒｄ（登録商標）技術に関連して説明したが、当業者であれば、本発明は、他の多数のプラットホームにも適用できることが理解されよう。これらプラットホームは、例えば、Ｋバーチャルマシン（ＫＶＭ）技術を含む。ＫＶＭ技術は、１９９９年６月８日、サン・マイクロシステムズ・インクの「The K Virtual Machine (KVM)-A White Paper」に説明されている。

本発明は、ソフトウェア又はファームウェア、並びにプログラマブルゲートアレーデバイス、アプリケーション指向の集積回路（ＡＳＩＣ）及び他のハードウェアにおいて実施されてもよい。

【００５０】

従って、言語サブセットの妥当性検査を行う新規な方法が開示された。本発明の実施形態及び用途を図示して説明したが、この開示の利益を得る当業者であれば、本発明の概念から逸脱せずに、多数の変更がなされ得ることが明らかであろう。それ故、本発明は、特許請求の範囲によってのみ限定されるものとする。

【図面の簡単な説明】

【図１Ａ】　ソースレベルにおける言語と言語サブセットとの間の関係を示すブロック図である。

【図１Ｂ】　クラスファイルレベルにおける言語と言語サブセットとの間の関係を示すブロック図である。

【図２】　本発明の１つの実施形態に基づき言語サブセット妥当性検査を遂行する方法を示すフローチャートである。

【図３】　本発明の１つの実施形態に基づきスタティックな妥当性検査を遂行するところを示すフローチャートである。

【図４Ａ】　本発明の１つの実施形態に基づきメソッドに対するスタティックな妥当性検査を遂行するところを示すフローチャートである。

【図４Ｂ】　本発明の１つの実施形態に基づきメソッドのバイトコードに対するスタティックな妥当性検査を遂行するところを示すフローチャートである。

【図５】　本発明の１つの実施形態に基づきフィールドに対するスタティックな妥当性検査を遂行するところを示すフローチャートである。

【図６】　本発明の１つの実施形態に基づきメソッドベースの妥当性検査を遂行するところを示すフローチャートである。

【図７】　本発明の１つの実施形態に基づきパッケージベースの妥当性検査を遂行するところを示すフローチャートである。

【図８】　本発明の１つの実施形態に基づきパッケージにおけるメソッドを妥当性検査するところを示すフローチャートである。

【図９Ａ】　本発明の１つの実施形態に基づきパッケージにおけるクラスを妥当性検査するところを示すフローチャートである。

【図９Ｂ】　本発明の１つの実施形態に基づき実施されたインターフェイスの数をカウントするところを示すブロック図である。

【図９Ｃ】　本発明の１つの実施形態に基づき継承したインターフェイスの数をカウントするところを示すブロック図である。

【図１０Ａ】　本発明の１つの実施形態に基づきクラスアクセス制御を妥当性検査するところを示すフローチャートである。

【図１０Ｂ】　本発明の１つの実施形態に基づきパッケージデフォールトメソッドがパブリック及び保護されたメソッドとされたかどうか決定するところを示すフローチャートである。

【図１１】　本発明の１つの実施形態に基づきパッケージの限界を妥当性検査するところを示すフローチャートである。

【図１２】　本発明の１つの実施形態に基づきカードをベースとする妥当性検査を遂行するところを示すフローチャートである。

【図１Ａ】

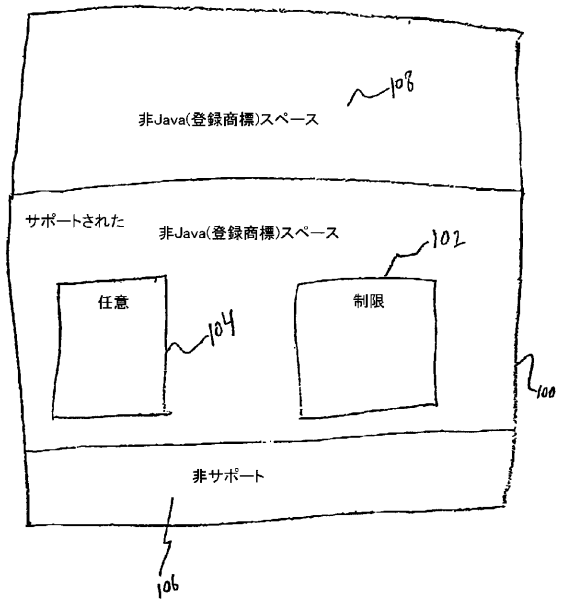

Fig. 1A

【図１Ｂ】

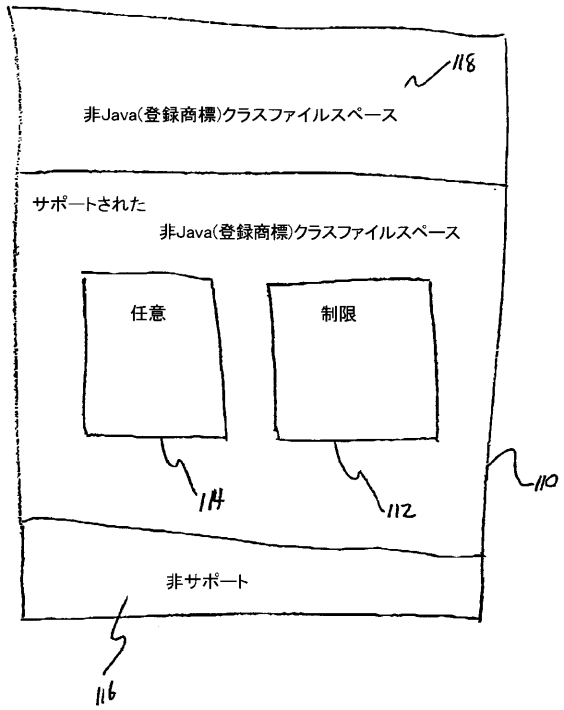

Fig. 1B

【図２】

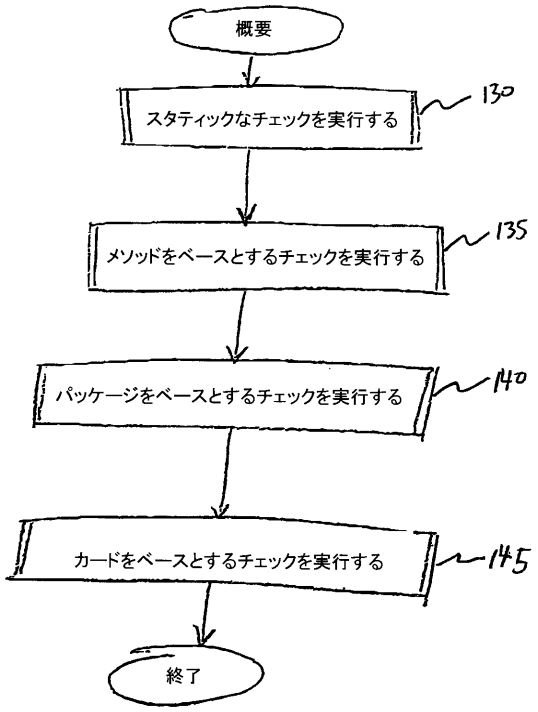

Fig. 2

【図３】

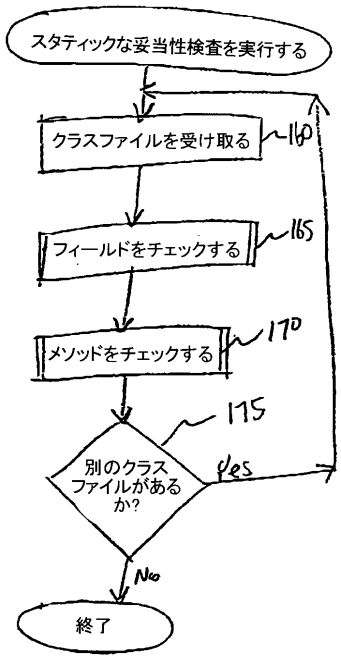

Fig. 3

【図４Ａ】

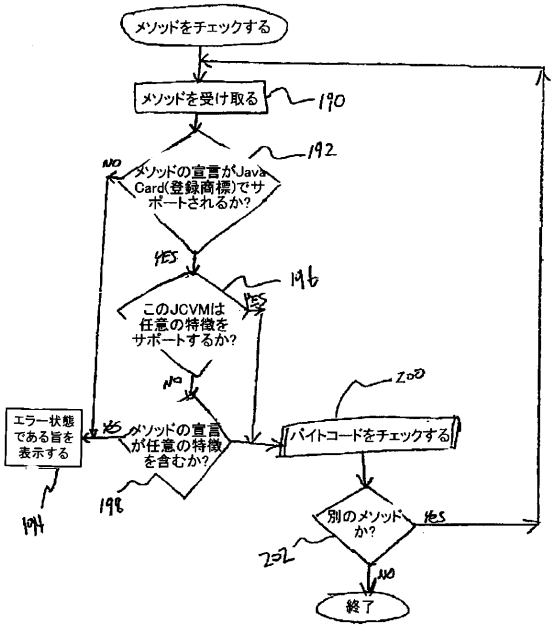

Fig. 4A

【図４Ｂ】

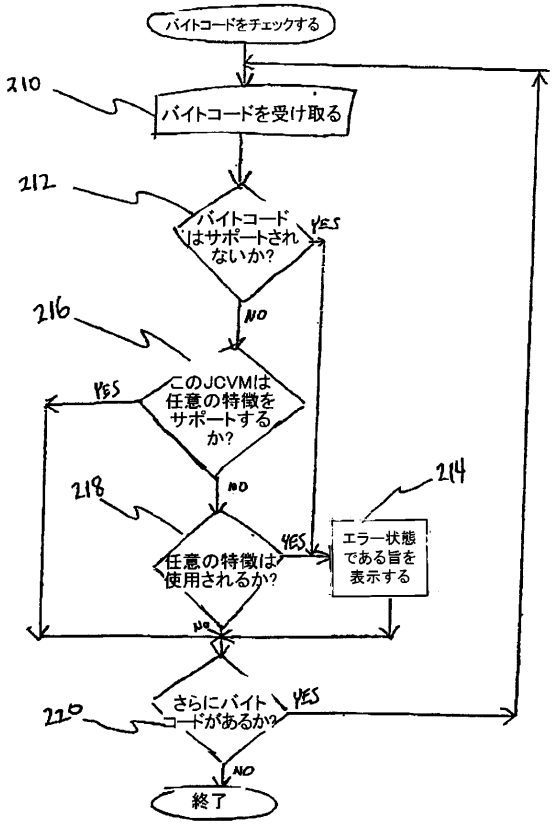

Fig. 4B

【図５】

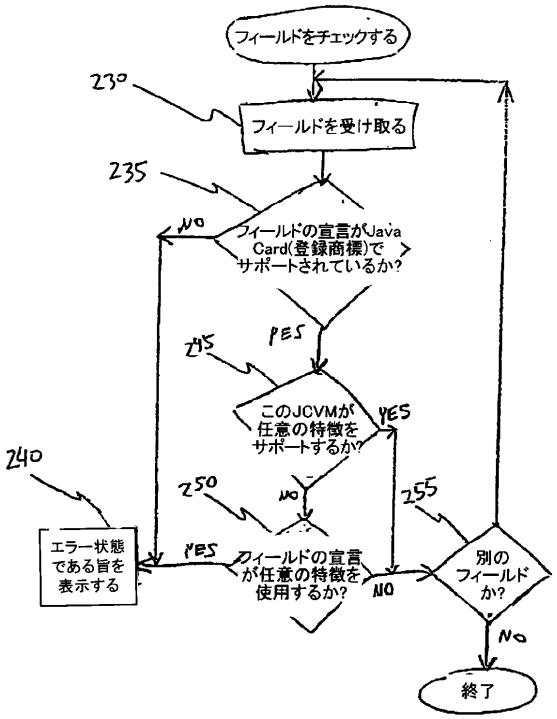

Fig. 5

【図６】

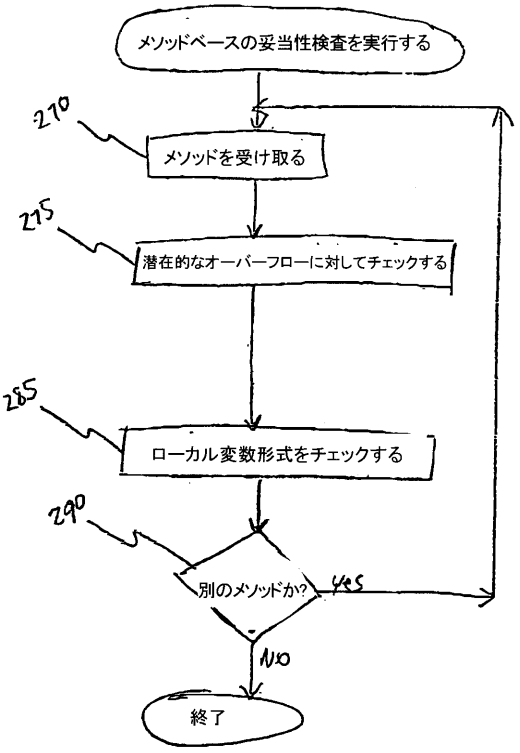

Fig. 6

【図７】

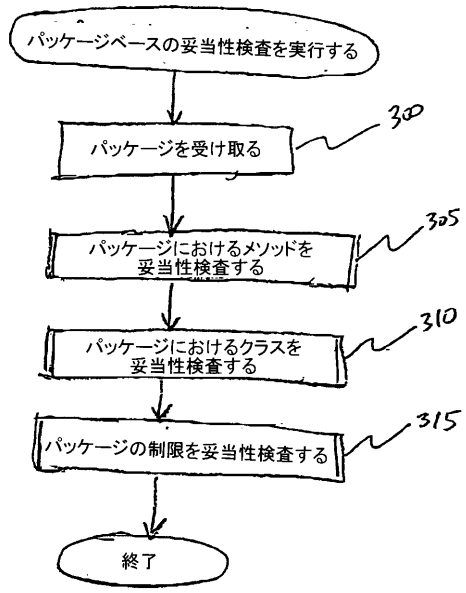

Fig. 7

【図８】

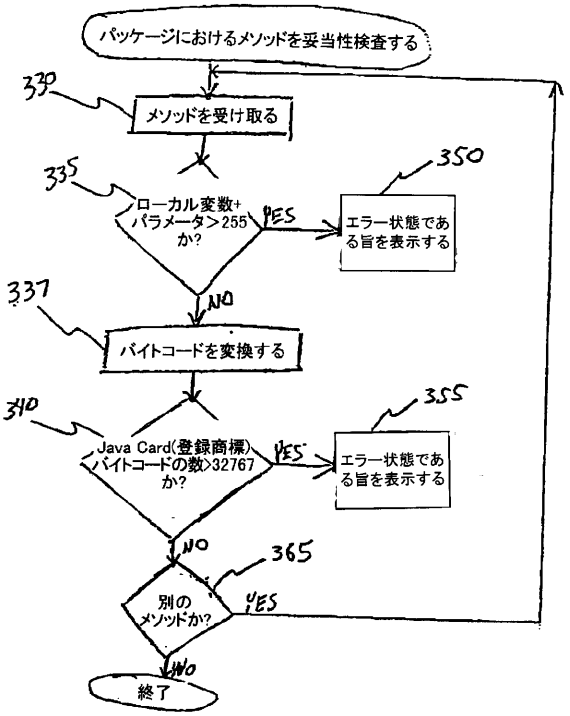

Fig. 8

【図９Ａ】

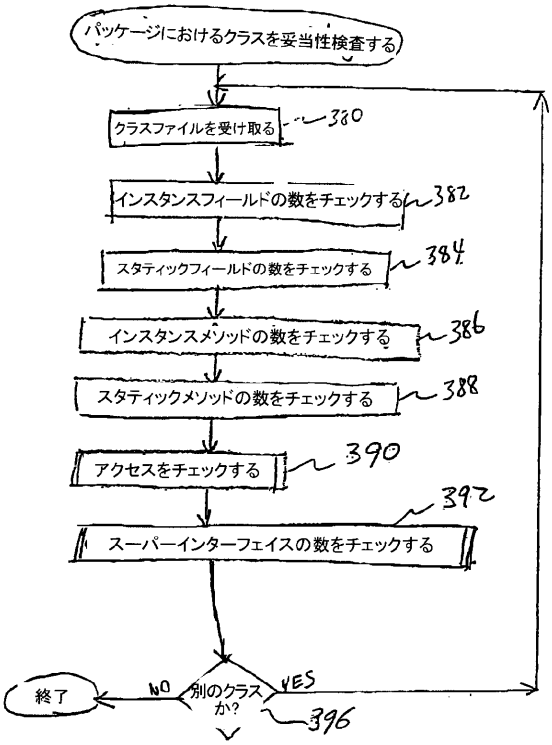

Fig. 9A

【図９Ｂ】

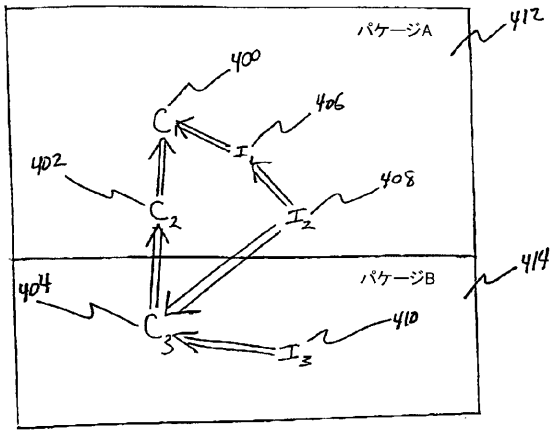

Fig. 9B

【図９Ｃ】

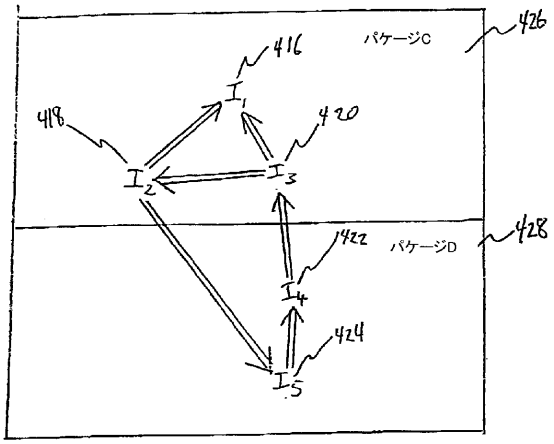

Fig. 9C

【図１０Ａ】

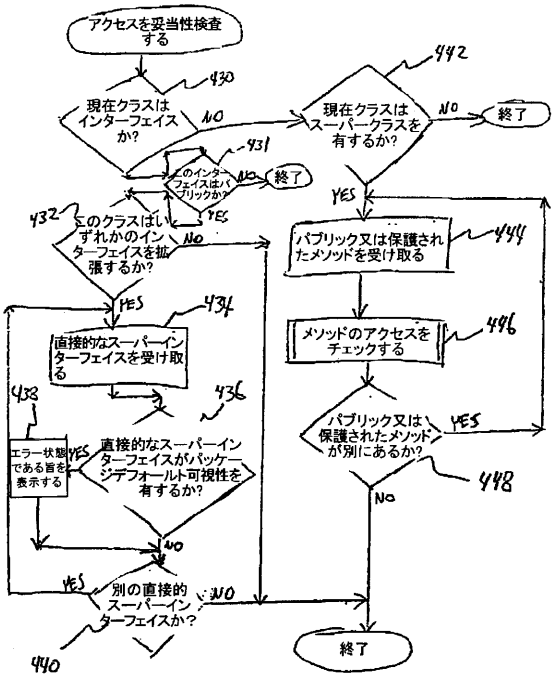

Fig. 10A

【図１０Ｂ】

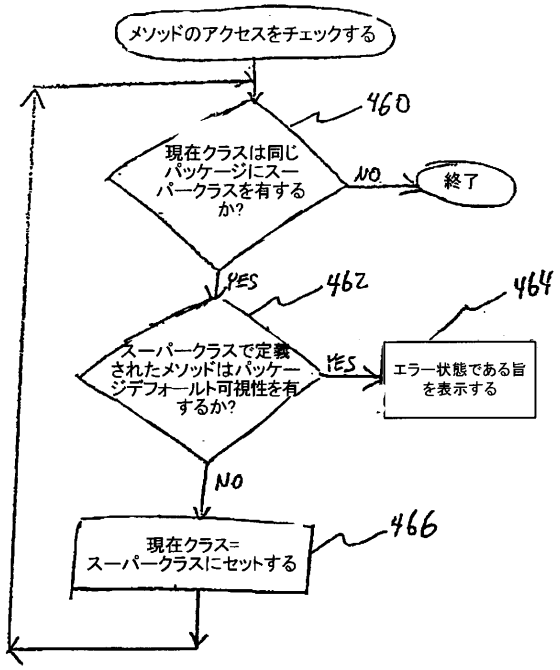

Fig. 10B

【図１１】

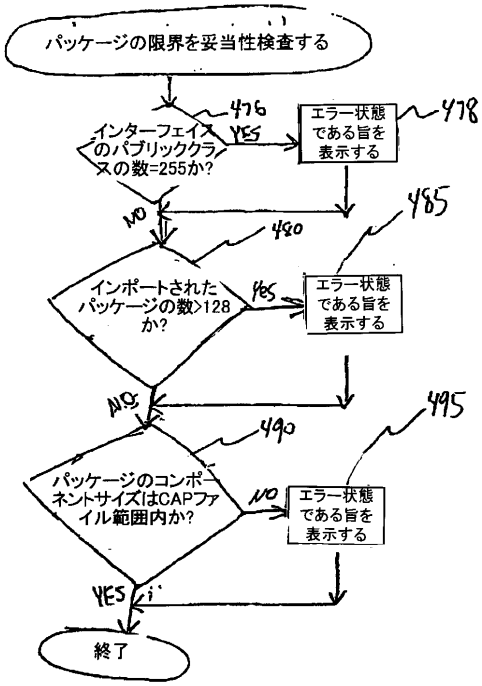

Fig. 11

【図１２】

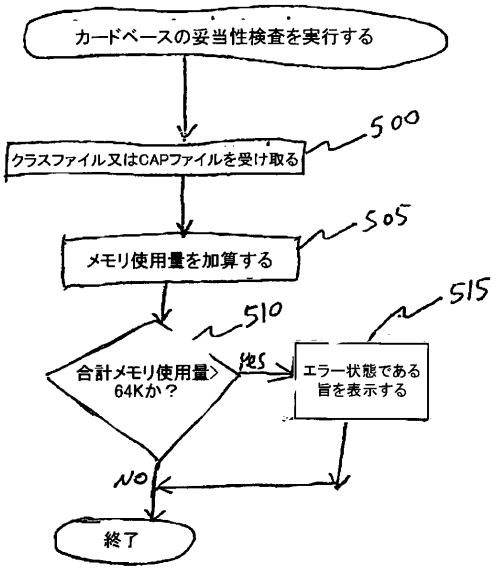
カードベースの妥当性検査を実行する
クラスファイル又はCAPファイルを受け取る
500
メモリ使用量を加算する
505
510
515
合計メモリ使用量>
64Kか？
YES
エラー状態である
旨を表示する
NO
終了

Fig. 12

フロントページの続き

(56)参考文献 特開平１１－０７３３２８（ＪＰ，Ａ）
国際公開第９９／０４９３９２（WO，Ａ１）
国際公開第９８／０３７５２６（WO，Ａ１）
河南 敏，最新Ｊａｖａコンセプトｖｏｌ．３ Java Card: プラットフォーム非依存の多機能アプリケーションの実現，Ｃｏｍｐｕｔｅｒ Ｔｏｄａｙ，日本，株式会社サイエンス社，１９９９年 1月 1日，第16巻、第１号，第68-73頁
早坂 利之，プログラマに聞く アプリケーション配布につまずかないための注意点，日経ソフトウェア，日本，日経ＢＰ社，１９９９年 8月２４日，第２巻、第10号，第108-115頁
平鍋 健児，Ｊａｖａとオブジェクト指向の真価を引き出すソフトウェア設計基礎講座 特別編 Ｊａｖａのデザイン・パターン，Ｊａｖａ ＷＯＲＬＤ，日本，株式会社ＩＤＧコミュニケーションズ，１９９９年１１月 1日，第３巻、第11号，第88-92頁
高田・田丸著，オープン化の為のリアルタイム技術とITRONプロジェクトにおける取組み，システム／制御／情報，日本，システム制御情報学会，１９９９年 1月１５日，第43巻、第１号，第34-41頁
チェン ジクン，Ｊａｖａが切り開くデバイス・ディベロップメントの新地平 Ｊａｖａデバイス・フロンティア 第2回 Ｊａｖａ Ｃａｒｄ用のアプレットを作成する，Ｊａｖａ ＷＯＲＬＤ，日本，株式会社ＩＤＧコミュニケーションズ，１９９９年１１月 1日，第３巻、第11号，第102-109頁

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 11/36
G06F 9/44
G06F 11/28