(12) 特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2011 年 4 月 21 日(21.04.2011)

PCT

(10) 国際公開番号
WO 2011/045949 A1

(51) 国際特許分類:
G06F 12/00 (2006.01) *G06F 12/02* (2006.01)
G06F 9/44 (2006.01)

(21) 国際出願番号: PCT/JP2010/053520

(22) 国際出願日: 2010 年 3 月 4 日(04.03.2010)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(30) 優先権データ:
特願 2009-236257 2009 年 10 月 13 日(13.10.2009) JP

(71) 出願人 (米国を除く全ての指定国について): 株式会社日立製作所 **(HITACHI, LTD.)** [JP/JP]; 〒1008280 東京都千代田区丸の内一丁目６番６号 Tokyo (JP).

(72) 発明者; および

(75) 発明者/出願人 (米国についてのみ): 山田　雅信 **(YAMADA Masanobu)** [JP/JP]; 〒2448555 神奈川県横浜市戸塚区戸塚町５０３０番地　株式会社日立製作所　ソフトウェア事業部内 Kanagawa (JP). 岡田　浩一 **(OKADA Koichi)** [JP/JP]; 〒2448555 神奈川県横浜市戸塚区戸塚町５０３０番地　株式会社日立製作所　ソフトウェア事業部内 Kanagawa (JP). 長瀬　卓真**(NAGASE Takuma)** [JP/JP]; 〒2448555 神奈川県横浜市戸塚区戸塚町５０３０番地　株式会社日立製作所　ソフトウェア事業部内 Kanagawa (JP).

(74) 代理人: 磯野　道造**(ISONO Michizo)**; 〒1020093 東京都千代田区平河町２丁目７番４号　砂防会館別館内　磯野国際特許商標事務所気付 Tokyo (JP).

(81) 指定国 (表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

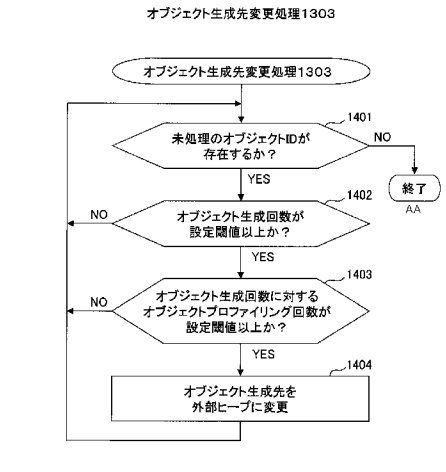
(84) 指定国 (表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), ヨーロッパ (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL,

[続葉有]

(54) **Title**: MEMORY MANAGEMENT METHOD, MEMORY MANAGEMENT PROGRAM, AND INFORMATION PROCESSING DEVICE

(54) 発明の名称 : メモリ管理方法、メモリ管理プログラム、及び、情報処理装置

[図14]



1303... OBJECT GENERATION DESTINATION CHANGE PROCESS
1401... IS UNPROCESSED OBJECT ID PRESENT?
1402... IS THE NUMBER OF OBJECT GENERATION EQUAL TO OR MORE THAN A SET THRESHOLD VALUE?
1403... IS THE NUMBER OF OBJECT PROFILING WITH RESPECT TO THE NUMBER OF OBJECT GENERATION EQUAL TO OR LARGER THAN A SET THRESHOLD VALUE?
1404... CHANGE OBJECT GENERATION DESTINATION TO EXTERNAL HEAP
AA... END

(57) **Abstract**: First, a reference relationship of a long lived object released in the Old area by FullGC is analyzed to detect an origin object. Next, with respect to each object generation command, the ratio at which the command-generated origin objects are released in the Old area by FullGC is profiled so as to detect an object generation command which generates a long lived origin object. If the object is long lived, a destination of the object generation is changed to an external heap, and the origin object is generated in the external heap area. Further, if a member object referred to from the origin object is to be moved from the New area to the Old area by CopyGC and the ratio at which member objects are released together with the origin object is high, the member object is moved to the same external heap as the origin object.

(57) 要約: まず、FullGC により Old 領域で解放される長命なオブジェクトの参照関係を解析して起点オブジェクトを検出する。次に、オブジェクト生成命令ごとに、その命令により生成した起点オブジェクトが FullGC により Old 領域で解放される割合をプロファイリングして長命な起点オブジェクトを生成するオブジェクト生成命令を検出する。長命であれば、そのオブジェクト生成先を外部ヒープに変更し、起点オブジェクトを外部ヒープ領域に生成する。更に、起点オブジェクトから参照されるメンバオブジェクトが CopyGC による New 領域から Old 領域への移動対象となり、起点オブジェクトと一緒に解放される割合が高ければ、起点オブジェクトと同じ外部ヒープに移動する。

WO 2011/045949 A1

NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

添付公開書類:

— 国際調査報告（条約第 21 条(3)）

明　細　書

発明の名称 ：

メモリ管理方法、メモリ管理プログラム、及び、情報処理装置

技術分野

[0001] 本発明は、メモリ管理方法、メモリ管理プログラム、及び、情報処理装置に関するものである。

背景技術

[0002] Java（登録商標）言語の処理系であるJava仮想マシン（以下、「JavaVM」という。「VM」は「Virtual Machine」の略。）は、ガベージコレクタを実装している。ガベージコレクタは、JavaVMがオブジェクトを格納するために確保するメモリ領域（以下、「Javaヒープ領域」という。）を管理し、プログラム実行中に動的に生成したオブジェクト（が割当てられたメモリ領域）のうち、不要になったオブジェクト（が割当てられたメモリ領域）をガベージコレクション（以下、「GC」という。「GC」は、「garbage collection」の略。）によって動的に解放する（自動解放機能）。説明の便宜上、メモリ領域の解放を、そのメモリ領域に格納されているオブジェクトの解放として説明する場合がある。

[0003] このGCの方式の１つに、世代別GCがある。GCの方式として世代別GCを採用しているJavaVMでは、Javaヒープ領域をNew領域とOld領域のように世代別に分けている。さらに、New領域をEden領域とSurvivor領域に、Survivor領域をFrom領域とTo領域のように分けている。

[0004] プログラム実行中に動的に生成されたオブジェクトは、最初、Eden領域に格納される。Eden領域が枯渇した場合には、ガベージコレクタが、アプリケーションプログラムの実行停止時間が短時間で済む、New領域のみを対象としたCopyGCを実行する。このCopyGCでは、レジスタ、実行時スタック、グローバル変数といったルートセットからオブジェクト参照を順次辿ることによって到達可能なオブジェクトのうち、Eden領域に格納されたオブジェクト、及

び、From領域に格納されたオブジェクトをコピー先の領域にコピーする。ここで、コピー先の領域は、コピー対象のオブジェクトが一定回数以上のCopyGCを経たオブジェクト、即ち、長命なオブジェクトの場合にはOld領域である。また、一定回数以上のCopyGCを経ていないオブジェクト、即ち、まだ長命であるか否かわからないオブジェクトの場合にはTo領域である。そして、コピーされることのなかったオブジェクトを不要なオブジェクトとして解放する。なお、CopyGCの終了後には、次回のCopyGCに備え、From領域とTo領域の役割を交代する。

[0005] 他方、Old領域が枯渇した場合には、ガベージコレクタが、アプリケーションプログラムの実行停止時間が長時間に及ぶ、New領域とOld領域の両方を含む全領域を対象としたFullGCを実行する。このFullGCでは、ルートセットからオブジェクト参照を順次辿ることによって到達可能なオブジェクトに対して、プログラムの実行に必要なオブジェクトであることを示す印をつけていく。そして、ルートセットから到達可能なオブジェクトを全て辿った後で、印のついていないオブジェクトを不要なオブジェクトとして解放する。

[0006] 近年、先に説明したJava を基盤としたシステム（以下、「Javaシステム」という。）は、金融系や基幹系といった２４時間、３６５日、正常に機能し続けなければならない等のミッションクリティカルな分野でも利用されることが多くなっている。また、Javaシステムは大規模化、複雑化の傾向にあり、扱うデータ量も増加している。しかし、Javaヒープ領域を大きくすると、ガベージコレクタが処理しなければならないデータ量も多くなるため、GCの発生による業務プログラムの実行停止時間が長時間化する傾向にある。そのため、JavaシステムにおけるGCの発生、特に、FullGCの発生が大きな問題の１つになっている。

[0007] この問題を解決する方法として、特許文献１に開示されている方法が知られている。特許文献１に開示されている方法では、JavaVMは、GCによるオブジェクト解放の対象となるJavaヒープ領域の他に、GCによらないオブジェクト解放の対象となるヒープ領域（以下、「外部ヒープ領域」という。）を有

する。外部ヒープ領域は、プログラム開発者によるメモリ管理が可能な領域であり、外部ヒープ領域に対するオブジェクトの生成、及び、解放は、プログラム開発者によるプログラムソースコード記述に従うようになっている。即ち、ガベージコレクタによる自動メモリ管理に対して、プログラムソースコード記述による明示的なメモリ管理（以下、「明示メモリ管理」という。）を行うことができる。

[0008] プログラム開発者は、外部ヒープ領域内の外部ヒープにオブジェクトを生成するようプログラムソースコードを記述することによって、プログラム実行中、必要に応じて外部ヒープが確保され、その外部ヒープにオブジェクトが生成される。また、外部ヒープに格納されたオブジェクトを解放するため、その外部ヒープを解放するようにプログラムソースコードを記述することによって、外部ヒープの解放が実行される。外部ヒープの解放では、解放対象の外部ヒープに以降のプログラム実行において必要なオブジェクトが存在するか否かをチェックする。チェックの結果、必要なオブジェクトが存在しない場合には、その外部ヒープを解放する。また、必要なオブジェクトが存在する場合には、解放対象の外部ヒープとは別のヒープ領域にそのオブジェクトを移動した上で、その外部ヒープを解放する。そして、外部ヒープを解放することによって、その外部ヒープに格納されたオブジェクトをGCによらず解放する。長命なオブジェクトを外部ヒープに確保、及び、外部ヒープで解放することによって、Old領域の使用量が抑えられ、Old領域の枯渇によるFullGCの発生を抑止することができる。

先行技術文献

特許文献

[0009] 特許文献1：特開２００９－３７５４７号公報

発明の概要

発明が解決しようとする課題

[0010] Old領域の枯渇によるFullGCの発生を抑止するために外部ヒープ領域を利用

する場合、プログラム開発者は、プログラム実行時のオブジェクトの寿命を推定し、外部ヒープ領域に格納すべき長命なオブジェクトが外部ヒープ領域に格納されるように、プログラムソースコードを記述する必要がある。しかし、オジェクトの寿命を推定するためには、プログラム実行時のオブジェクトの参照関係に基づくオブジェクトのライフサイクルを考慮する必要があり、高度な知識と経験が必要である。

また、Javaシステムの大規模化や複雑化、複数人分担によるプログラム開発、及び、プログラムソースコードに記述されていない暗黙的なオブジェクト生成とオブジェクト参照の存在といった様々な理由によって、プログラム開発者が、プログラム実行時に動的に変化するオブジェクトの参照関係を予め把握することは極めて困難である。そのため、外部ヒープ領域に格納すべき長命なオブジェクトであっても、実際は外部ヒープ領域に格納できていないオブジェクトが存在する可能性がある。逆に、このようなオブジェクトを動的に検出し外部ヒープ領域に格納できれば、FullGCの発生をさらに抑止できる。

[0011] 本発明の目的は、外部ヒープ領域に格納すべき長命なオブジェクトの多くを動的に検出できるようにすること、それらオブジェクトを外部ヒープ領域に格納できるようにすることである。

課題を解決するための手段

[0012] 上記目的を達成するために、本発明は、Javaシステムのような、メモリ管理機構を備える言語処理実行システムにおいて、まず、FullGCによってOld領域で解放される長命なオブジェクトの参照関係を解析することで、参照関係のルートに当たるオブジェクト（以下、「起点オブジェクト」という。）を動的に検出する。次に、オブジェクト生成命令ごとに、そのオブジェクト生成命令によって生成した起点オブジェクトがFullGCによってOld領域で解放される割合をプロファイリングすることで、長命な起点オブジェクトを生成しているオブジェクト生成命令を検出する。そして、長命な起点オブジェクトを生成しているオブジェクト生成命令については、そのオブジェクト生成先

を外部ヒープに変更し、起点オブジェクトを外部ヒープ領域に生成するようにする。さらに、起点オブジェクトから参照されているオブジェクト（以下、「メンバオブジェクト」という。）について、メンバオブジェクトがCopyGCによるNew領域からOld領域への移動対象となった場合には、起点オブジェクトが解放されることによって一緒に解放される割合の高いメンバオブジェクトを起点オブジェクトと同じ外部ヒープに移動させる。

このようにして、外部ヒープ領域に格納すべき長命なオブジェクトを外部ヒープ領域に格納することができるので、GCの発生、特に、FullGCの発生を抑止することができる。

詳細は後記する。

発明の効果

[0013] 本発明によれば、外部ヒープ領域に格納すべき長命のオブジェクトの多くを動的に検出し、実際に外部ヒープ領域に格納することができる。

図面の簡単な説明

[0014] [図1]本発明の実施の形態のJavaVMが動作する計算機システム１０１の構成、及び、入出力データを示す図である。

[図2]Java言語により記述されたJavaソースファイル１０４の例を示す図である。

[図3]本発明の実施の形態のオブジェクト解析情報１２６の一例を示す図である。

[図4]本発明の実施の形態のオブジェクト生成情報テーブル３０１の一例を示す図である。

[図5]本発明の実施の形態のオブジェクトプロファイリング情報テーブル３０２の一例を示す図である。

[図6]本発明の実施の形態のメンバオブジェクト情報３０３の一例を示す図である。

[図7]本発明の実施の形態のオブジェクトプロファイリング用オブジェクトのデータ構造の一例を示す図である

[図8]本発明の実施の形態のオブジェクト種別ビットの一例を示す図である。

[図9]本発明の実施の形態の外部ヒープ１１７のデータ構造の一例を示す図である。

[図10]本発明の実施の形態のFullGC処理１７０３のステップ１８０１の処理の終了時点におけるJavaヒープ領域１１５のOld領域に格納されたオブジェクトの状態を示す図である。

[図11]本発明の実施の形態のオブジェクトプロファイリング処理１８０３のステップ１９０１からステップ１９０４までの処理を実行することによって、起点オブジェクトを特定する具体的な手順を示す図である。

[図12]本発明の実施の形態のCopyGC処理におけるメンバオブジェクトの移動の手順を説明するための図である。

[図13]本発明の実施の形態のバイトコード読み込み処理の手順を示すフローチャートである。

[図14]本発明の実施の形態のオブジェクト生成先変更処理１３０３の手順を示すフローチャートである。

[図15]本発明の実施の形態のメソッド実行処理の手順を示すフローチャートである。

[図16]本発明の実施の形態のオブジェクト生成処理１５０２の手順を示すフローチャートである。

[図17]本発明の実施の形態のGC処理１６０２の手順を示すフローチャートである。

[図18]本発明の実施の形態のFullGC処理１７０３の手順を示すフローチャートである。

[図19]本発明の実施の形態のオブジェクトプロファイリング処理１８０３の手順を示すフローチャートである。

[図20]本発明の実施の形態のメンバオブジェクト処理１９０４の手順を示すフローチャートである。

[図21]本発明の実施の形態のCopyGC処理１７０２の手順を示すフローチャー

トである。

[図22]本発明の実施の形態のメンバオブジェクト移動処理の手順を示すフローチャートである。

[図23]本発明の実施の形態のネットワーク構成を示す図である。

発明を実施するための形態

[0015] 以下、本発明を実施するための形態として、JavaVMを例に、適宜図面を参照して詳細に説明する。

図１は、本発明の実施の形態のJavaVMが動作する計算機システム１０１の構成、及び、入出力データを示す図である。

[0016] 計算機システム（情報処理装置）１０１では、Java言語によって記述されたJavaソースファイル１０４がJavaコンパイラ１０９によってコンパイルされることによって、JavaVM１１０がインタプリタ実行により解釈、及び、実行可能な形式であるバイトコードで記述されたJavaクラスファイル１０３が生成される。さらに、Javaクラスファイル１０３がJavaVM１１０によって実行されることで、所定の業務処理が実行される。従って、Javaクラスファイル１０３は、前記所定の業務処理を実行するためのプログラムといえる。

[0017] 計算機システム１０１は、プロセッサ（制御部）１２９、主記憶（記憶部）１０７、補助記憶（記憶部）１０２、入力装置１３０、及び、出力装置１３１といったハードウェア資源を含む。プロセッサ１２９、主記憶１０７、補助記憶１０２、入力装置１３０、及び、出力装置１３１は、バス１２８を介して互いに接続される。

[0018] プロセッサ１２９は、主記憶１０７に記憶された各種プログラムを実行することによって各種処理を実行する。

主記憶１０７は、プログラム、プログラムの実行に必要な情報、及び、プログラムの処理結果を記憶する。主記憶１０７は、揮発性メモリであってもよいし、不揮発性メモリであってもよい。主記憶１０７は、Javaコンパイラ１０９、JavaVM１１０、データ領域１０８、及び、オペレーティングシステム１２７を記憶する。

[0019] 補助記憶１０２は、業務処理の実行に使用されるJavaクラスファイル１０３、このJavaクラスファイル１０３を生成するためのJavaソースファイル１０４、オブジェクト解析情報１２６の保存に使用されるオブジェクト解析情報ファイル１０５、及び、JavaVM１１０の動作を制御するフラグやパラメータの閾値を記録する設定ファイル１０６を含む。

[0020] 入力装置１３０は、計算機システム１０１に必要な情報を入力するためのインタフェースである。入力装置１３０は、例えば、キーボード、又は、マウスなどである。

出力装置１３１は、処理結果などの情報を外部に出力する装置である。出力装置１３１は、例えば、ディスプレイなどである。

[0021] データ領域１０８は、プログラムの実行に必要な情報、及び、プログラムの処理結果が記憶される領域である。データ領域１０８は、メソッドのバイトコードであるバイトコード情報１１２、GC対象のオブジェクト格納領域であるJavaヒープ領域（第１のメモリ領域）１１５、GC対象外のオブジェクト格納領域である外部ヒープ領域（第２のメモリ領域）１１６、及び、オブジェクトの生成に係る情報やオブジェクトのプロファイリングに係る情報などであるオブジェクト解析情報１２６を含む。外部ヒープ領域１１６は、さらに、外部ヒープ１（１１７）、外部ヒープ２（１１８）、外部ヒープ３（１１９）、・・・といったように、外部ヒープ領域１１６内で必要に応じて確保と解放が可能な領域を含む。

[0022] ここで、主記憶１０７に記憶されるJavaVM１１０について説明する。JavaVM１１０は、不要になったメモリ領域を動的に解放するメモリ管理機構を備える言語処理実行システムである。

[0023] JavaVM１１０は、バイトコード読み込み部１１１、オブジェクト生成先変更部１１３、メソッド実行部１１４、メモリ管理部１２０、及び、オブジェクト解析部１２５を含む。バイトコード読み込み部１１１、オブジェクト生成先変更部１１３、メソッド実行部１１４、メモリ管理部１２０、及び、オブジェクト解析部１２５は、プロセッサ１２９によって実行されるプログラ

ムである。

[0024] バイトコード読み込み部１１１は、JavaVM１１０が受け付けたJavaクラスファイル１０３からメソッドのバイトコードを読み込み、データ領域１０８に当該メソッドに係るバイトコード情報１１２を生成する。その際、オブジェクト生成先変更部１１３を実行することによってバイトコード情報１１２に存在するオブジェクト生成命令のオブジェクト生成先を変更する場合や、バイトコード情報１１２に係るオブジェクト解析情報１２６を生成、更新する場合がある。バイトコード読み込み部１１１の処理の詳細は、図１３で説明する。

[0025] オブジェクト生成先変更部１１３は、オブジェクト生成情報テーブル３０１（図４参照）を参照して、オブジェクト生成命令に係るオブジェクト生成先を、適宜、Javaヒープ領域１１５から外部ヒープ領域１１６に変更する。オブジェクト生成先変更部１１３の処理の詳細は、図１４で説明する。

[0026] メソッド実行部１１４は、バイトコード情報１１２の各命令に従い、メソッドを実行する。そして、次に実行する命令がオブジェクト生成命令の場合には、オブジェクト生成部１２１にオブジェクト生成処理を要求する（オブジェクト生成処理要求）。メソッド実行部１１４は、最初、メソッドをインタプリタ実行する。一方、インタプリタ実行において実行頻度が高いメソッドについては、バイトコード情報１１２の命令列を動的にコンパイルし、メソッドをコンパイルコード実行する。なお、メソッド実行部１１４は、最初から最後までメソッドをインタプリタ実行してもよいし、最初からコンパイルコード実行してもよい。また、メソッド実行部１１４が、メソッドをコンパイルコード実行している途中で、バイトコード情報１１２が書き換えられた場合、メソッド実行部１１４は、メソッドをコンパイルコード実行するのを止め、インタプリタ実行に一旦戻り処理を継続する場合もある。メソッド実行部１１４の処理の詳細は、図１５で説明する。

[0027] メモリ管理部１２０は、JavaVM１１０で使用するメモリ領域を管理する。メモリ管理部１２０は、オブジェクト生成部１２１、及び、GC部１２２を含

む。

[0028] オブジェクト生成部１２１は、メソッド実行部１１４からのオブジェクト生成処理要求を受け付けると、オブジェクト生成処理要求の内容に従い、Javaヒープ領域１１５、又は、外部ヒープ領域１１６へのオブジェクト生成を試みる。ここで、オブジェクトを生成する領域に空きがあり、オブジェクト生成に成功した場合には、生成したオブジェクトを呼び元に返却する。

一方、Javaヒープ領域１１５に空きがなく、Javaヒープ領域１１５へのオブジェクト生成に失敗した場合には、GC部１２２にGC処理を要求し（GC処理要求）、GC部１２２がGC処理を実行し、Javaヒープ領域１１５で不要になったメモリ領域を解放する。そして、再びオブジェクト生成を試み、さらに失敗した場合にはメモリ不足エラーを発生させる。

また、外部ヒープ領域１１６に空きがなく、外部ヒープ領域１１６へのオブジェクト生成に失敗した場合には、Javaヒープ領域１１５へのオブジェクト生成を試みる。さらにJavaヒープ領域１１５へのオブジェクト生成にも失敗した場合には、GC部１２２にGC処理を要求し、GC部１２２がGC処理を実行し、Javaヒープ領域１１５で不要になったメモリ領域を解放する。そして、再びオブジェクト生成を試み、さらに失敗した場合にはメモリ不足エラーを発生させる。

[0029] なお、オブジェクト生成部１２１は、生成したオブジェクトに対し、当該オブジェクト生成命令によって生成されたオブジェクトであることを識別するためのオブジェクトIDを生成したオブジェクトに記録し、当該オブジェクト生成命令によってオブジェクトを生成したことをオブジェクトプロファイリング情報テーブル３０２（図５参照）に記録する場合がある。オブジェクト生成部１２１の処理の詳細は、図１６で説明する。

[0030] GC部１２２は、CopyGC部１２４、及び、FullGC部１２３を含む。GC部１２２は、オブジェクト生成部１２１からのGC処理要求を受け付けると、New領域が枯渇している場合にはCopyGC部１２４、Old領域が枯渇している場合にはFullGC部１２３に対して、実際のGC処理を要求する。GC部１２２の処理の詳細

は、図１６で説明する。

[0031] CopyGC部１２４は、GC部１２２からのGC処理要求を受け付けると、Javaヒープ領域１１５内のNew領域のみを対象としたCopyGCを実行する。その際、CopyGCによってJavaヒープ領域１１５のNew領域からOld領域の移動対象となったオブジェクトをNew領域から外部ヒープ１１７（外部ヒープ領域１１６内に確保した外部ヒープの一つであり、「外部ヒープ１」と称する場合もある。）に移動させる場合や、不要な外部ヒープ１１７を解放する場合、及び、オブジェクト生成先変更部１１３を実行することによってバイトコード情報１１２に存在するオブジェクト生成命令のオブジェクト生成先を変更する場合がある。CopyGC部１２４の処理の詳細は、図２１で説明する。

[0032] FullGC部１２３は、GC部１２２からのGC処理要求を受け付けると、Javaヒープ領域１１５内のNew領域とOld領域を含む、GC対象の全領域を対象としたFullGCを実行する。その際、オブジェクト解析部１２５を実行し、オブジェクトプロファイリング処理を実行する場合がある。FullGC部１２３の処理の詳細は、図１８で説明する。

[0033] オブジェクト解析部１２５は、FullGCによってOld領域で解放される長命なオブジェクトの参照関係を解析することで、参照関係のルートに当たる起点オブジェクトを動的に検出する。そして、オブジェクト生成命令ごとに、そのオブジェクト生成命令によって生成した起点オブジェクトがFullGCによってOld領域で解放された回数をプロファイリングする。また、起点オブジェクトから参照されているメンバオブジェクトについて、起点オブジェクトが解放されたことによってメンバオブジェクトも一緒に解放された回数をプロファイリングする。オブジェクト解析部１２５の処理の詳細は、図１９で説明する。

[0034] なお、JavaVM１１０は、動作を制御するためのフラグ（以下、「設定フラグ」という。）やパラメータの閾値（以下、「設定閾値」という。）をもつ。そして、設定フラグや設定閾値のデフォルト値は、設定ファイル１０６や入力装置１３０で指定することによって変更することができる。

[0035] JavaVM１１０は、FullGCによってOld領域で解放される長命なオブジェクトの参照関係を解析することで、外部ヒープ領域１１６に格納すべきオブジェクトを検出し、それらオブジェクトを外部ヒープ領域１１６に格納していくことで、FullGCを抑止していく。一方、FullGCによってOld領域で解放される長命なオブジェクトの参照関係の解析結果を記録したオブジェクト解析情報ファイル１０５を使用し、設定フラグによるJavaVM１１０の動作変更によって、外部ヒープ領域１１６に格納すべきオブジェクトを検出するためのオブジェクトプロファイリングを行わないようにし、JavaVM１１０の起動時からFullGCを抑止した状態で実行してもよい。

[0036] 図２は、Java言語により記述されたJavaソースファイル１０４の例を示す図である。

１行目から１７行目は、Sampleクラスの内容を示している。

３行目から９行目、及び、１１行目から１５行目は、それぞれ、methodA()メソッド、及び、methodB()メソッドの処理の内容を示している。

５行目、及び、１３行目は、クラスの型がClassAであるオブジェクトをJavaヒープ領域１１５に生成することをnew演算子により指示している。また、６行目は、クラスの型がClassZであるオブジェクトをJavaヒープ領域１１５に生成することをnew演算子により指示している。これにより、例えばmethodA()を５回実行すると、クラスの型がClassAであるオブジェクトが５個、クラスの型がClassZであるオブジェクトが５個、それぞれ生成される。さらにmethodB()を１０回実行すると、クラスの型がClassAであるオブジェクトがもう１０個生成される。

[0037] ここで、methodA()とmethodB()の実行により、クラスの型が同じClassAであるオブジェクトが生成されるが、methodA()の５行目で生成を指示されたオブジェクトと、methodB()の１３行目で生成を指示されたオブジェクトとでは、クラスの型が同じClassAであっても同じようなオブジェクトの寿命になるとは限らない。一方、methodA()の実行により、methodA()の５行目で生成を指示された５個のオブジェクトについては、methodA()の同じ処理により生成

されたオブジェクトであるため、同じようなオブジェクトの寿命になる可能性が高いと言える。

[0038] オブジェクトプロファイリングでは、クラスの型が同じであっても、オブジェクトの生成を指示された場所が異なるオブジェクトは区別することで、より詳細にオブジェクトをプロファイリングすることができる。

[0039] 図３は、本発明の実施の形態のオブジェクト解析情報１２６の一例を示す図である。

オブジェクト解析情報１２６は、バイトコード情報１１２に係るオブジェクト解析の情報を格納しており、オブジェクト生成情報テーブル３０１、オブジェクトプロファイリング情報テーブル３０２、及び、メンバオブジェクト情報３０３を含む。メンバオブジェクト生成情報テーブル３０１、オブジェクトプロファイリング情報テーブル３０２、及び、メンバオブジェクト情報３０３の詳細は、それぞれ、図４、図５、及び、図６で説明する。

[0040] 図４は、本発明の実施の形態のオブジェクト生成情報テーブル３０１の一例を示す図である。

オブジェクト生成情報テーブル３０１の各行は、オブジェクト生成命令に係る情報を格納しており、オブジェクト生成命令アドレス４０１、オブジェクトID(Identifier)４０２、クラス名４０３、及び、オブジェクト生成先４０４を含む。

[0041] オブジェクト生成命令アドレス４０１は、オブジェクト生成命令の位置を格納する。オブジェクト生成命令の位置は、例えば、図２に示すJavaソースファイル１０４の例におけるnew演算子のようなオブジェクト生成指示に対応するバイトコード情報１１２内のオブジェクト生成命令が存在するアドレスである。なお、オブジェクト解析情報１２６をオブジェクト解析情報ファイル１０５に保存する場合など、オブジェクト生成命令の位置は、バイトコード情報１１２の先頭からオブジェクト生成命令が存在する位置までのオフセットで示される場合もある。

[0042] オブジェクトID４０２は、オブジェクト生成命令ごとに割当てられる一意

の値を格納する。そして、オブジェクトプロファイリング処理の対象であるオブジェクトのオブジェクトIDフィールド７０２（図７参照）に、当該オブジェクトを生成したオブジェクト生成命令に係るオブジェクトID４０２が格納される。これにより、オブジェクトがどのオブジェクト生成命令によって生成されたかを判別できる。例えば、図２に示すJavaソースファイル１０４の例において、５行目で生成を指示されたクラスの型がClassAであるオブジェクトと１３行目で生成を指示されたクラスの型がClassAであるオブジェクトでは、オブジェクトヘッダ７０１に格納されたオブジェクトIDが#0、#4といったように異なる。一方、５行目で生成を指示されたクラスの型がClassAであるオブジェクトが複数存在しても、オブジェクトの生成を指示された位置が同じであるオブジェクトでは、オブジェクトIDが#0といったように同じになる。

[0043] クラス名４０３は、オブジェクト生成命令によって生成されるオブジェクトのクラスの型の名前を格納する。

[0044] オブジェクト生成先４０４は、オブジェクト生成命令によって生成されるオブジェクトの生成先を格納する。オブジェクトの生成先は、初期状態では“Javaヒープ領域”となっているが、オブジェクト生成先変更部１１３によって、オブジェクトプロファイリング処理の結果に応じて“外部ヒープ１”などに変更される。また、オブジェクトの生成先は、“Javaヒープ固定”にすることで、オブジェクト生成先変更部１１３によって変更されないようにしたり、“指定外部ヒープ”にしたりすることで、明示メモリ管理API（明示メモリ管理を実行するAPIをいう。「API」は、「application programming interface」の略。）によって指定された外部ヒープにしたりすることも可能である。なお、明示メモリ管理APIによって指定された外部ヒープに生成されたオブジェクトは、明示メモリ管理APIによって解放され、不要な外部ヒープ１１７ごと自動的に解放されることはない。オブジェクトプロファイリング処理の詳細は、図１９で説明する。

[0045] 図５は、本発明の実施の形態のオブジェクトプロファイリング情報テーブ

ル３０２の一例を示す図である。

オブジェクトプロファイリング情報テーブル３０２は、オブジェクトID４０２に係るオブジェクトについてのオブジェクトプロファイリング処理の結果を格納しており、オブジェクトID５０１、オブジェクト生成回数５０２、オブジェクトプロファイリング回数５０３、及び、メンバオブジェクト情報アドレス５０４を含む。

[0046] オブジェクトID５０１は、オブジェクト生成情報テーブル３０１の行データとオブジェクトプロファイリング情報テーブル３０２の行データを結びつけるためのオブジェクトIDを格納する。このオブジェクトIDをキーとした検索により、このオブジェクトIDによって識別されるオブジェクトについてのオブジェクト生成情報テーブル３０１の行データとオブジェクトプロファイリング情報テーブル３０２の行データが参照可能である。ここで、例えば、オブジェクト生成情報テーブル３０１の行データに対するオブジェクトプロファイリング情報テーブル３０２の行データを予め用意する場合、オブジェクト生成情報テーブル３０１の行データとオブジェクトプロファイリング情報テーブル３０２の行データは１対１に対応するため、オブジェクト生成情報テーブル３０１とオブジェクトプロファイリング情報テーブル３０２を１つのテーブルにまとめてしまってもよい。また、オブジェクトプロファイリング情報テーブル３０２のために確保するメモリ領域を節約するために、オブジェクトID４０２で識別されるオブジェクトが初めて生成されたとき、オブジェクト生成情報テーブル３０１の行データに対するオブジェクトプロファイリング情報テーブル３０２の行データを登録するようにしてもよい。なお、本発明の実施の形態の一例では、オブジェクト生成情報テーブル３０１の行データに対するオブジェクトプロファイリング情報テーブル３０２の行データを予め用意するが、便宜上、オブジェクト生成情報テーブル３０１とオブジェクトプロファイリング情報テーブル３０２を別々のテーブルとして示す。

[0047] オブジェクト生成回数５０２は、オブジェクトID５０１で識別されるオブ

ジェクトをオブジェクトID５０１に係るオブジェクト生成命令によって実際に生成した回数を格納する。

[0048] オブジェクトプロファイリング回数５０３は、オブジェクトID５０１で識別されるOld領域に格納してあるオブジェクトがFullGCによって解放された回数、つまり、オブジェクトID５０１で識別されるオブジェクトに対して、オブジェクトプロファイリング処理を実行した回数を格納する。オブジェクトプロファイリング処理の詳細は、図１９で説明する。

[0049] メンバオブジェクト情報アドレス５０４は、オブジェクトID５０１で識別されるオブジェクトに係るメンバオブジェクト情報３０３のアドレスを格納する。メンバオブジェクト情報３０３の詳細は、図６で説明する。

[0050] 図６は、本発明の実施の形態のメンバオブジェクト情報３０３の一例を示す図である。

メンバオブジェクト情報３０３は、オブジェクトプロファイリング処理によってプロファイリングされたオブジェクトの参照関係の状況についての情報を格納しており、起点オブジェクトについての情報を格納するデータ構造（以下、「起点オブジェクトデータ構造」という。）、及び、メンバオブジェクトについての情報を格納するデータ構造（以下、「メンバオブジェクトデータ構造」という。）を含む。起点オブジェクトは、オブジェクトプロファイリング処理の起点となるOld領域に格納してあるオブジェクトであり、前回のFullGCの終了後から今回のFullGCの開始前までの間に、ルートセットや必要オブジェクトから参照されなくなったオブジェクト、又は、ルートセットや必要オブジェクトから参照されなくなった可能性の高いオブジェクトである。また、メンバオブジェクトは、起点オブジェクトからオブジェクト参照を順次辿ることによって到達可能なオブジェクトのうち、起点オブジェクトがFullGCによって解放されることによって一緒に解放されるオブジェクト、又は、一緒に解放される可能性の高いオブジェクトである。オブジェクトプロファイリング処理の詳細は、図１９で説明する。

[0051] 本発明の実施の形態の一例では、メンバオブジェクト情報３０３は、起点

オブジェクトデータ構造６０１、及び、メンバオブジェクトデータ構造６０２～６０７がポインタによって接続された木構造として示す。また、図６において、起点オブジェクトデータ構造は二重線の長方形で示し、メンバオブジェクトデータ構造は一重線の長方形で示す。

[0052] 起点オブジェクトデータ構造６０１は、起点オブジェクトから参照先メンバオブジェクトへの参照関係の状況を記録するために、メンバオブジェクトに対応するメンバオブジェクトデータ構造へのポインタをオブジェクト参照フィールドごとに格納する。例えば、本発明の実施の形態の一例は、起点オブジェクトデータ構造６０１が、メンバオブジェクトデータ構造６０２、及び、メンバオブジェクトデータ構造６０３をポインタによって接続していることを示す。

[0053] メンバオブジェクトデータ構造６０２～６０７は、メンバオブジェクトから参照先メンバオブジェクトの参照関係の状況を記録するために、参照先メンバオブジェクトに対応するメンバオブジェクトデータ構造へのポインタをオブジェクト参照フィールドごとに格納する。例えば、本発明の実施の形態の一例は、メンバオブジェクトデータ構造６０３が、メンバオブジェクトデータ構造６０５とメンバオブジェクトデータ構造６０６をポインタによって接続していることを示す。なお、ポインタ６０８は、クラスの型がClassFであるオブジェクトにオブジェクト参照フィールドは存在するが、そのオブジェクト参照フィールドによって参照しているオブジェクトは記録されていないことを示す（NULL）。

[0054] また、メンバオブジェクトデータ構造６０２～６０７は、メンバオブジェクトのクラスの型の名前と、起点オブジェクトがFullGCによって解放されたことによってメンバオブジェクトも一緒に解放された回数、つまり、起点オブジェクトに対してオブジェクトプロファイリング処理を実行したときに、メンバオブジェクトに対してもオブジェクトプロファイリング処理を実行した回数（以下、「メンバオブジェクトプロファイリング回数」という。）を格納する。例えば、本発明の実施の形態の一例は、起点オブジェクトデータ

構造６０１によって示される起点オブジェクトがFullGCによって解放されたことによってメンバオブジェクトデータ構造６０５によって示されるClassEのメンバオブジェクトも一緒に解放された回数が２回であることを示す。

ここで、ポインタに付された数字は、便宜上のメンバオブジェクト適合率を示す。メンバオブジェクト適合率は、起点オブジェクトがFullGCによって解放されたことによってメンバオブジェクトも一緒に解放された割合である。例えば、メンバオブジェクトデータ構造６０６によって示されるClassFのメンバオブジェクト適合率は、メンバオブジェクトデータ構造６０１によって示されるClassAの起点オブジェクトが４回解放されたうちClassFのメンバオブジェクトは３回解放されたので、0.75になる。また、メンバオブジェクトデータ構造６０７によって示されるClassGのメンバオブジェクト適合率は、メンバオブジェクトデータ構造６０１によって示されるClassAの起点オブジェクトが４回解放されたうちClassGのメンバオブジェクトは１回解放されたので、0.25になる。

また、破線の囲み線はメンバオブジェクト適合率が0.6である便宜上の境界を示しており、破線で囲まれた内側は、メンバオブジェクト適合率が0.6以上であることを示している。

[0055] なお、メンバオブジェクト情報アドレス５０４には、メンバオブジェクト情報３０３のアドレスとして、起点オブジェクトデータ構造のアドレスが格納される。例えば、本発明の実施の形態の一例は、オブジェクトID５０１に格納されているオブジェクトIDが#4であるオブジェクトに対するメンバオブジェクト情報アドレス５０４に、起点オブジェクトデータ構造６０１のアドレスである0x81b2e23cが格納されていることを示す。また、本発明の実施の形態では一例として、起点オブジェクトデータ構造６０１がオブジェクトID（下付き数字の「４」）、オブジェクトのクラスの型の名前（ClassA）、プロファイリングの実行回数（括弧内の数字の「４」）を格納するように便宜上記載してある。

[0056] 図７は、本発明の実施の形態のオブジェクトプロファイリング用オブジェ

クトのデータ構造の一例を示す図である。オブジェクトプロファイリング用オブジェクトは、オブジェクトヘッダ７０１、オブジェクトIDフィールド７０２、及び、オブジェクトデータ７０３から構成される。オブジェクトプロファイリング用オブジェクトとは、オブジェクトプロファイリング処理の対象となるオブジェクトである。

[0057] オブジェクトヘッダ７０１は、オブジェクトの管理情報である。

オブジェクトIDフィールド７０２は、オブジェクトを識別するためのオブジェクトIDを格納する。このオブジェクトIDは、当該オブジェクトを生成したオブジェクト生成命令に係るオブジェクトID４０２である。なお、オブジェクトプロファイリング処理の対象ではない通常オブジェクトのデータ構造は、オブジェクトヘッダ７０１とオブジェクトデータ７０３で構成される。また、オブジェクトプロファイリング処理の対象であっても、オブジェクトヘッダ７０１の空き領域やオブジェクトデータ７０３、及び、オブジェクトのデータ構造以外の領域にオブジェクトIDを格納するなどしてオブジェクトを識別できれば、オブジェクトIDフィールドのための領域を別途確保する必要はない。

オブジェクトデータ７０３は、オブジェクトのデータ本体である。

[0058] なお、オブジェクトへのポインタの下位数ビットは、オブジェクトヘッダ７０１のサイズに応じて０とすることができるため、ビットフラグとして使用可能である。例えば、オブジェクトヘッダ７０１のサイズが８バイトである場合、オブジェクトへのポインタの下位３ビットは０になる。本発明の実施の形態では、オブジェクトへのポインタの下位数ビットをビットフラグとして使用する場合がある。

[0059] 図８は、本発明の実施の形態のオブジェクト種別ビットの一例を示す図である。

オブジェクト種別ビット８０２の値は、オブジェクト種別８０１に対応する。本発明の実施の形態では、オブジェクトヘッダ７０１に含まれる２ビットをオブジェクト種別ビットとして使用する。

[0060] 具体的には、オブジェクト種別ビット８０２が“００”の場合には、GC処理やオブジェクトプロファイリング処理において未解析なオブジェクトである“未解析”オブジェクトであることを示す。

また、オブジェクト種別ビット８０２が“０１”の場合には、FullGC処理においてプログラムの実行に必要なオブジェクトであると判定済みの“必要”オブジェクトであることを示す。

同様に、オブジェクト種別ビット８０２が“１０”の場合には、オブジェクトプロファイリング処理の起点となるオブジェクトであり、前回のFullGCから今回のFullGCまでの間に、必要オブジェクトから参照されなくなったオブジェクト、又は、必要オブジェクトから参照されなくなった可能性の高いオブジェクトである“起点”オブジェクトであることを示す。

さらに、オブジェクト種別ビット８０２が“１１”の場合には、起点オブジェクトからオブジェクト参照を順次辿ることによって到達可能なオブジェクトのうち、起点オブジェクトがFullGCによって解放されることによって一緒に解放されるオブジェクト、又は、一緒に解放される可能性の高いオブジェクトである“メンバ”オブジェクトであることを示す。

[0061] 図９は、本発明の実施の形態の外部ヒープ１１７のデータ構造の一例を示す図である。外部ヒープ１１７は、外部ヒープヘッダ９０１、オブジェクト格納領域９０２から構成される。

外部ヒープヘッダ９０１は、オブジェクトの管理情報であり、外部ヒープ１１７に格納された起点オブジェクトを生成したオブジェクト生成命令に係るメンバオブジェクト参照情報アドレス５０４に格納されたメンバオブジェクト情報３０３のアドレスを格納する。

オブジェクト格納領域９０２は、オブジェクトを格納するための領域である。

[0062] 図１３は、本発明の実施の形態のバイトコード読み込み処理の手順を示すフローチャートである。本処理は、プロセッサ１２９がバイトコード読み込み部１１１を処理することによって、実行される。なお、図１０～図１２に

関する説明は後記する。

[0063] プロセッサ１２９がバイトコード読み込み部１１１を実行すると、バイトコード読み込み部１１１は、まず、JavaVM１１０が受け付けたJavaクラスファイル１０３からメソッドのバイトコードを読み込み、データ領域１０８に当該メソッドに係るバイトコード情報１１２を生成する（ステップ１３０１）。

[0064] ステップ１３０１の処理の終了後には、JavaVM１１０の設定フラグを参照し、バイトコード読み込み処理時に、当該メソッドに係るバイトコード情報１１２に存在するオブジェクト生成命令のオブジェクト生成先を変更するか否かを判定する（ステップ１３０２）。

[0065] オブジェクト生成先を変更する場合には（ステップ１３０２の結果が「ＹＥＳ」）、JavaVM１１０が受け付けたオブジェクト解析情報ファイル１０５に保存してある当該メソッドに係るオブジェクト解析情報の実行結果をオブジェクト解析情報１２６に読み込み、当該メソッドに係るオブジェクトプロファイリング情報テーブル３０２に登録された行データで示されるオブジェクトに対して、オブジェクト生成先変更部１１３を実行し、オブジェクト生成先変更処理を実行する（ステップ１３０３）。

[0066] オブジェクト生成先を変更しない場合（ステップ１３０２の結果が「ＮＯ」）、又は、ステップ１３０３の処理が終了した場合には、JavaVM１１０の設定フラグを参照し、オブジェクトプロファイリングを行うか否かを判定する（ステップ１３０４）。

[0067] オブジェクトプロファイリングを行う場合には（ステップ１３０４の結果が「ＹＥＳ」）、ステップ１３０１の処理で生成したバイトコード情報１１２に係るオブジェクト解析情報１２６を生成し、バイトコード情報１１２に存在する全てのオブジェクト生成命令について、オブジェクト生成命令のアドレス、オブジェクト生成命令に割り当てられる一意のオブジェクトID、及び、オブジェクト生成命令で生成されるオブジェクトのクラス名を、それぞれ、オブジェクト生成命令アドレス４０１、オブジェクトID４０２、及び、

クラス名４０３に登録する（更新も含む）。さらに、オブジェクト生成先４０４を“Javaヒープ領域”で登録する（ステップ１３０５）。ただし、ステップ１３０３の処理で、JavaVM１１０が受け付けたオブジェクト解析情報ファイル１０５に保存してある当該メソッドに係るオブジェクト解析情報の実行結果をオブジェクト解析情報１２６に反映しているオブジェクト生成命令については、ステップ１３０５の処理を行わない。

[0068] オブジェクトプロファイリングを行わない場合（ステップ１３０４の結果が「ＮＯ」）、又は、ステップ１３０５の処理の終了後には、本処理を終了する。

[0069] 図１４は、本発明の実施の形態のオブジェクト生成先変更処理１３０３の手順を示すフローチャートである。本処理は、プロセッサ１２９がオブジェクト生成先変更部１１３を処理することによって、実行される。

[0070] プロセッサ１２９がオブジェクト生成先変更部１１３を実行すると、オブジェクト生成先変更部１１３は、まず、処理対象のオブジェクトプロファイリングテーブル３０２について、オブジェクトID５０１に未処理のオブジェクトIDが存在するか否かを判定する（ステップ１４０１）。

[0071] オブジェクトID５０１に未処理のオブジェクトIDが存在する場合には（ステップ１４０１の結果が「ＹＥＳ」）、当該オブジェクトIDで識別されるオブジェクトのオブジェクト生成回数５０２が、JavaVM１１０の設定閾値（第１の閾値）以上であるか否かを判定する（ステップ１４０２）。このステップ１４０２では、オブジェクト生成先を変更するか否かを判定するのに十分な回数のオブジェクト生成が実施済みか否かを判定する。例えば、オブジェクト生成回数の設定閾値が1000回の場合、オブジェクトID５０１が#0であるオブジェクト生成命令は、オブジェクト生成回数５０２が5872回であることから（図５参照）、オブジェクト生成先を変更するか否かを判定するのに十分な回数のオブジェクト生成を実施済みと判定できる。一方、オブジェクトID５０１が#104であるオブジェクト生成命令は、オブジェクト生成回数５０２が950回であることから（図５参照）、オブジェクト生成先を変更するか否か

を判定するのに十分な回数のオブジェクト生成を実施済みでないと判定できる。

[0072] オブジェクト生成回数５０２が設定閾値以上である場合には（ステップ１４０２の結果が「ＹＥＳ」）、当該オブジェクトIDに係るオブジェクト生成回数５０２に対するオブジェクトプロファイリング回数５０３が、JavaVM１１０の設定閾値（第２の閾値）以上であるか否かを判定する（ステップ１４０３）。このステップ１４０３では、当該オブジェクトIDに係るオブジェクト生成命令によって生成されたオブジェクトが、FullGCによって解放された割合の高さ、つまり、当該オブジェクト生成命令によって生成されたオブジェクトが、外部ヒープ領域に格納すべき長命なオブジェクトになる割合の高さに基づき、当該オブジェクト生成命令のオブジェクト生成先を外部ヒープ領域に変更するべきか否かを判定する。例えば、オブジェクト生成回数に対するオブジェクトプロファイリング回数の設定閾値（割合）が0.50の場合、オブジェクトID５０１が#0であるオブジェクト生成命令によって生成されたオブジェクトは5872個であり（図５参照）、そのうち、5830個がFullGCによって解放されたので、FullGCによって解放された割合は0.99であり、設定閾値より大きいので、当該オブジェクト生成命令のオブジェクト生成先を外部ヒープ領域に変更するべきと判定できる。一方、オブジェクトID５０１が#103であるオブジェクト生成命令によって生成されたオブジェクトは10281個であり、そのうち、237個がFullGCによって解放されたので、FullGCによって解放された割合は0.02であり、設定閾値0.50より小さいので、当該オブジェクト生成命令のオブジェクト生成先を外部ヒープ領域に変更するべきではないと判定できる。

[0073] オブジェクト生成回数５０２に対するオブジェクトプロファイリング回数５０３が設定閾値以上である場合には（ステップ１４０３の結果が「ＹＥＳ」）、オブジェクト生成情報テーブル３０１を参照して、当該オブジェクトIDに係るオブジェクト生成先４０４を“Javaヒープ領域”以外の、例えば、“外部ヒープ領域１”に変更する（ステップ１４０４）。

[0074] オブジェクト生成回数５０２が設定閾値以上でない場合（ステップ１４０２の結果が「ＮＯ」）、オブジェクト生成回数５０２に対するオブジェクトプロファイリング回数５０３が設定閾値以上でない場合（ステップ１４０３の結果が「ＮＯ」）、又は、ステップ１４０４の処理の終了後には、ステップ１４０１の処理に戻る。そして、未処理のオブジェクトIDに対して、本処理の実行を継続する。

[0075] オブジェクトID５０１に未処理のオブジェクトIDが存在しない場合には（ステップ１４０１の結果が「ＮＯ」）、本処理を終了する。なお、ステップ１４０２やステップ１４０３の判定では、当該オブジェクトIDに係るオブジェクトのクラスサイズなどに応じて、比較対象値に重み付けを行ってもよい。

[0076] 図１５は、本発明の実施の形態のメソッド実行処理の手順を示すフローチャートである。本処理は、プロセッサ１２９がメソッド実行部１１４を処理することによって、実行される。

バイトコード読み込み部１１１が生成した実行対象メソッドに係るバイトコード情報１１２は、一つ以上の命令を含み、各命令には、命令種別が付与されている。

[0077] プロセッサ１２９がメソッド実行部１１４を実行すると、メソッド実行部１１４は、まず、実行対象のメソッドの命令種別がオブジェクト生成命令であるか否かを判定する（ステップ１５０１）。

[0078] 命令種別がオブジェクト生成命令でない場合には（ステップ１５０１の結果が「ＮＯ」）、当該命令を実行する（ステップ１５０４）。

命令種別がオブジェクト生成命令である場合には（ステップ１５０１の結果が「ＹＥＳ」）、オブジェクト生成部１２１を実行し、オブジェクト生成処理を実行する（ステップ１５０２）。

[0079] ステップ１５０２の処理の終了後には、オブジェクトの生成に成功したか否かを判定する（ステップ１５０３）。

オブジェクトの生成に失敗した場合には（ステップ１５０３の結果が「Ｎ

Ｏ」）、メソッドの実行を中断し、メモリ不足エラーを発生させる（ステップ１５０５）。

[0080] オブジェクトの生成に成功した場合（ステップ１５０３の結果が「ＹＥＳ」）、又は、ステップ１５０４の処理の終了後には、次に実行すべき命令があるか否かを判定する（ステップ１５０６）。

次に実行すべき命令がある場合には（ステップ１５０６の結果が「ＹＥＳ」）、ステップ１５０１の処理を実行し、メソッド実行処理を継続する。

[0081] 次に実行すべき命令がない、つまり、すべての命令の処理が終了した場合には（ステップ１５０６の結果が「ＮＯ」）、又は、ステップ１５０５の処理の終了後には、本処理を終了する。

[0082] 図１６は、本発明の実施の形態のオブジェクト生成処理１５０２の手順を示すフローチャートである。本処理は、プロセッサ１２９がオブジェクト生成部１２１を処理することによって、実行される。

[0083] プロセッサ１２９がオブジェクト生成部１２１を実行すると、オブジェクト生成部１２１は、まず、生成を要求されたオブジェクトを生成するのに必要な量のメモリを当該オブジェクト生成命令に係るオブジェクト生成先４０４（図４参照）に格納されたオブジェクト生成先に確保可能であるか否かを判定する。オブジェクト生成先が外部ヒープ領域１１６であり、生成を要求されたオブジェクトを生成するのに必要な量のメモリを外部ヒープ領域１１６に確保可能でない場合には、さらに、生成を要求されたオブジェクトを生成するのに必要な量のメモリをJavaヒープ領域１１５に確保可能であるか否かを判定する。（ステップ１６０１）。

[0084] 生成を要求されたオブジェクトを生成するのに必要な量のメモリをオブジェクト生成先に確保できない場合には（ステップ１６０１の結果が「ＮＯ」）、Javaヒープ領域１１５の不要なメモリ領域を解放するために、GC部１２２を実行し、GC処理を実行する（ステップ１６０２）。

[0085] ステップ１６０２の処理の実行後、生成を要求されたオブジェクトを生成するのに必要な量のメモリをJavaヒープ領域１１５に確保可能であるか否か

を再び判定する（ステップ１６０３）。

[0086] 生成を要求されたオブジェクトを生成するのに必要な量のメモリをJavaヒープ領域１１５に確保できない場合には（ステップ１６０３の結果が「ＮＯ」）、オブジェクト生成処理の実行元にオブジェクト生成に失敗した旨を通知する（ステップ１６１１）。

[0087] 生成を要求されたオブジェクトを生成するのに必要な量のメモリをオブジェクト生成先に確保可能な場合には（ステップ１６０１、又は、ステップ１６０３の結果が「ＹＥＳ」）、今回のオブジェクト生成先がどこであるかを判定する（ステップ１６０４）。

[0088] 今回のオブジェクト生成先がJavaヒープ領域１１５である場合には（ステップ１６０４の結果が「Javaヒープ領域」）、JavaVM１１０の設定フラグを参照し、オブジェクトプロファイリングを行うか否かを判定する（ステップ１６０６）。

[0089] オブジェクトプロファイリングを行う場合には（ステップ１６０６の結果が「ＹＥＳ」）、オブジェクトプロファイリング用オブジェクトをJavaヒープ領域１１５に生成し、当該オブジェクト生成命令によって生成されたオブジェクトであることを識別するための当該オブジェクト生成命令に係るオブジェクトID４０２を当該オブジェクトのオブジェクトIDフィールド７０２に記録する（ステップ１６０７）。

[0090] ステップ１６０７の処理の終了後には、当該オブジェクト生成命令によってオブジェクトを生成したことを記録するために、当該オブジェクト生成命令に係るオブジェクト生成回数５０２の値に１を加算して、オブジェクトプロファイリング情報テーブル３０２を更新する（ステップ１６０９）。

[0091] 今回のオブジェクト生成先が外部ヒープ領域１１６である場合には（ステップ１６０４の結果が「外部ヒープ領域」）、オブジェクト生成先となる外部ヒープ１１７を生成する。そして、外部ヒープ１１７の外部ヒープヘッダ９０１に、当該生成命令に係るメンバオブジェクト情報アドレス５０４を記録する（ステップ１６０５）。なお、外部ヒープを生成する際の初期サイズ

は、当該生成命令に係る起点オブジェクトのクラスサイズ、及び、当該生成命令に係り、メンバオブジェクト適合率が設定閾値以上であるメンバオブジェクトのクラスサイズの合計に基づき設定される。

[0092] オブジェクトプロファイリングを行わない場合（ステップ１６０６の結果が「ＮＯ」）、又は、ステップ１６０５の処理の終了後には、オブジェクトIDフィールド７０２を含まない、通常オブジェクトをオブジェクト生成先に生成する（ステップ１６０８）。

[0093] ステップ１６０８の処理の終了後、又は、ステップ１６０９の処理の終了後には、オブジェクト生成処理の実行元に生成したオブジェクトを通知する（ステップ１６１０）。

ステップ１６１０の処理の終了後、又は、ステップ１６１１の処理の終了後には、本処理を終了する。

[0094] なお、オブジェクト生成先やオブジェクトプロファイリングを行うか否かに応じた専用のオブジェクト生成命令と、それに対するオブジェクト生成処理を用意し、バイトコード情報１１２のオブジェクト生成命令を適宜書き換えることで、所定のオブジェクト生成処理を実行するようにしてもよい。その場合、例えば、ステップ１６０４やステップ１６０７の判定が不要になることもある。

[0095] 図１７は、本発明の実施の形態のGC処理１６０２の手順を示すフローチャートである。本処理は、プロセッサ１２９がGC部１２２を処理することによって、実行される。

プロセッサ１２９がGC部１２２を実行すると、GC部１２２は、まず、オブジェクト生成処理１５０２で枯渇している領域がJavaヒープ領域１１５のNew領域とOld領域のどちらかを判定する（ステップ１７０１）。

[0096] オブジェクト生成処理１５０２で枯渇している領域がJavaヒープ領域１１５のNew領域である場合には（ステップ１７０１の結果が「New領域」）、CopyGC部１２４を実行し、CopyGC処理を実行する（ステップ１７０２）。

[0097] オブジェクト生成処理１５０２で枯渇している領域がJavaヒープ領域１１

５のOld領域である場合には（ステップ１７０１の結果が「Old領域」）、FullGC部１２３を実行し、FullGC処理を実行する（ステップ１７０３）。

ステップ１７０２の処理の終了後、又は、ステップ１７０３の処理の終了後には、本処理を終了する。

[0098] 図１８は、本発明の実施の形態のFullGC処理１７０３の手順を示すフローチャートである。本処理は、プロセッサ１２９がFullGC部１２３を処理することによって、実行される。

[0099] プロセッサ１２９がFullGC部１２３を実行すると、FullGC部１２３は、まず、FullGC対象の全オブジェクトに対して、オブジェクトヘッダ７０１にオブジェクト種別“未解析”を設定する。また、以降のプログラムの実行に必要なオブジェクトであることを示すために、レジスタ、実行時スタック、グローバル変数といったルートセットからオブジェクト参照を順次辿ることによって到達可能なオブジェクトに対して、オブジェクトヘッダ７０１にオブジェクト種別“必要”を設定する。そして、ルートセットから到達可能なオブジェクトを全て辿った後で、オブジェクトヘッダ７０１のオブジェクト種別が“必要”であるオブジェクトを以降のプログラムの実行に必要なオブジェクト、オブジェクトヘッダ７０１のオブジェクト種別が“未解析”であるオブジェクトを以降のプログラムの実行に不要なオブジェクトとして、オブジェクトの必要・不要を判定する（ステップ１８０１）。

[0100] 図１０は、本発明の実施の形態のFullGC処理１７０３のステップ１８０１の処理の終了時点におけるJavaヒープ領域１１５のOld領域に格納されたオブジェクトの状態を示す図である。

[0101] 各ノードはオブジェクトを示している。二重丸で示すオブジェクトは、FullGC処理１７０３のステップ１８０１の処理によってオブジェクトヘッダ７０１にオブジェクト種別“必要”が設定されたオブジェクトである。なお、ノード内の記号はクラス名、クラス名の右下の数字はオブジェクトIDフィールド７０２に格納されたオブジェクトIDを示している。

[0102] また、オブジェクトを連結する矢印は、オブジェクトの参照関係を示して

いる。実線の矢印は、FullGC処理１７０３のステップ１８０１の処理によって辿られたオブジェクト参照であり、破線の矢印は、オブジェクト参照は存在するが、FullGC処理１７０３のステップ１８０１の処理によって辿られることのなかったオブジェクト参照を示している。

[0103] ルートセット１００１は、レジスタ、実行時スタック、グローバル変数などである。ルートセット１００１に保持されるオブジェクト参照を起点としてFullGC処理１７０３のステップ１８０１の処理が実行されると、ルートセット１００１からオブジェクト参照を順次辿ることによって到達可能なオブジェクトは全て必要オブジェクトと判定される。したがって、図１０に示すように、オブジェクト１００２、オブジェクト１００３、オブジェクト１００４、オブジェクト１００５などの二重丸で示すオブジェクトは必要オブジェクトである。一方、ルートセット１００１からオブジェクト参照を順次辿ることによって到達することのできなかったオブジェクトは全て未解析オブジェクト（不要オブジェクト）と判定される。したがって、図１０に示すように、オブジェクト１００６、オブジェクト１００７、オブジェクト１００８などの一重丸で示すオブジェクトは未解析オブジェクトである。これら未解析オブジェクトは、FullGC処理１７０３によって実際に解放されるオブジェクトであるが、Javaヒープ１１５のOld領域の使用量を抑え、FullGC処理１７０３の発生を抑止するためには、外部ヒープ領域１１６に格納し、GCによらず解放することが望ましい。

[0104] また、未解析オブジェクトの参照関係において参照関係のルートに当たるオブジェクト１００６、オブジェクト１００７、オブジェクト１００８などの起点オブジェクトは、これらオブジェクトがルートセット１００１や必要オブジェクトから参照されなくなったことによって、起点オブジェクトからオブジェクトを順次辿ることによってのみ到達可能なオブジェクトも一緒に解放される。そのため、起点オブジェクトを含め、起点オブジェクトからオブジェクトを順次辿ることによって到達可能なオブジェクトを外部ヒープ領域１１６に格納するようにすれば、Javaヒープ１１５のOld使用量をより多く

抑えることができる。そのためには、起点オブジェクトを特定し、それら起点オブジェクトを外部ヒープ領域１１６に生成するようにする必要がある。この起点オブジェクトの特定方法については、図１１で説明する。

[0105] ここで、図１８のFullGC処理１７０３のフローチャートの説明に戻る。

ステップ１８０１の処理の終了後には、JavaVM１１０の設定フラグを参照し、オブジェクトプロファイリングを行うか否かを判定する（ステップ１８０２）。

[0106] オブジェクトプロファイリングを行う場合には（ステップ１８０２の結果が「ＹＥＳ」）、オブジェクト解析部１２５を実行し、オブジェクトプロファイリング処理を実行する（ステップ１８０３）。

[0107] オブジェクトプロファイリングを行わない場合（ステップ１８０２の結果が「ＮＯ」）、又は、ステップ１８０３の処理の終了後には、オブジェクトヘッダ７０１のオブジェクト種別が“必要”でないオブジェクトを不要なオブジェクトとして解放する（ステップ１８０４）。

ステップ１８０４の処理の終了後には、本処理を終了する。

[0108] 図１９は、本発明の実施の形態のオブジェクトプロファイリング処理１８０３の手順を示すフローチャートである。本処理は、プロセッサ１２９がオブジェクト解析部１２５を処理することによって、実行される。

本処理のステップ１９０１からステップ１９０４までの処理を実行することによって、起点オブジェクトを特定することができる。

[0109] 本処理を実行すると、まず、ステップ１８０１で不要なオブジェクトと判定された全オブジェクトについて、未処理の不要オブジェクトが存在するか否かを判定する（ステップ１９０１）。

[0110] 未処理の不要オブジェクトが存在する場合には（ステップ１９０１の結果が「ＹＥＳ」）、当該オブジェクトのオブジェクトヘッダ７０１のオブジェクト種別を参照し、オブジェクト種別が“未解析”であるか否かを判定する（ステップ１９０２）。

[0111] オブジェクト種別が“未解析”である場合には（ステップ１９０２の結果

が「ＹＥＳ」）、当該オブジェクトのオブジェクトヘッダ７０１のオブジェクト種別を“起点”に設定する（ステップ１９０３）。

[0112] ステップ１９０３の処理の終了後には、ステップ１９０３の起点オブジェクトに対して、メンバオブジェクト処理を呼び出し、メンバオブジェクト処理を実行する（ステップ１９０４）。

[0113] オブジェクト種別が“未解析”でない場合（ステップ１９０２の結果が「ＮＯ」）、又は、ステップ１９０４の処理の終了後には、ステップ１９０１の処理に戻る。そして、未処理の不要オブジェクトに対して、本処理の実行を継続する。

[0114] 未処理の不要オブジェクトが存在しない場合には（ステップ１９０１の結果が「ＮＯ」）、オブジェクトヘッダ７０１のオブジェクト種別が“起点”である起点オブジェクトについて、当該起点オブジェクトのオブジェクトIDフィールド７０２を参照し、当該起点オブジェクトのオブジェクトIDを取得する。そして、当該オブジェクトIDに係るオブジェクトプロファイリング回数５０３の値に１を加算して、オブジェクトプロファイリング情報テーブル３０２を更新する。また、当該オブジェクトIDに係るメンバオブジェクト情報アドレス５０４に格納してあるポインタにより当該起点オブジェクトに係るメンバオブジェクト情報３０３を参照する。なお、当該起点オブジェクトに係るメンバオブジェクト情報３０３が存在しない場合には、当該起点オブジェクトに係るメンバオブジェクト情報３０３を生成する。そして、当該起点オブジェクトと当該起点オブジェクトからオブジェクト参照を順次辿ることによって到達可能なメンバオブジェクトの参照関係に応じて、メンバオブジェクトデータ構造を当該メンバオブジェクト情報３０３に適宜追加し、該当するメンバオブジェクトデータ構造に格納されている、メンバオブジェクトプロファイリング回数に１を加算して、メンバオブジェクト情報３０３を更新する（ステップ１９０５）。

ステップ１９０５の処理の終了後には、本処理を終了する。

[0115] 図２０は、本発明の実施の形態のメンバオブジェクト処理１９０４の手順

を示すフローチャートである。本処理は、オブジェクトプロファイリング処理１８０３から呼び出されることやメンバオブジェクト処理１９０４から再帰的に呼び出されることによって、実行される。

[0116] 本処理を実行すると、まず、本処理の処理対象となったオブジェクトについて、未処理の参照先オブジェクトが存在するか否かを判定する（ステップ２００１）。

未処理の参照先オブジェクトが存在する場合には（ステップ２００１の結果が「ＹＥＳ」）、参照先オブジェクトのオブジェクトヘッダ７０１のオブジェクト種別を参照し、オブジェクト種別を判定する（ステップ２００２）。

[0117] オブジェクト種別が“起点”である場合には（ステップ２００２の結果が「起点」）、参照先オブジェクトがメンバオブジェクト処理１８０３の呼び出し元の起点オブジェクトと同じであるか否かを判定する（ステップ２００３）。

参照先オブジェクトがメンバオブジェクト処理１８０３の呼び出し元の起点オブジェクトと同じでない場合には（ステップ２００３の結果が「ＮＯ」）、参照先オブジェクトのオブジェクトヘッダ７０１のオブジェクト種別を“メンバ”に設定する（ステップ２００５）。

[0118] オブジェクト種別が“未解析”である場合には（ステップ２００２の結果が「未解析」）、参照先オブジェクトのオブジェクトヘッダ７０１のオブジェクト種別を“メンバ”に設定する（ステップ２００４）。

ステップ２００４の処理の終了後には、当該参照先オブジェクトに対して、メンバオブジェクト処理１９０４を実行する（ステップ１９０４）。

[0119] オブジェクト種別が“メンバ”である場合（ステップ２００２の結果が「メンバ」）、参照先オブジェクトがメンバオブジェクト処理１８０３の呼び出し元の起点オブジェクトと同じである場合（ステップ２００３の結果が「ＹＥＳ」）、ステップ２００５の処理の終了後、又は、ステップ２００４の後に実行するステップ１９０４の処理の終了後には、ステップ２００１の処

理に戻る。そして、未処理の参照先オブジェクトに対して（ステップ２００１の結果が「ＹＥＳ」）、本処理の実行を継続する。

未処理の参照先オブジェクトが存在しない場合には（ステップ２００１の結果が「ＮＯ」）、本処理を終了する。

[0120] 図１１は、本発明の実施の形態のオブジェクトプロファイリング処理１８０３のステップ１９０１からステップ１９０４までの処理を実行することによって、起点オブジェクトを特定する具体的な手順を示す図である。

[0121] 各ノードはオブジェクトを示している。細線の一重丸で示すオブジェクトは、FullGC処理１７０３のステップ１８０１の処理によってオブジェクトヘッダ７０１にオブジェクト種別“未解析”が設定されたままの不要オブジェクトである。なお、ノード内の記号はクラス名、クラス名の右下の数字はオブジェクトIDフィールド７０２に格納されたオブジェクトIDを示している。

[0122] また、オブジェクトを連結する矢印は、オブジェクトの参照関係を示している。実線の矢印は、オブジェクトプロファイリング処理１８０３のステップ１９０１からステップ１９０４までの処理を実行することによって辿られたオブジェクト参照であり、破線の矢印は、オブジェクト参照は存在するが、オブジェクトプロファイリング処理１８０３のステップ１９０１からステップ１９０４までの処理を実行することによってまだ辿られていないオブジェクト参照を示している。

[0123] 図１１(a)は、FullGC処理１７０３のステップ１８０１の処理の終了時点の状態を示している。なお、オブジェクトの参照関係が判明していない状態から、オブジェクト１００９、オブジェクト１０１１、オブジェクト１００８、オブジェクト１０１０、オブジェクト１０１２の順にオブジェクトを処理していくものとする。

[0124] 図１１(b)は、オブジェクト１００９を処理し、オブジェクトプロファイリング処理１８０３のステップ１９０２の判定においてオブジェクト種別が“未解析”であるため、オブジェクト１００９のオブジェクトヘッダ７０１のオブジェクト種別を“起点”に設定した状態を示している。その結果、起点

オブジェクトであるオブジェクト１００９は太線の一重丸で示す。

[0125] 図１１(c)は、オブジェクト１００９の参照先であるオブジェクト１０１１を処理し、メンバオブジェクト処理１９０４のステップ２００２の判定においてオブジェクト種別が“未解析”であるため、オブジェクト１０１１のオブジェクトヘッダ７０１のオブジェクト種別を“メンバ”に設定した状態を示す。その結果、メンバオブジェクトであるオブジェクト１０１１は×を付した一重丸で示す。オブジェクト１０１１の参照先オブジェクトは存在しないため、オブジェクト１００９の処理はここで終了となる。

また、オブジェクト１０１１を処理し、オブジェクトプロファイリング処理１８０３のステップ１９０２の判定においてオブジェクト種別が“未解析”でないため、オブジェクト１０１１の処理はここで終了となる。

[0126] 図１１(d)は、オブジェクト１００８を処理し、オブジェクトプロファイリング処理１８０３のステップ１９０２の判定においてオブジェクト種別が“未解析”であるため、オブジェクト１００８のオブジェクトヘッダ７０１のオブジェクト種別を“起点”に設定した状態を示している。

[0127] 図１１(e)は、オブジェクト１００８の参照先であるオブジェクト１００９を処理し、メンバオブジェクト処理１９０４のステップ２００２の判定においてオブジェクト種別が“起点”であり、メンバオブジェクト処理１９０４のステップ２００３の判定において、参照先であるオブジェクト１００９と呼び出し元のオブジェクト１００８は同じでないため、参照先であるオブジェクト１００９のオブジェクトヘッダ７０１のオブジェクト種別を“メンバ”に設定した状態を示している。

また、オブジェクト１００８の参照先であるオブジェクト１０１０を処理し、メンバオブジェクト処理１９０４のステップ２００２の判定においてオブジェクト種別が“未解析”であるため、参照先であるオブジェクト１０１０のオブジェクトヘッダ７０１のオブジェクト種別を“メンバ”に設定した状態を示している。さらに、オブジェクト１０１０の参照先であるオブジェクト１０１２を処理し、メンバオブジェクト処理１９０４のステップ２００

２の判定においてオブジェクト種別が“未解析”であるため、参照先であるオブジェクト１０１２のオブジェクトヘッダ７０１にオブジェクト種別“メンバ”を設定した状態を示している。なお、オブジェクト１０１２の参照先であるオブジェクト１００８は、メンバオブジェクト処理１９０４のステップ２００２の判定においてオブジェクト種別が“起点”であり、メンバオブジェクト処理１９０４のステップ２００３の判定において、呼び出し元のオブジェクト１００８と同じであるため、オブジェクト１００８の処理はここで終了となる。

さらに、オブジェクト１０１０を処理し、オブジェクトプロファイリング処理１８０３のステップ１９０２の判定においてオブジェクト種別が“未解析”でないため、オブジェクト１０１０の処理はここで終了となる。オブジェクト１０１２についても同様である。

[0128] 図１１(f)は、オブジェクト１００９、オブジェクト１０１１、オブジェクト１００８、オブジェクト１０１０、オブジェクト１０１２の処理が全て終了した状態を示しており、このオブジェクトの参照関係の解析により、オブジェクト１００８が起点オブジェクトであると特定することができる。また、オブジェクト１００９、オブジェクト１０１１、オブジェクト１０１０、及び、オブジェクト１０１２はメンバオブジェクトであり、メンバオブジェクトから起点オブジェクトへのオブジェクト参照１０１３は、存在しないものとして処理する。

[0129] 図２１は、本発明の実施の形態のCopyGC処理１７０２の手順を示すフローチャートである。本処理は、プロセッサ１２９がCopyGC部１２４を処理することによって、実行される。

[0130] プロセッサ１２９がCopyGC部１２４を実行すると、CopyGC部１２４は、まず、New領域に格納してある以降のプログラム実行に必要なオブジェクトをNew領域内の別の領域に順次移動させ、移動されることのなかったオブジェクトを不要なオブジェクトとして解放する（ステップ２１０１）。なお、ステップ２１０１の処理において、外部ヒープ領域１１６に格納されている起点オ

ブジェクトからオブジェクト参照を順次辿ることによって到達可能なオブジェクトについては、起点オブジェクトに対して、メンバオブジェクト移動処理を呼び出し、メンバオブジェクト移動処理を実行することで、適切な移動先に移動させる。メンバオブジェクト移動処理の詳細は、図２２で説明する。

[0131] ステップ２１０１の処理の終了後には、例えば、外部ヒープ１１７へのオブジェクト参照を検査し、どこからも参照されていない不要な外部ヒープ１１７が存在するか否かを判定する（ステップ２１０２）。

不要な外部ヒープ１１７が存在する場合には（ステップ２１０２の結果が「ＹＥＳ」）、不要な外部ヒープ領域１１６、例えば、外部ヒープ１や外部ヒープ３を解放することで、外部ヒープ１や外部ヒープ３に格納してある不要なオブジェクトを解放する（ステップ２１０３）。

[0132] 不要な外部ヒープ１１７が存在しない場合（ステップ２１０２の結果が「ＮＯ」）、又は、ステップ２１０３の処理の終了後には、JavaVM１１０の設定フラグを参照し、CopyGC処理時にオブジェクト生成命令のオブジェクト生成先を変更するか否かを判定する（ステップ２１０４）。

オブジェクト生成先を変更する場合には（ステップ２１０４の結果が「ＹＥＳ」）、オブジェクト生成情報テーブル３０１に対して、オブジェクト生成先変更部１１３を実行し、オブジェクト生成先変更処理を実行する（ステップ１３０３）。

オブジェクト生成先を変更しない場合（ステップ２１０４の結果が「ＮＯ」）、又は、ステップ１３０３の処理の終了後には、本処理を終了する。

[0133] なお、本発明の実施の形態では、CopyGC処理時に不要な外部ヒープの解放とオブジェクト生成命令のオブジェクト生成先の変更を実施しているが、CopyGC処理時以外の所定のタイミングでこれらの処理を実施してもよい。

[0134] 図２２は、本発明の実施の形態のメンバオブジェクト移動処理の手順を示すフローチャートである。本処理は、CopyGC処理１７０２のステップ２１０１の処理において、実行される。

[0135] 本処理を実行すると、まず、本処理の処理対象となったオブジェクトについて、未処理の参照先オブジェクトが存在するか否かを判定する（ステップ２２０１）。

未処理の参照先オブジェクトが存在する場合には（ステップ２２０１の結果が「ＹＥＳ」）、参照先オブジェクトがCopyGC処理１７０２のステップ２１０１の処理によるNew領域からOld領域への移動対象であるか否かを判定する（ステップ２２０２）。なお、参照先オブジェクトがCopyGC処理によるNew領域からOld領域への移動対象であるとは、少なくともその参照先オブジェクトがOld領域に移動するのに十分な回数のCopyGCを経ている長命なオブジェクトであることを意味する。

[0136] 参照先オブジェクトがNew領域からOld領域への移動対象である場合には（ステップ２２０２の結果が「ＹＥＳ」）、参照先オブジェクトのメンバオブジェクト適合率が、JavaVM１１０の設定閾値（第３の閾値）以上であるか否かを判定する（ステップ２２０３）。なお、この判定では、メンバオブジェクトのクラスサイズなどに応じて、メンバオブジェクト適合率に重み付けを行ってもよい。

[0137] 参照先オブジェクトのメンバオブジェクト適合率が、設定閾値以上である場合には（ステップ２２０３の結果が「ＹＥＳ」）、参照先オブジェクトを起点オブジェクトと同じ外部ヒープに移動する（ステップ２２０４）。

[0138] 図１２は、本発明の実施の形態のメンバオブジェクト移動処理の(a)開始時点、及び、(b)終了時点における外部ヒープ１、外部ヒープ２、及び、Javaヒープ１１５のNew領域とOld領域に格納されたオブジェクトの状態を示す図である。

各ノードはオブジェクトを示している。なお、ノード内の記号はクラス名の略称を示している。例えば、ノード内の記号が“A”であるオブジェクト１２０１のクラス名は“ClassA”になる。また、オブジェクトを連結する矢印は、オブジェクトの参照関係を示している。

[0139] ここで、オブジェクト１２０１は、外部ヒープ１に格納された起点オブジ

ェクトである。また、オブジェクト１２０８は、外部ヒープ２に格納された起点オブジェクトである。オブジェクト１２０１が格納された外部ヒープ１の外部ヒープヘッダ９０１には、オブジェクト１２０１に係るメンバオブジェクト情報３０３へのポインタが格納してあり、このポインタによりオブジェクト１２０１に係るメンバオブジェクト情報３０３を参照する。同様に、オブジェクト１２０８が格納された外部ヒープ２の外部ヒープヘッダ９０１には、オブジェクト１２０８に係るメンバオブジェクト情報３０３へのポインタが格納してあり、このポインタによりオブジェクト１２０８に係るメンバオブジェクト情報３０３を参照する。なお、ここでは、オブジェクト１２０１に係るメンバオブジェクト情報３０３は図６に示してあるメンバオブジェクト情報３０３であるものとする。このメンバオブジェクト情報３０３の起点オブジェクトデータ構造６０１を参照して、ClassAの起点オブジェクトはFullGCによってOld領域で４回解放されたことがわかる。また、メンバオブジェクト情報３０３のメンバオブジェクトデータ構造６０６を参照して、ClassAの起点オブジェクトがFullGCによって解放されたことによってClassFのメンバオブジェクトがOld領域で３回解放されたこと、メンバオブジェクト情報３０３のメンバオブジェクトデータ構造６０５を参照して、ClassAの起点オブジェクトがFullGCによって解放されたことによってClassEのメンバオブジェクトがOld領域で２回解放されたことなどがわかる。

[0140] ここで、ClassB、ClassC、ClassD、ClassE、ClassF、及び、ClassGのメンバオブジェクト適合率は、それぞれ、1.00、1.00、1.00、0.50、0.75、0.25である。メンバオブジェクト適合率が高いメンバオブジェクトは、起点オブジェクトが不要になると同時に不要になる可能性が高いので、メンバオブジェクト適合率が高いメンバオブジェクトを起点オブジェクトと同じ外部ヒープに格納するようにすることによって、起点オブジェクトが不要になると同時に、その外部ヒープを解放することによって、メンバオブジェクト適合率が高いメンバオブジェクトを解放できる可能性も高くなる。そのため、メンバオブジェクト移動処理では、メンバオブジェクト適合率の高いメンバオブ

ジェクトを起点オブジェクトと同じ外部ヒープに移動するようにする。例えば、メンバオブジェクト適合率の設定閾値が0.60の場合、メンバオブジェクト適合率が0.60以上であるオブジェクト１２０２、オブジェクト１２０３、オブジェクト１２０４、及び、オブジェクト１２０６は、メンバオブジェクト移動処理によって、オブジェクト１２０１と同じ外部ヒープ１に移動する。一方、メンバオブジェクト適合率が0.60以上でないオブジェクト１２０５は、メンバオブジェクト移動処理によって、ステップ２２０３の結果が「ＹＥＳ」となる別の起点オブジェクトであるオブジェクト１２０８と同じ外部ヒープ２に移動する。また、オブジェクト１２０７は、外部ヒープ領域１１６に格納された起点オブジェクトからオブジェクト参照を順次辿ることによって到達可能なオブジェクトであるが、メンバオブジェクト適合率が0.60以上ではないので、Javaヒープ領域１１５のOld領域に移動する。

[0141] ここで、図２２のメンバオブジェクト移動処理のフローチャートの説明に戻る。

ステップ２２０４の処理の終了後には、参照先オブジェクトに対して、メンバオブジェクト移動処理を呼び出し、メンバオブジェクト移動処理を実行する（ステップ２２０５）。

[0142] 参照先オブジェクトがNew領域からOld領域への移動対象でない場合（ステップ２２０２の結果が「ＮＯ」）、参照先オブジェクトのメンバオブジェクト適合率が、設定閾値以上でない場合（ステップ２２０３の結果が「ＮＯ」）、及び、ステップ２２０５の処理の終了後には、ステップ２２０１の処理に戻る。そして、未処理の参照先オブジェクトに対して、本処理の実行を継続する。

未処理の参照先オブジェクトが存在しない場合には（ステップ２２０１の結果が「ＮＯ」）、本処理を終了する。

[0143] なお、このメンバオブジェクト移動処理において、ステップ２２０２の処理は省略してもよい。つまり、参照先オブジェクト（メンバオブジェクト）を起点オブジェクトが存在する外部ヒープと同じ外部ヒープに移動するか否

かの判断は、ステップ２２０３の判定だけで行ってもよい。換言すれば、場合によっては、参照先オブジェクトがOld領域へ移るほど長命でなくても、参照先オブジェクトを起点オブジェクトと同じ外部ヒープに移動するように処理してGCの発生を抑えるほうが好都合である。ただ、一概にはいえないが、図２２の処理のように、ステップ２２０２の判定を設けて、より長命な参照先オブジェクトを起点オブジェクトと同じ外部ヒープに移動することは効果的である。

[0144] 本発明の実施の形態では、例えば図２３のネットワーク構成に示すように、複数の計算機システム１０１とウェブサーバ２０１とがネットワーク２０３を介して通信可能に接続している。計算機システム１０１は、Javaソースファイル１０４の記述内容に従って機能するJavaアプリケーション１３２を有しており、JavaVM１１０がJavaアプリケーション１３２を実行することで所定のサービスを実現する。

[0145] 一方、ウェブサーバ２０１は、入力装置、出力装置、プロセッサ、記憶装置といったハードウェア資源を有するコンピュータである。このウェブサーバ２０１は、例えばJavaサーブレット２０２を有している。ウェブサーバ２０１は、Javaサーブレット２０２を機能させることにより、動的にウェブページを生成し、そのウェブページをリクエストのあった計算機システム１０１に送信する。

[0146] なお、計算機システム１０１は、Javaアプリケーション１３２ではなく、Javaアプレットを有していてもよい。この場合、ウェブサーバ２０１が提供するウェブページを閲覧するウェブブラウザを用いてJavaアプレットを実行することで所定のサービスを実現する。

また、計算機システム１０１が有するJavaVM１１０は、予め出荷時に搭載されたものであってもよいし、ウェブサーバ２０１から取得したものであってもよい。

[0147] ≪その他≫

前記した各実施形態は、本発明を実施するために好適のものであるが、そ

の実施形式はこれらに限定されるものでなく、本発明の要旨を変更しない範囲内において種々変形することが可能である。

[0148] 例えば、本実施形態で取り扱ったGCは、世代別GCであったが、本発明が適用可能なGCはこれに限らない。例えばパラレルGCに対しても適用可能である。

[0149] また、オブジェクトプロファイリング情報テーブル３０２において登録するオブジェクトプロファイリング回数５０３は、FullGCの回数でなく、CopyGCの回数であったり、他のGCの回数であったりしてもよい。

[0150] その他、ハードウェア、ソフトウェア、各フローチャート等の具体的な構成について、本発明の趣旨を逸脱しない範囲で適宜変更が可能である。

符号の説明

[0151] １０１ 計算機システム（情報処理装置）
１０２ 補助記憶（記憶部）
１０３ Javaクラスファイル
１０４ Javaソースファイル
１０５ オブジェクト解析情報ファイル
１０６ 設定ファイル
１０７ 主記憶（記憶部）
１０８ データ領域
１０９ Javaコンパイラ
１１０ JavaVM
１１１ バイトコード読み込み部
１１２ バイトコード情報
１１３ オブジェクト生成先変更部
１１４ メソッド実行部
１１５ Javaヒープ領域（第１のメモリ領域）
１１６ 外部ヒープ領域（第２のメモリ領域）
１１７ 外部ヒープ１

１１８　外部ヒープ２

１１９　外部ヒープ３

１２０　メモリ管理部

１２１　オブジェクト生成部

１２２　GC部

１２３　FullGC部

１２４　CopyGC部

１２５　オブジェクト解析部

１２６　オブジェクト解析情報

１２７　オペレーティングシステム

１２８　バス

１２９　プロセッサ（制御部）

１３０　入力装置

１３１　出力装置

１３２　Javaアプリケーション

２０１　ウェブサーバ

２０２　Javaサーブレット

２０３　ネットワーク

３０１　オブジェクト生成情報テーブル

３０２　オブジェクトプロファイリング情報テーブル

３０３　メンバオブジェクト情報

請求の範囲

[請求項1] プログラムを実行するために確保したメモリ領域において、不要になったメモリ領域は自動的に解放する自動解放機能を備えた仮想マシンが稼働する情報処理装置におけるメモリ管理方法であって、

前記情報処理装置の記憶部は、

前記メモリ領域として、前記自動解放機能の対象となる第１のメモリ領域と、

前記メモリ領域として、前記自動解放機能の対象外となる第２のメモリ領域と、を有し、

プログラムを実行するときに生成されるオブジェクトごとに、当該オブジェクトに係るオブジェクト生成命令と、前記オブジェクト生成命令により当該オブジェクトを前記第１のメモリ領域に生成した回数を示すオブジェクト生成回数と、前記自動解放機能により前記第１のメモリ領域を解放した回数を示すオブジェクトプロファイリング回数と、を含むオブジェクト解析情報を記憶しており、

前記情報処理装置の制御部は、

前記オブジェクト解析情報を参照して、前記オブジェクトのオブジェクト生成回数が、前記記憶部に記憶されている第１の閾値以上であるか否かを判定するステップと、

前記オブジェクト生成回数が前記第１の閾値以上であるとき、前記オブジェクト生成回数に対する前記オブジェクトプロファイリング回数の割合が、前記記憶部に記憶されている第２の閾値以上であるか否かを判定するステップと、

前記割合が前記第２の閾値以上であるとき、前記オブジェクト生成命令によるオブジェクトの生成先を前記第２のメモリ領域に変更するステップと、を実行する

ことを特徴とするメモリ管理方法。

[請求項2] 前記情報処理装置の制御部は、

前記第１のメモリ領域に生成されたオブジェクトのうち、前記仮想マシンが直接参照可能なオブジェクト、および前記直接参照可能なオブジェクトが直接的または間接的に参照するオブジェクトを除くオブジェクトを特定するステップと、

前記オブジェクト解析情報を参照して、前記特定したオブジェクトのオブジェクト生成回数が前記第１の閾値以上であり、当該オブジェクト生成回数に対する前記オブジェクトプロファイリング回数の割合が前記第２の閾値以上であるとき、前記オブジェクト生成命令によるオブジェクトの生成先を前記第２のメモリ領域に変更するステップと、を実行する

ことを特徴とする請求の範囲第１項に記載のメモリ管理方法。

[請求項3] 前記情報処理装置の制御部は、

前記特定したオブジェクトにおいて、前記仮想マシンから直接参照されず、並びにいずれのオブジェクトからも参照されないオブジェクトである起点オブジェクト、および前記起点オブジェクトが直接的または間接的に参照するオブジェクトであるメンバオブジェクトを設定するステップと、

前記起点オブジェクトのオブジェクトプロファイリング回数に対する前記メンバオブジェクトのオブジェクトプロファイリング回数の割合が、前記記憶部に記憶されている第３の閾値以上であるか否かを判定するステップと、

前記割合が前記第３の閾値以上であるとき、前記メンバオブジェクトも前記第２のメモリ領域に移動するステップと、を実行する

ことを特徴とする請求の範囲第２項に記載のメモリ管理方法。

[請求項4] 前記自動解放機能は、世代別ガベージコレクションによる自動解放機能であり、

前記第１のメモリ領域は、長命でないオブジェクトが格納されるNew領域と、長命なオブジェクトが格納されるOld領域とを含んで構成さ

れる

ことを特徴とする請求の範囲第１項から第３項のいずれか一項に記載のメモリ管理方法。

[請求項5] 前記オブジェクトプロファイリング回数が示す前記第１のメモリ領域の解放は、前記New領域および前記Old領域の両方を含む領域を対象にする解放である

ことを特徴とする請求の範囲第４項に記載のメモリ管理方法。

[請求項6] プログラムを実行するために確保したメモリ領域において、不要になったメモリ領域は自動的に解放する自動解放機能を備えた仮想マシンが稼働する情報処理装置として機能させるメモリ管理プログラムであって、

前記情報処理装置の記憶部は、

前記メモリ領域として、前記自動解放機能の対象となる第１のメモリ領域と、

前記メモリ領域として、前記自動解放機能の対象外となる第２のメモリ領域と、を有し、

プログラムを実行するときに生成されるオブジェクトごとに、当該オブジェクトに係るオブジェクト生成命令と、前記オブジェクト生成命令により当該オブジェクトを前記第１のメモリ領域に生成した回数を示すオブジェクト生成回数と、前記自動解放機能により前記第１のメモリ領域を解放した回数を示すオブジェクトプロファイリング回数と、を含むオブジェクト解析情報を記憶しており、

前記メモリ管理プログラムは、前記情報処理装置の制御部に、

前記オブジェクト解析情報を参照して、前記オブジェクトのオブジェクト生成回数が、前記記憶部に記憶されている第１の閾値以上であるか否かを判定する処理と、

前記オブジェクト生成回数が前記第１の閾値以上であるとき、前記オブジェクト生成回数に対する前記オブジェクトプロファイリング回

数の割合が、前記記憶部に記憶されている第２の閾値以上であるか否かを判定する処理と、

前記割合が前記第２の閾値以上であるとき、前記オブジェクト生成命令によるオブジェクトの生成先を前記第２のメモリ領域に変更する処理と、を実行させる

ことを特徴とするメモリ管理プログラム。

[請求項7] プログラムを実行するために確保したメモリ領域において、不要になったメモリ領域は自動的に解放する自動解放機能を備えた仮想マシンが稼働する情報処理装置であって、

前記メモリ領域として、前記自動解放機能の対象となる第１のメモリ領域と、

前記メモリ領域として、前記自動解放機能の対象外となる第２のメモリ領域と、を有し、

プログラムを実行するときに生成されるオブジェクトごとに、当該オブジェクトに係るオブジェクト生成命令と、前記オブジェクト生成命令により当該オブジェクトを前記第１のメモリ領域に生成した回数を示すオブジェクト生成回数と、前記自動解放機能により前記第１のメモリ領域を解放した回数を示すオブジェクトプロファイリング回数と、を含むオブジェクト解析情報を記憶している記憶部と、

前記オブジェクト解析情報を参照して、前記オブジェクトのオブジェクト生成回数が、前記記憶部に記憶されている第１の閾値以上であるか否かを判定する制御と、

前記オブジェクト生成回数が前記第１の閾値以上であるとき、前記オブジェクト生成回数に対する前記オブジェクトプロファイリング回数の割合が、前記記憶部に記憶されている第２の閾値以上であるか否かを判定する制御と、

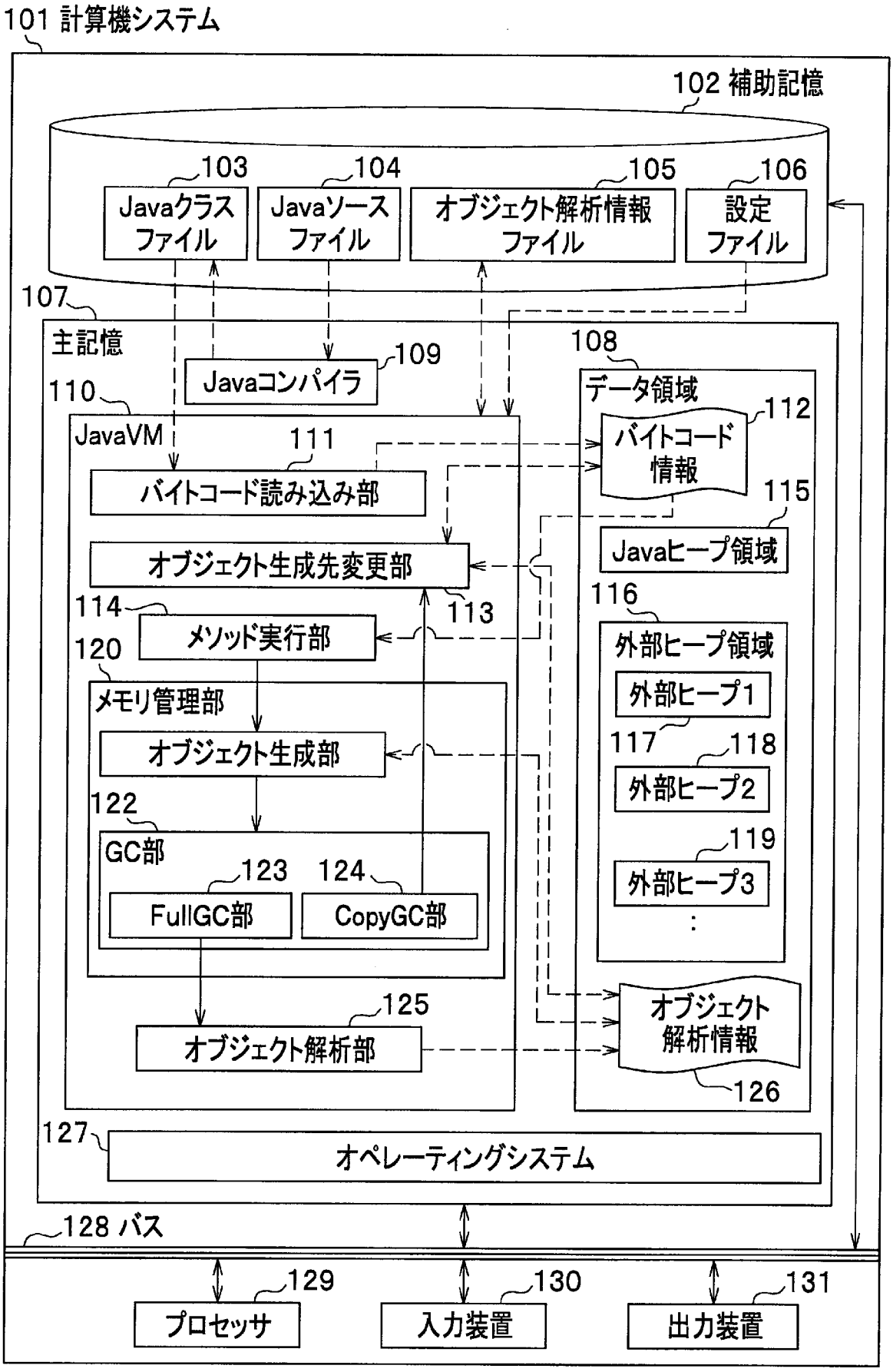
前記割合が前記第２の閾値以上であるとき、前記オブジェクト生成命令によるオブジェクトの生成先を前記第２のメモリ領域に変更する

制御と、を実行する制御部とを有する

ことを特徴とする情報処理装置。

WO 2011/045949 PCT/JP2010/053520

[図1]



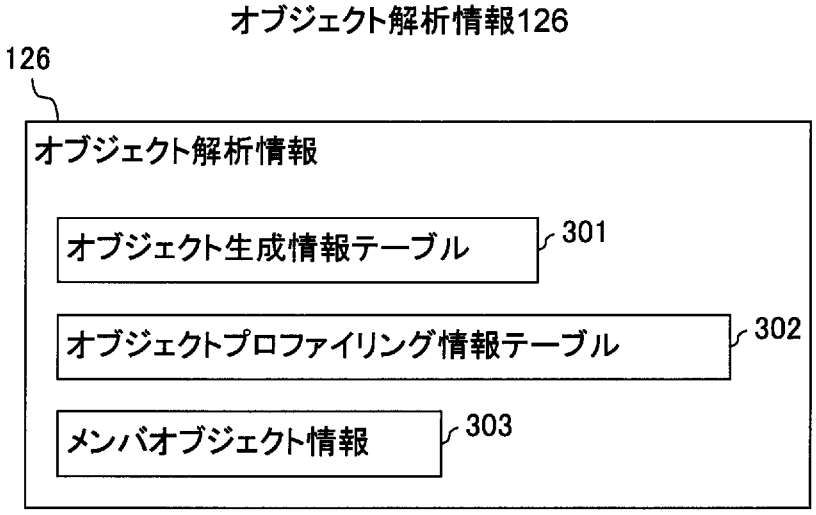
[図2]

Javaソースファイル104の例

104 Javaソースファイル

行番号:	Javaソースコード
1:	public class Sample {
2:	・・・
3:	ClassA methodA() {
4:	// 長期的に使用するClassAのオブジェクトを生成する
5:	ClassA objA = new ClassA();
6:	ClassZ objZ = new ClassZ();
7:	・・・
8:	return objA;
9:	}
10:	・・・
11:	void methodB() {
12:	// 短期的に使用するClassAのオブジェクトを生成する
13:	ClassA objA = new ClassA();
14:	・・・
15:	}
16:	・・・
17:	}

[図3]



[図4]

オブジェクト生成情報テーブル 301

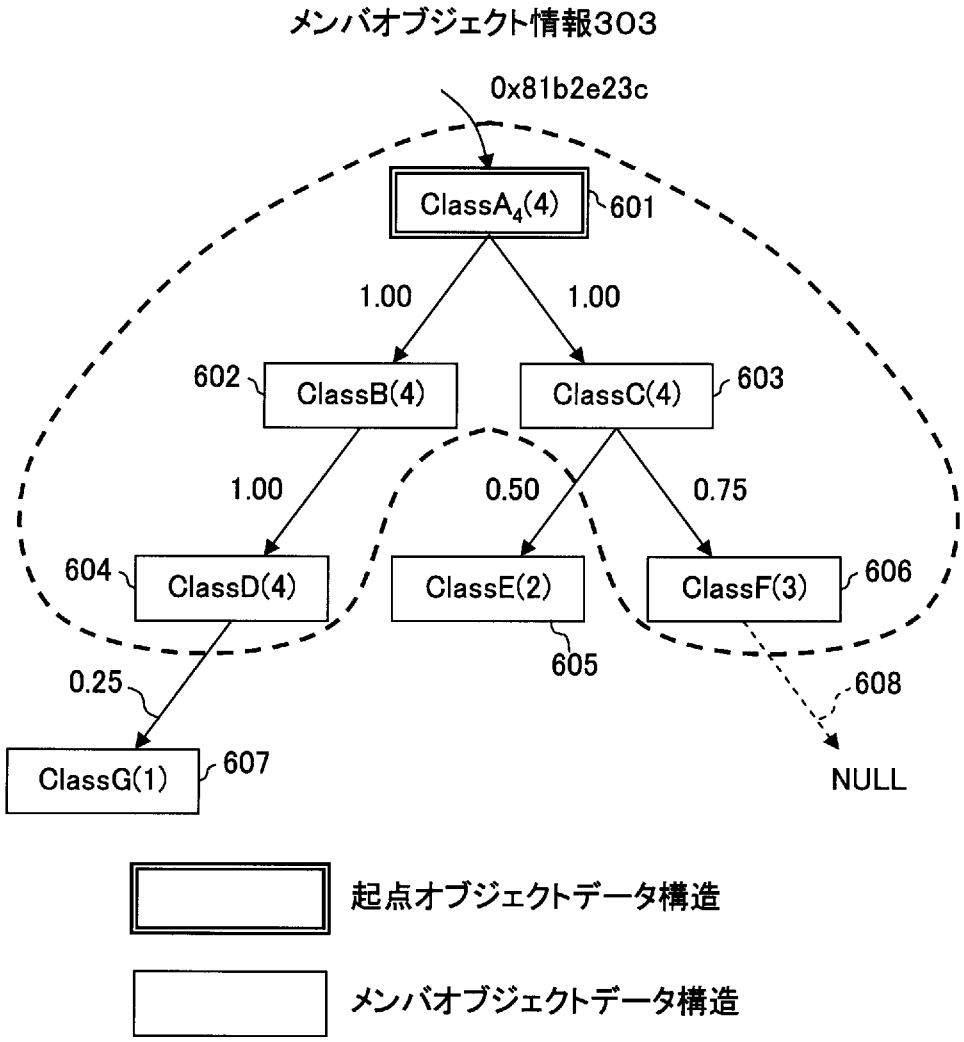
401 オブジェクト生成命令アドレス	402 オブジェクトID	403 クラス名	404 オブジェクト生成先
0x70003284	#0	ClassA	外部ヒープ1
0x700032a8	#1	ClassZ	Javaヒープ
0x700032fc	#2	ClassB	Javaヒープ
0x70003304	#3	ClassC	外部ヒープ2
0x700034c0	#4	ClassA	Javaヒープ
:	:	:	:
0x70014ff0	#103	ClassA	Javaヒープ
0x70015bc8	#104	ClassZ	Javaヒープ
0x70016014	#105	ClassX	Javaヒープ固定
0x700162b0	#106	ClassY	指定外部ヒープ
-	-	-	-

[図5]

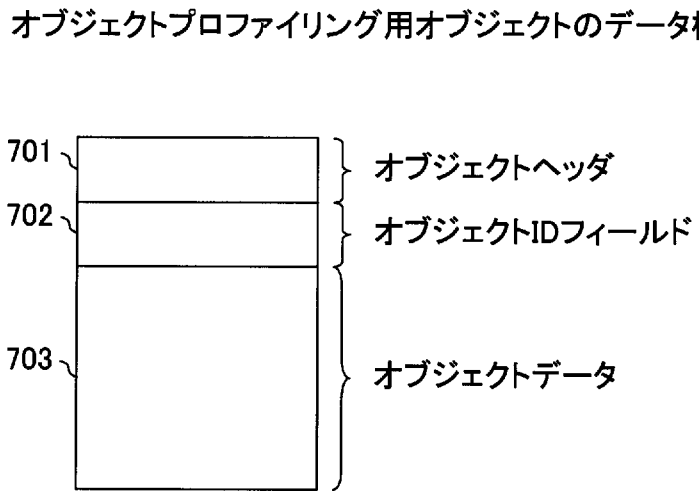
オブジェクトプロファイリング情報テーブル302

501 オブジェクトID	502 オブジェクト生成回数	503 オブジェクト プロファイリング回数	504 メンバオブジェクト情報 アドレス
#0	5872	5830	0x81ac3298
#1	5872	0	NULL
#2	0	0	NULL
#3	7923	7879	0x81af4370
#4	4	4	0x81b2e23c
:	:	:	:
#103	10281	237	0x83cff130
#104	950	950	0x83cff2a4
#105	-	-	-
#106	-	-	-
-	-	-	-

[図6]


メンバオブジェクト情報303
0x81b2e23c
ClassA4(4)
601
1.00
1.00
602
ClassB(4)
ClassC(4)
603
1.00
0.50
0.75
604
ClassD(4)
ClassE(2)
ClassF(3)
606
605
0.25
608
ClassG(1)
607
NULL
起点オブジェクトデータ構造
メンバオブジェクトデータ構造

[図7]


オブジェクトプロファイリング用オブジェクトのデータ構造
701
オブジェクトヘッダ
702
オブジェクトIDフィールド
703
オブジェクトデータ

[図8]

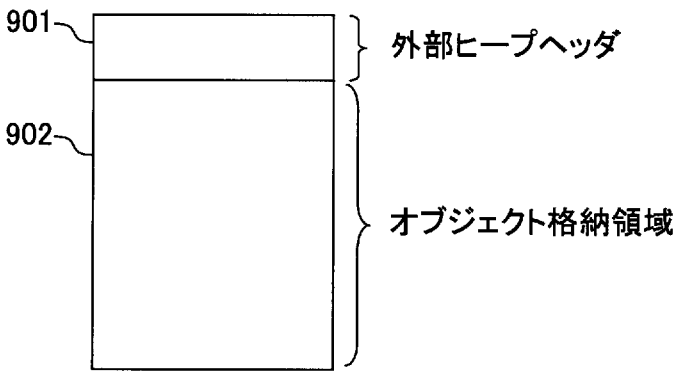
オブジェクト種別ビット

801 802

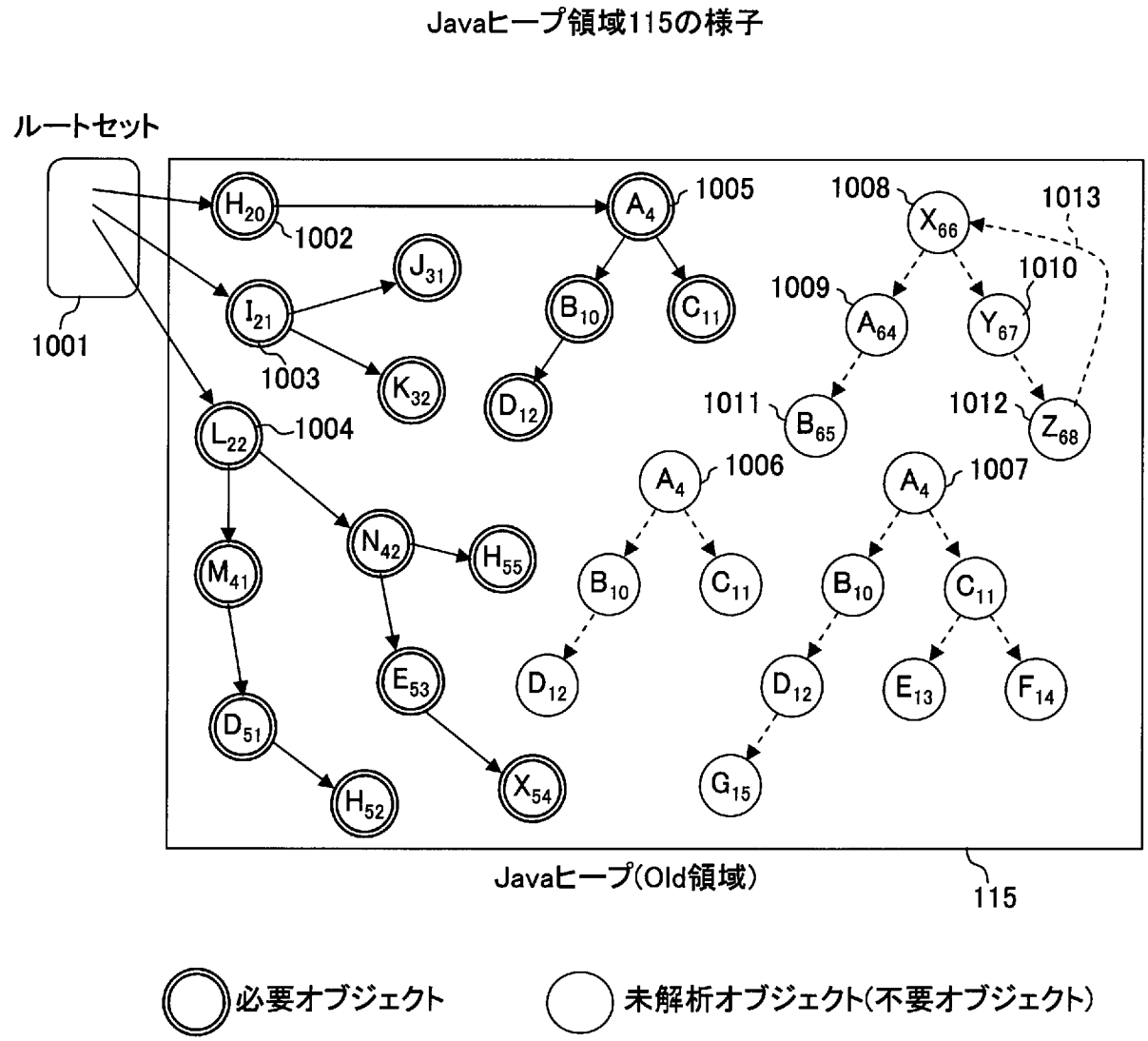
オブジェクト種別	オブジェクト種別ビット
未解析	00
必要	01
起点	10
メンバ	11

[図9]

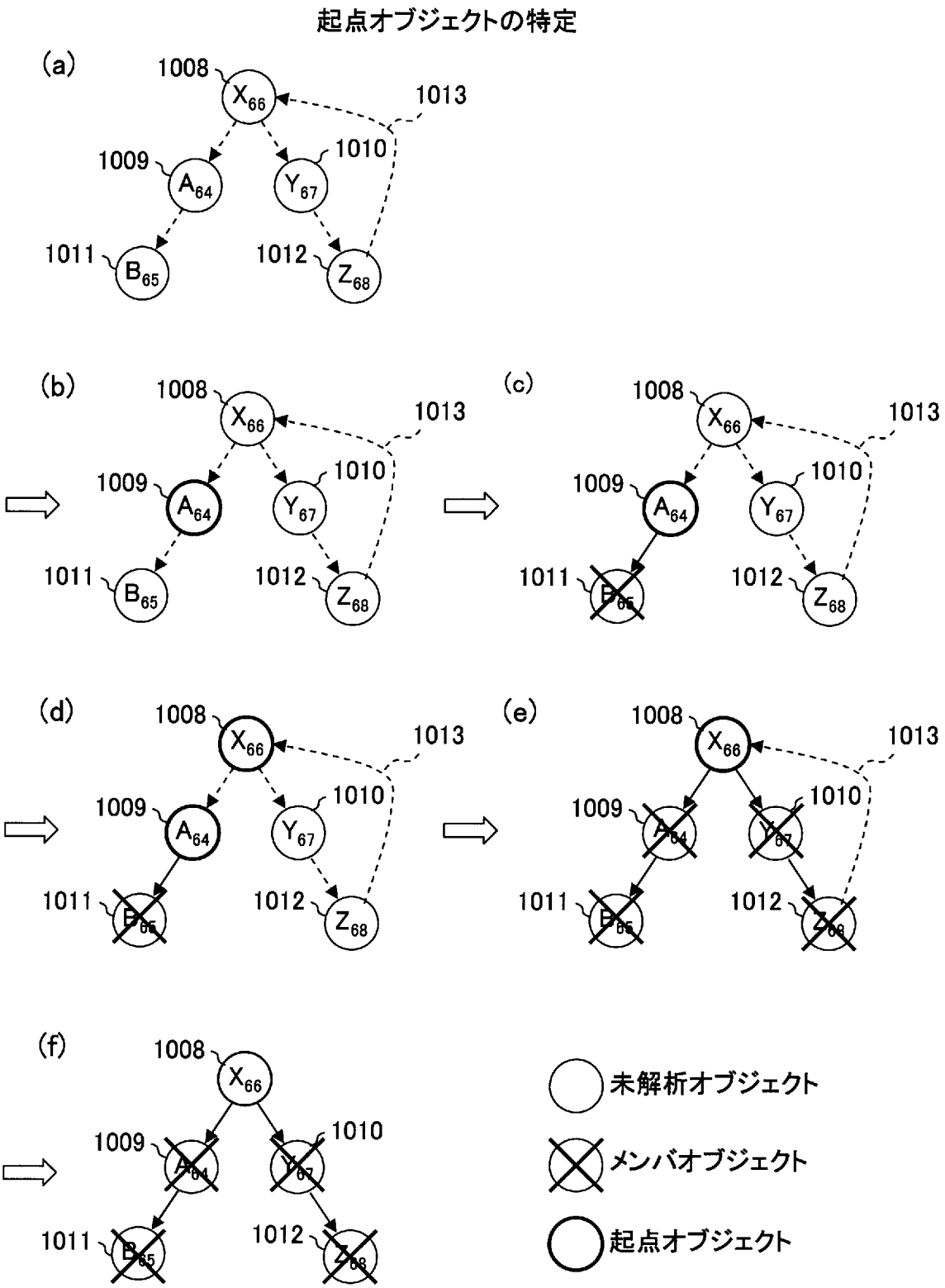
外部ヒープのデータ構造



[図10]

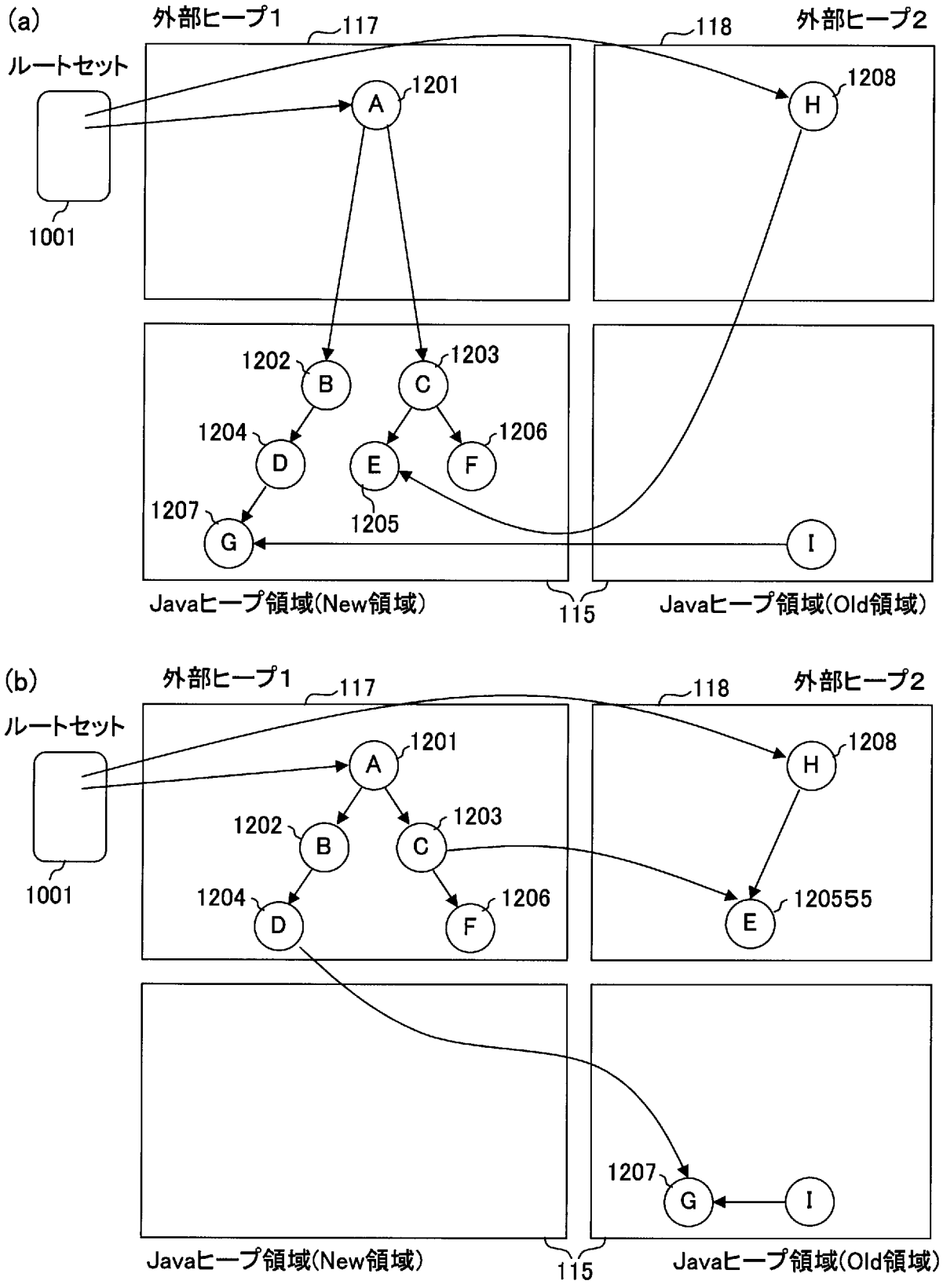

Javaヒープ領域115の様子
ルートセット
1001
H20
1002
I21
1003
L22
1004
M41
N42
H55
D51
H52
J31
K32
E53
X54
A4
1005
B10
C11
D12
A4
1006
B10
C11
D12
E53
A4
1007
B10
C11
D12
E13
F14
G15
1008
X66
1013
1009
A64
1010
Y67
1011
B65
1012
Z68
Javaヒープ(Old領域)
115
必要オブジェクト
未解析オブジェクト(不要オブジェクト)

[図11]

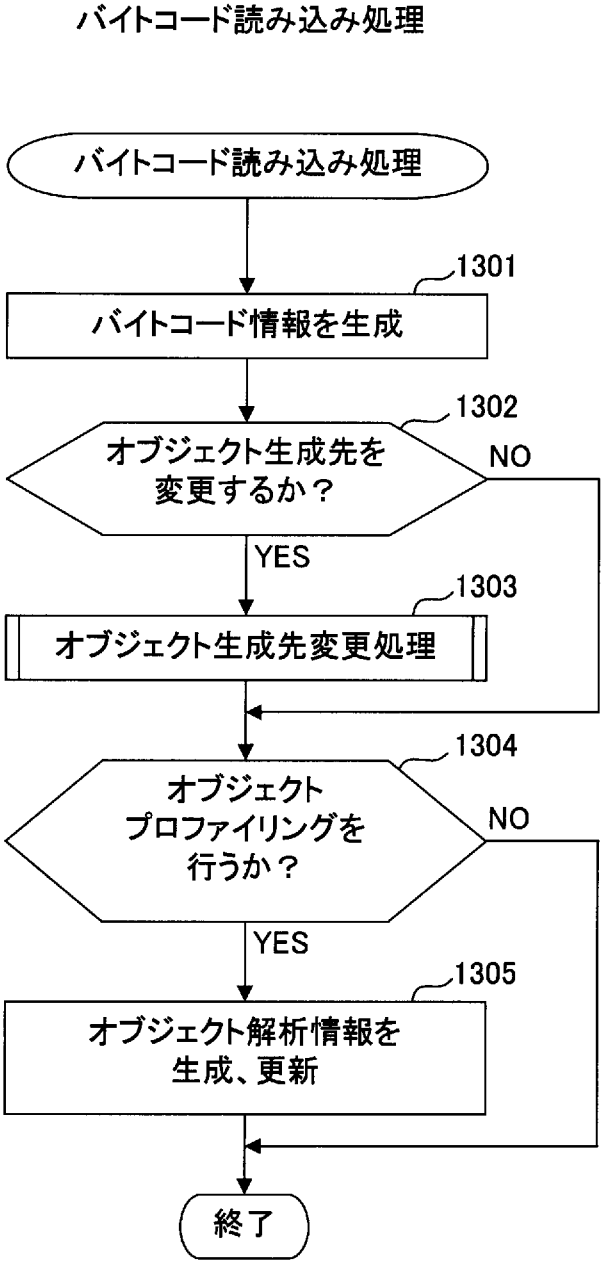

起点オブジェクトの特定
(a)
(b)
(c)
(d)
(e)
(f)
1008
1009
1010
1011
1012
1013
X66
A64
Y67
B65
Z68
未解析オブジェクト
メンバオブジェクト
起点オブジェクト

[図12]

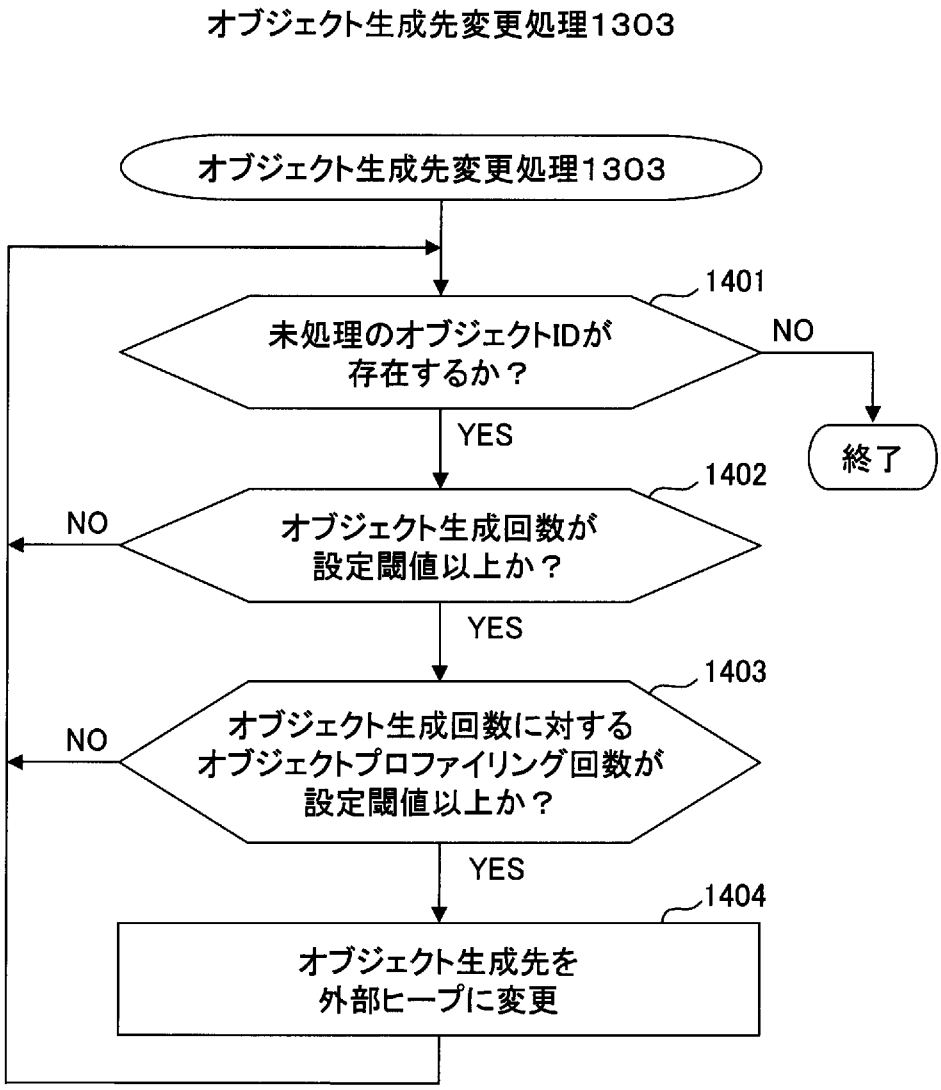
CopyGC処理によるメンバオブジェクトの移動の例



[図13]

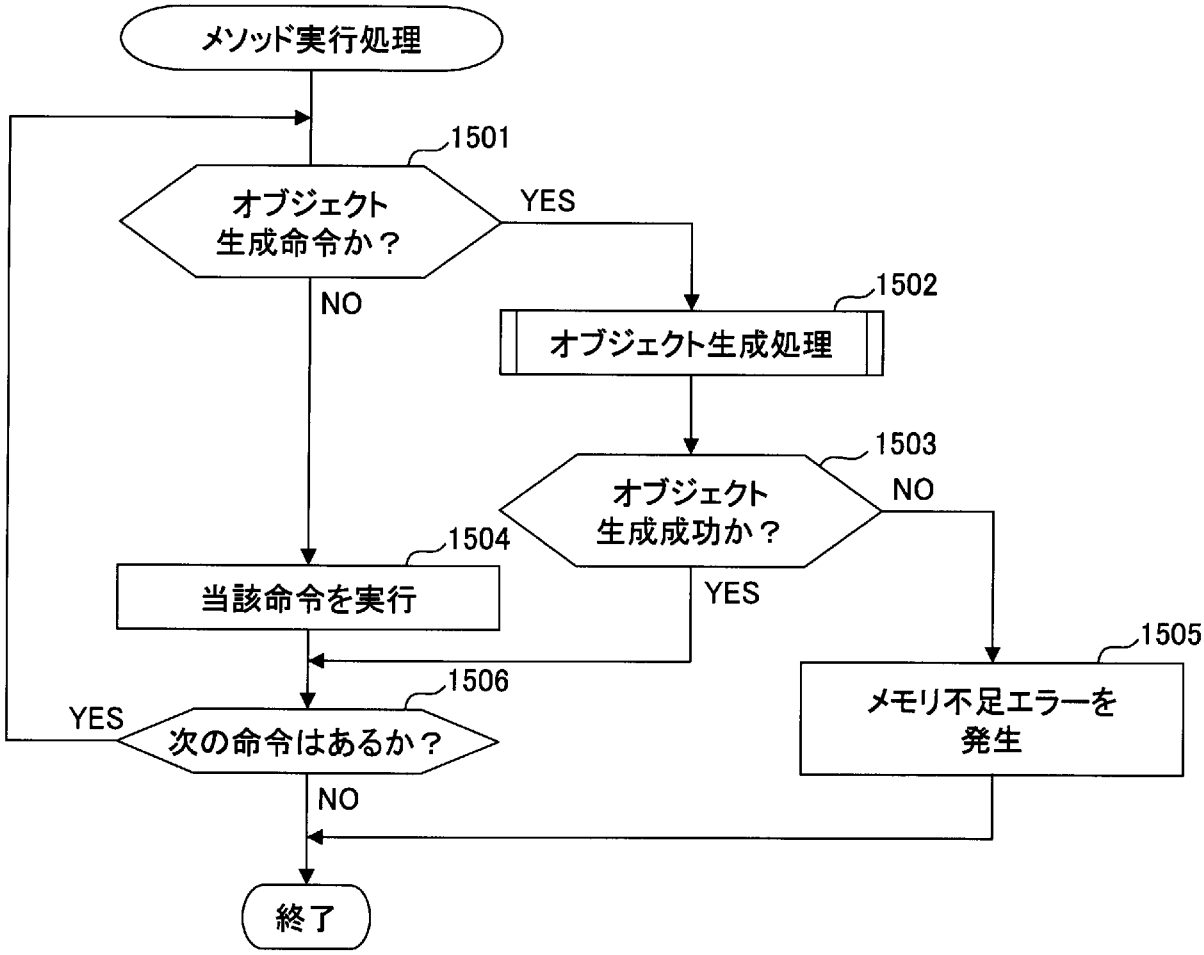

バイトコード読み込み処理
バイトコード読み込み処理
1301
バイトコード情報を生成
1302
オブジェクト生成先を
変更するか？
NO
YES
1303
オブジェクト生成先変更処理
1304
オブジェクト
プロファイリングを
行うか？
NO
YES
1305
オブジェクト解析情報を
生成、更新
終了

[図14]

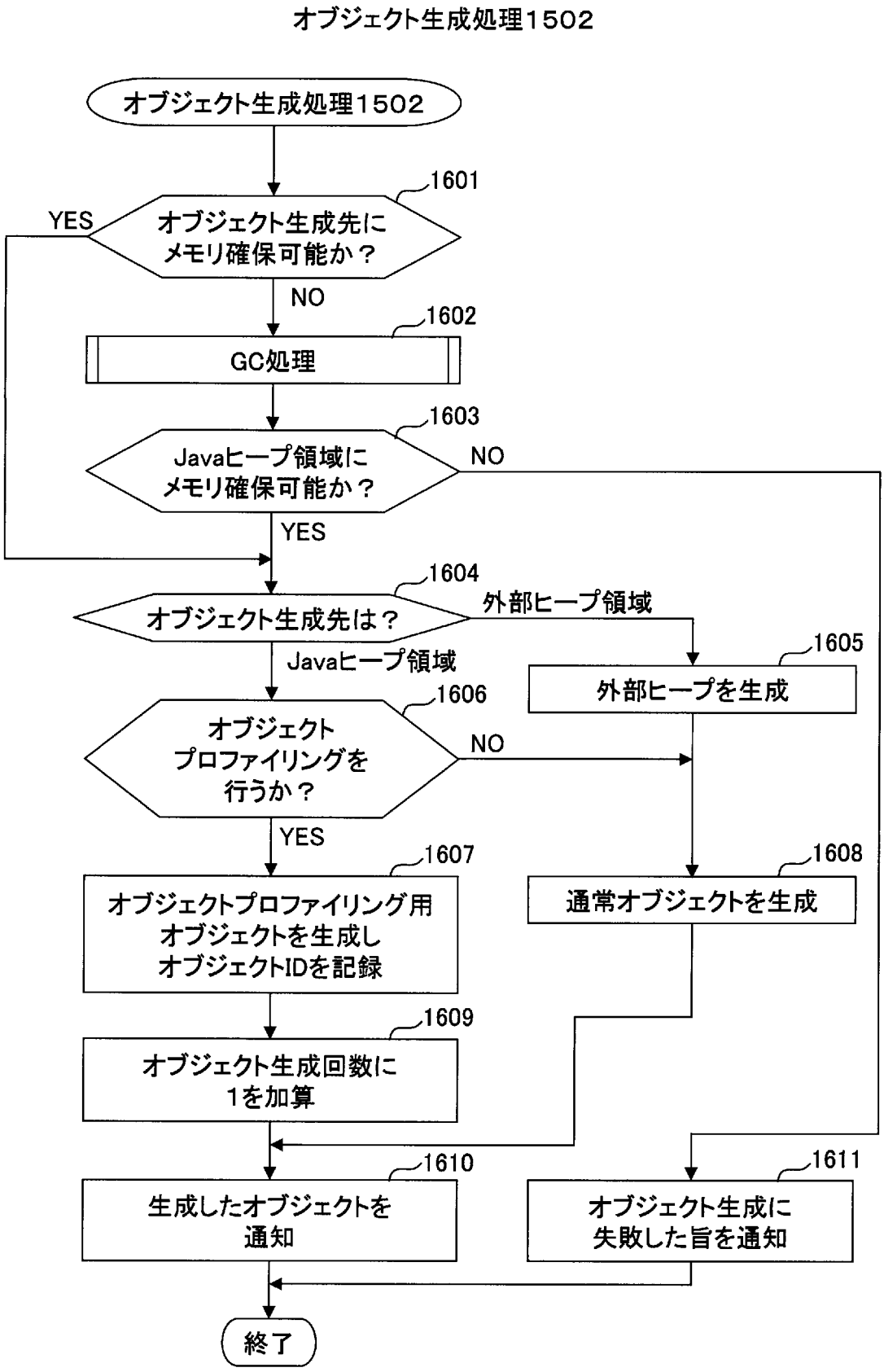

オブジェクト生成先変更処理1303
オブジェクト生成先変更処理1303
1401
未処理のオブジェクトIDが
存在するか？
NO
終了
YES
1402
NO
オブジェクト生成回数が
設定閾値以上か？
YES
1403
NO
オブジェクト生成回数に対する
オブジェクトプロファイリング回数が
設定閾値以上か？
YES
1404
オブジェクト生成先を
外部ヒープに変更

[図15]

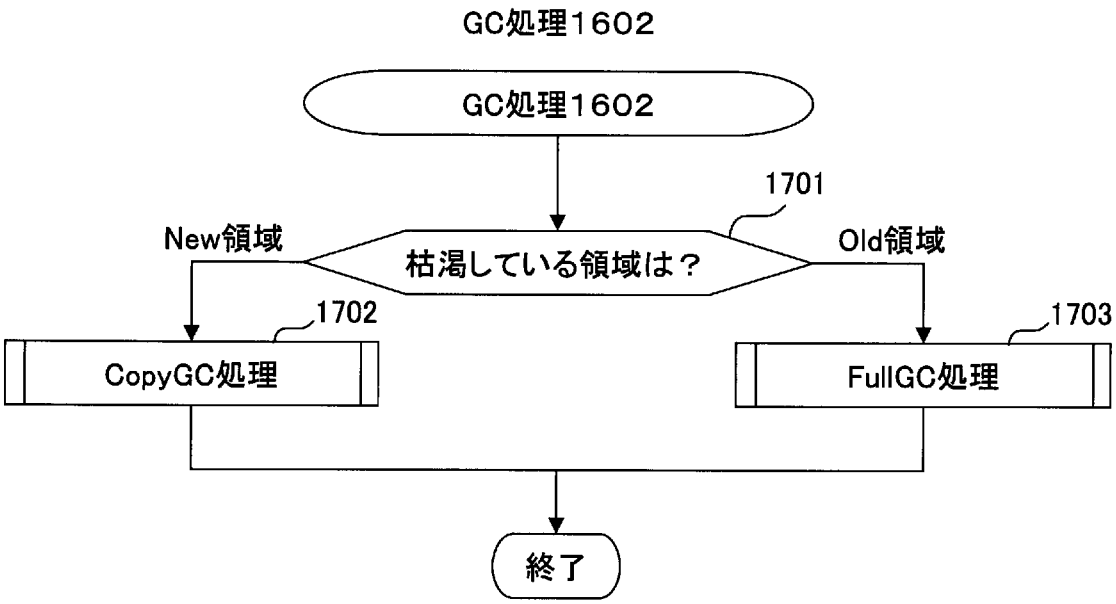
メソッド実行処理



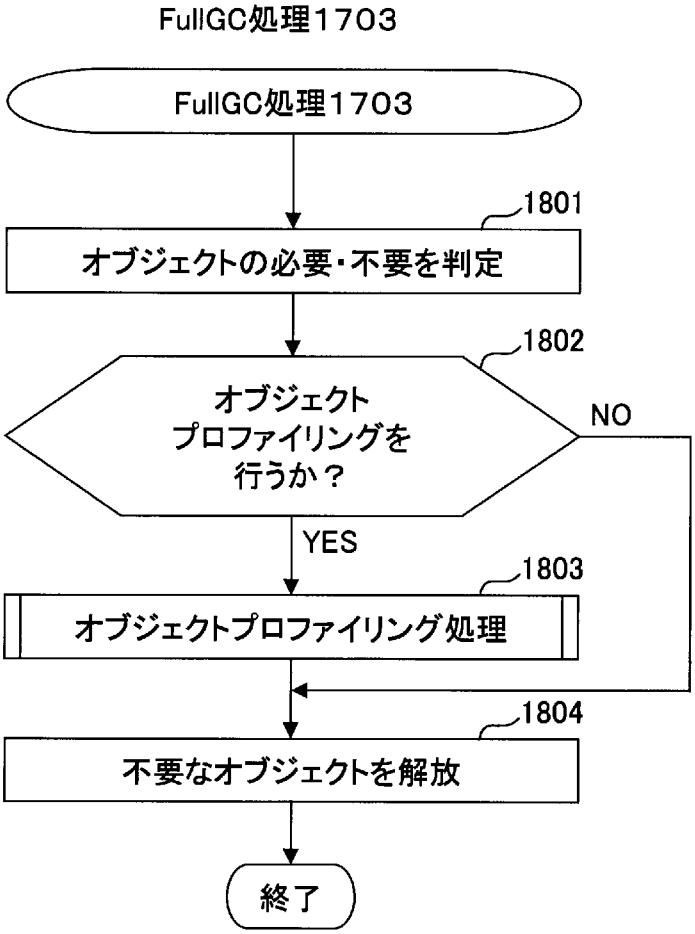
[図16]


オブジェクト生成処理1502
オブジェクト生成処理1502
1601
オブジェクト生成先に
メモリ確保可能か？
YES
NO
1602
GC処理
1603
Javaヒープ領域に
メモリ確保可能か？
NO
YES
1604
オブジェクト生成先は？
外部ヒープ領域
Javaヒープ領域
1605
外部ヒープを生成
1606
オブジェクト
プロファイリングを
行うか？
NO
YES
1607
オブジェクトプロファイリング用
オブジェクトを生成し
オブジェクトIDを記録
1608
通常オブジェクトを生成
1609
オブジェクト生成回数に
1を加算
1610
生成したオブジェクトを
通知
1611
オブジェクト生成に
失敗した旨を通知
終了

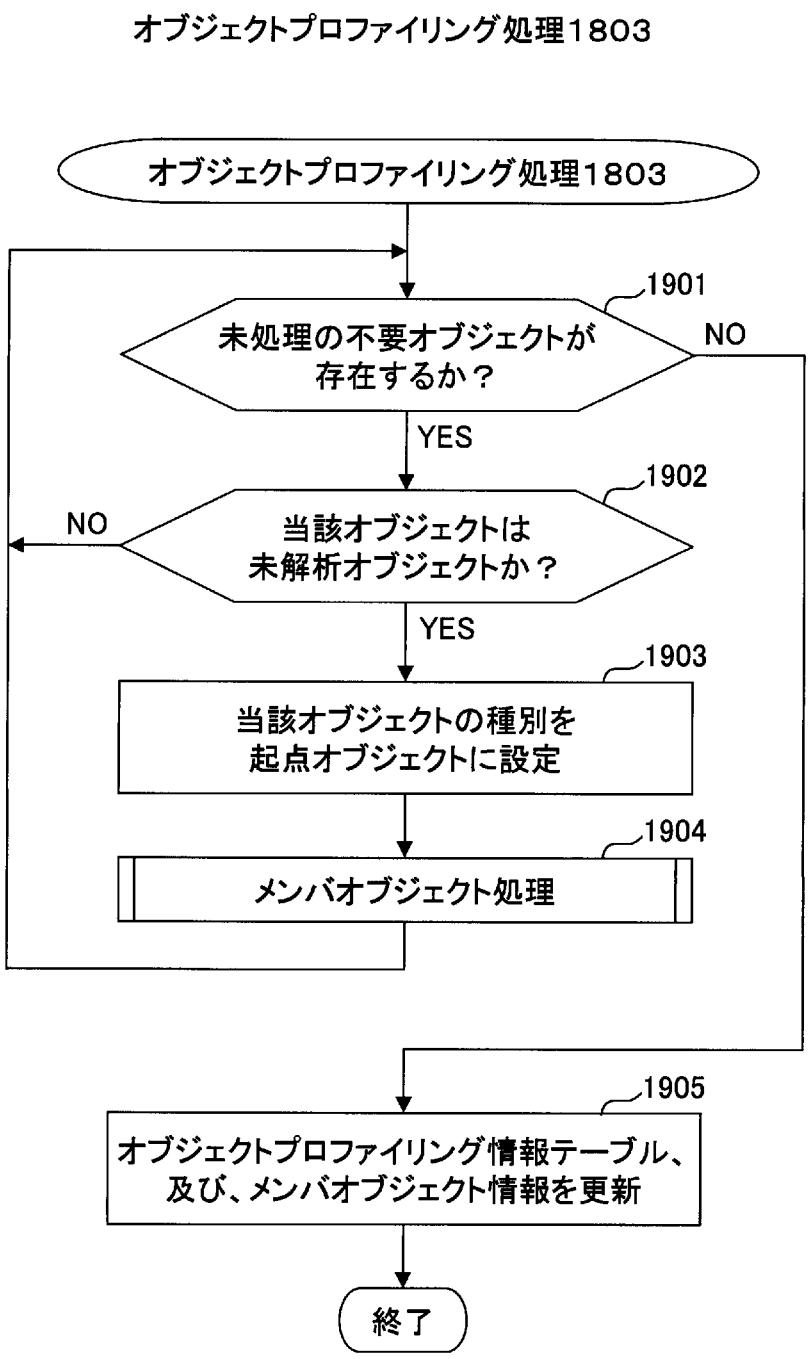
[図17]


GC処理1602
GC処理1602
1701
New領域
枯渇している領域は？
Old領域
1702
1703
CopyGC処理
FullGC処理
終了

[図18]

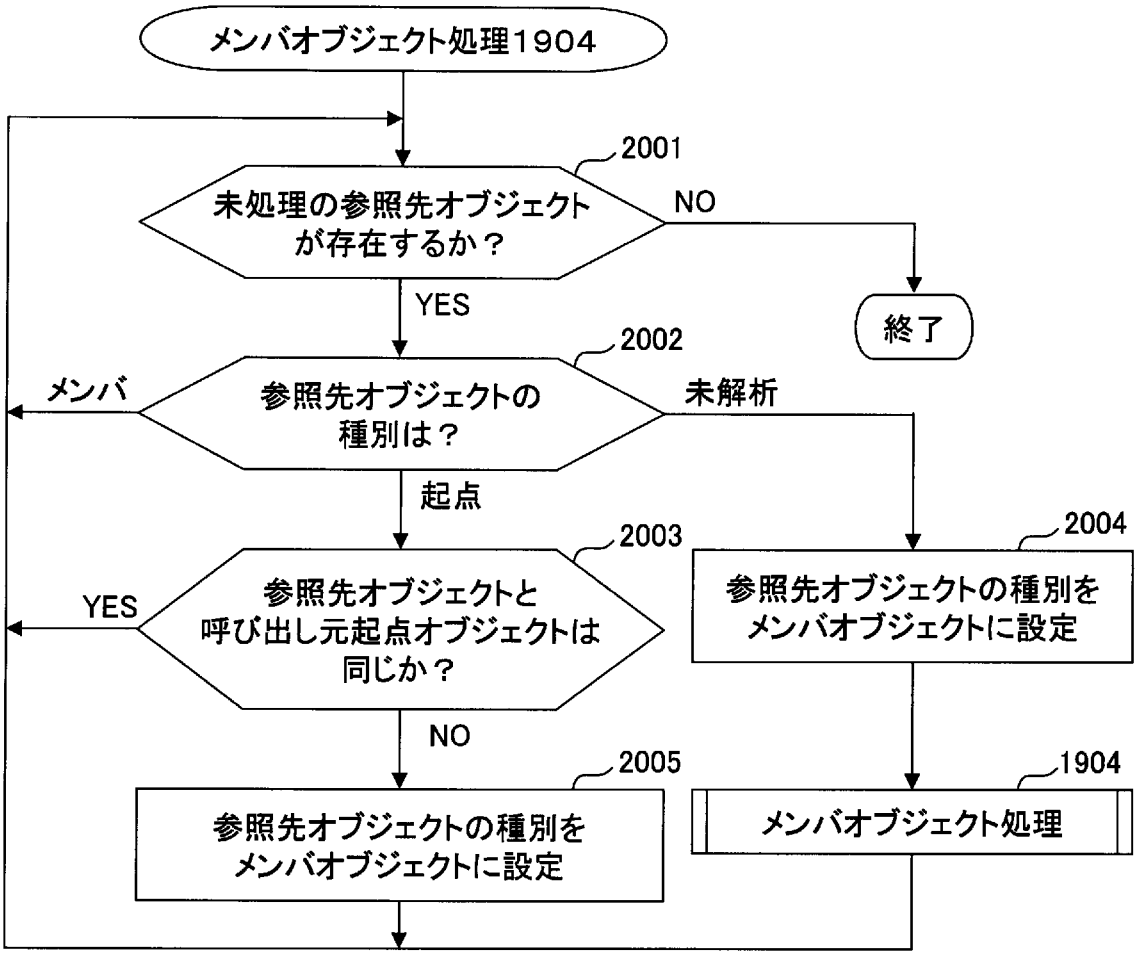

FullGC処理1703
FullGC処理1703
1801
オブジェクトの必要・不要を判定
1802
オブジェクト
プロファイリングを
行うか？
NO
YES
1803
オブジェクトプロファイリング処理
1804
不要なオブジェクトを解放
終了

[図19]

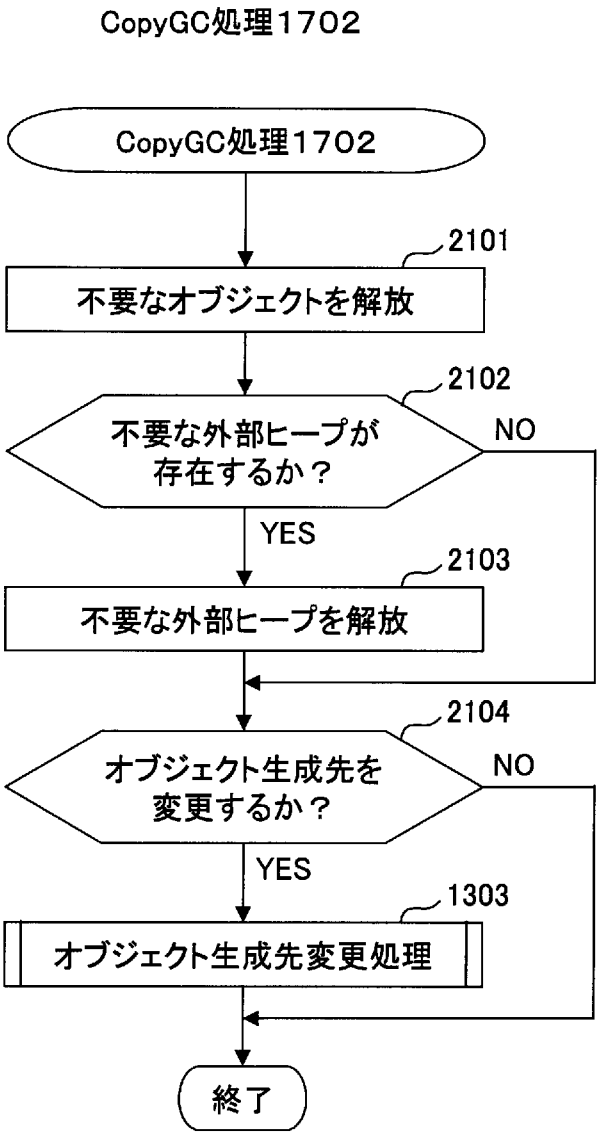

オブジェクトプロファイリング処理1803
オブジェクトプロファイリング処理1803
1901
未処理の不要オブジェクトが
存在するか？
NO
YES
1902
NO
当該オブジェクトは
未解析オブジェクトか？
YES
1903
当該オブジェクトの種別を
起点オブジェクトに設定
1904
メンバオブジェクト処理
1905
オブジェクトプロファイリング情報テーブル、
及び、メンバオブジェクト情報を更新
終了

[図20]

メンバオブジェクト処理1904

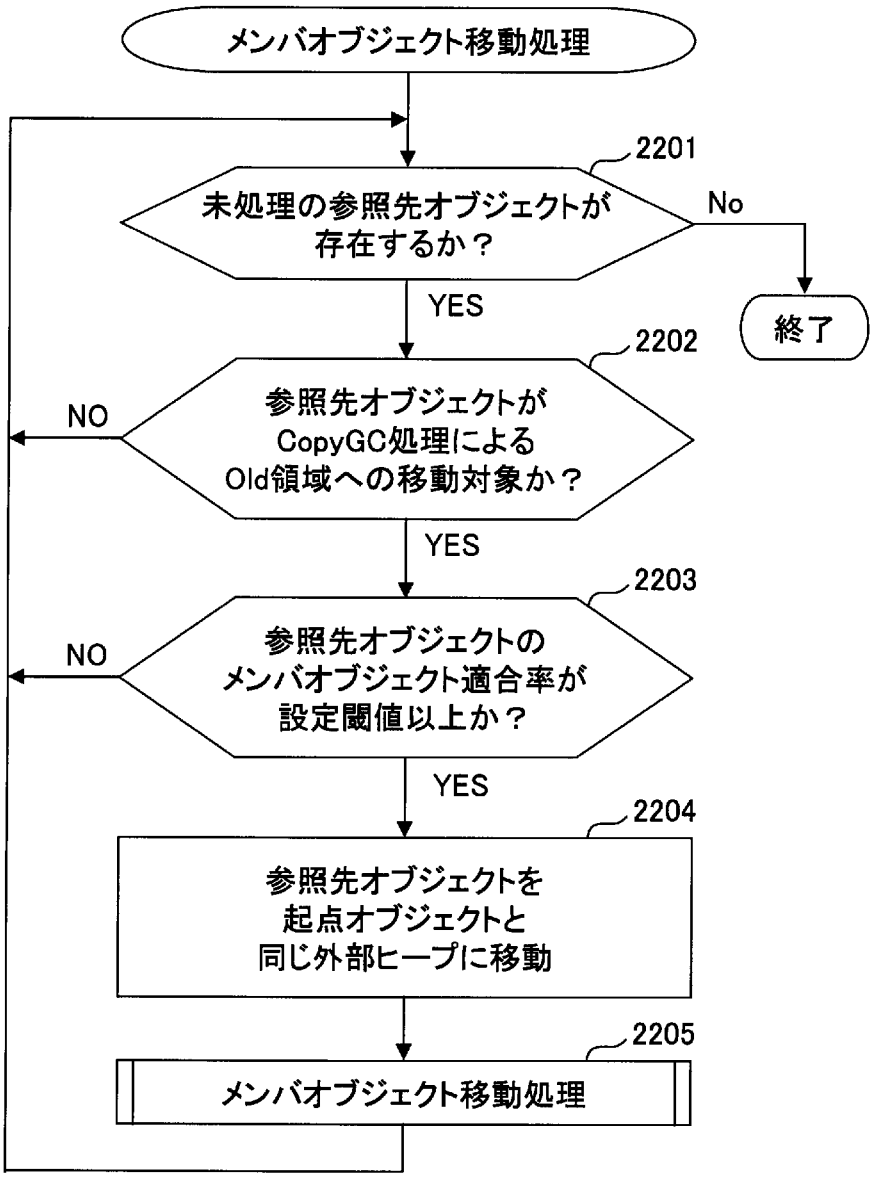


[図21]

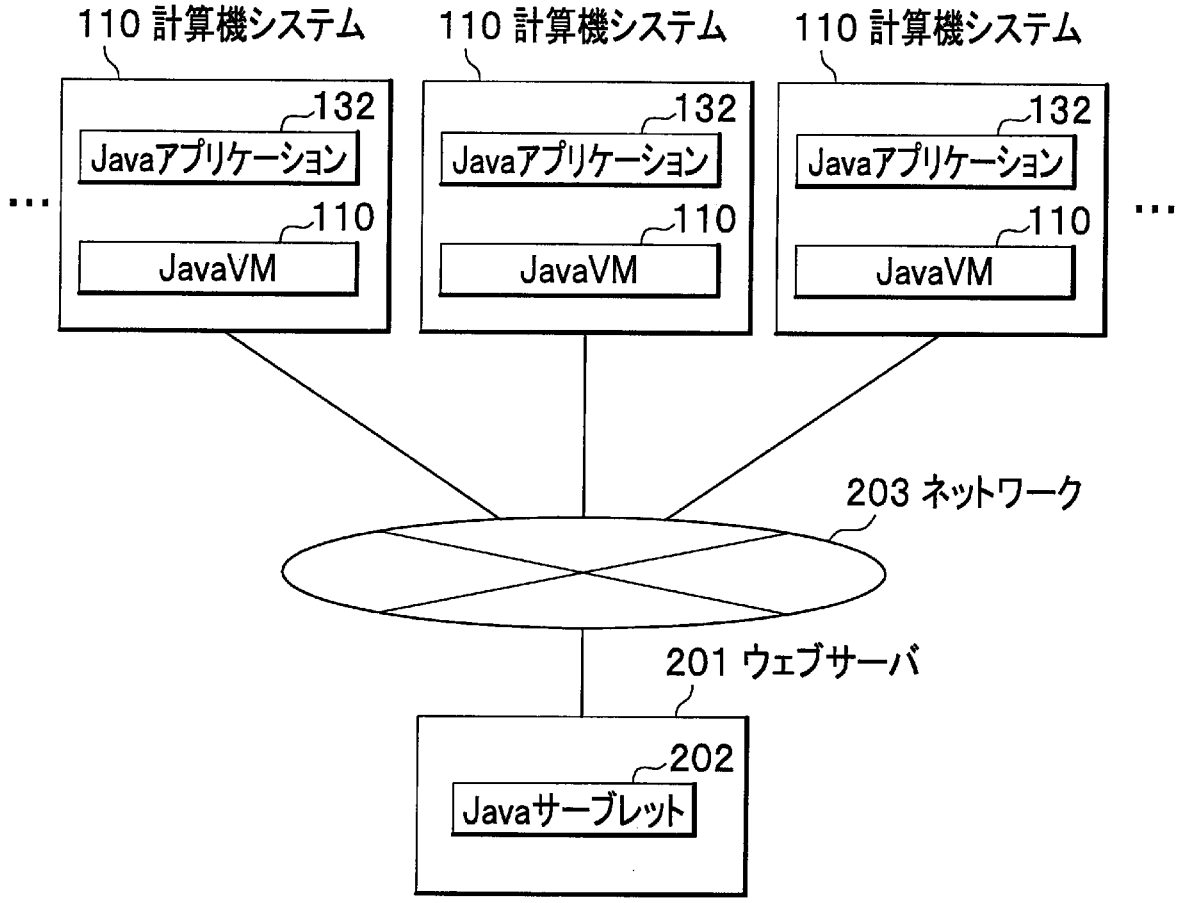

CopyGC処理1702
CopyGC処理1702
2101
不要なオブジェクトを解放
2102
不要な外部ヒープが
存在するか？
NO
YES
2103
不要な外部ヒープを解放
2104
オブジェクト生成先を
変更するか？
NO
YES
1303
オブジェクト生成先変更処理
終了

[図22]

メンバオブジェクト移動処理



[図23]


110 計算機システム
110 計算機システム
110 計算機システム
132
Javaアプリケーション
110
JavaVM
132
Javaアプリケーション
110
JavaVM
132
Javaアプリケーション
110
JavaVM
203 ネットワーク
201 ウェブサーバ
202
Javaサーブレット

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2010/053520

A. CLASSIFICATION OF SUBJECT MATTER
G06F12/00(2006.01)i, *G06F9/44*(2006.01)i, *G06F12/02*(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F12/00, G06F9/44, G06F12/02

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho	1922-1996	Jitsuyo Shinan Toroku Koho	1996-2010
Kokai Jitsuyo Shinan Koho	1971-2010	Toroku Jitsuyo Shinan Koho	1994-2010

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Motoki KOHATA et al., "Java ni Okeru Meijiteki Memory Kanri", Transactions of Information Processing Society of Japan, Ronbunshi Journal, 15 July 2009 (15.07.2009), vol.50, no.7, pages 1693 to 1715(particularly, page 1713, left column, line 23 to right column, line 28)	1-7
A	US 2007/0180002 A1 (SUN MICROSYSTEMS, INC.), 02 August 2007 (02.08.2007), all pages; all drawings (Family: none)	1-7
A	US 6681234 B2 (SUN MICROSYSTEMS, INC.), 20 January 2004 (20.01.2004), abstract; fig. 2, 3 & US 2002/0073404 A1 & WO 2002/057913 A2	1-7

☒ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
30 June, 2010 (30.06.10)	13 July, 2010 (13.07.10)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2010/053520

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 2003-241967 A (Matsushita Electric Industrial Co., Ltd.), 29 August 2003 (29.08.2003), abstract; paragraphs [0054] to [0066], [0091] to [0095]; fig. 2 (Family: none)	1-7
P,A	JP 2010-044532 A (Hitachi, Ltd.), 25 February 2010 (25.02.2010), all pages; all drawings & US 2010/0049938 A1	1-7

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ２０１０／０５３５２０

Ａ．　発明の属する分野の分類（国際特許分類（ＩＰＣ））

Int.Cl.　G06F12/00(2006.01)i, G06F9/44(2006.01)i, G06F12/02(2006.01)i

Ｂ．　調査を行った分野

調査を行った最小限資料（国際特許分類（ＩＰＣ））

Int.Cl.　G06F12/00, G06F9/44, G06F12/02

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報	１９２２－１９９６年
日本国公開実用新案公報	１９７１－２０１０年
日本国実用新案登録公報	１９９６－２０１０年
日本国登録実用新案公報	１９９４－２０１０年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

Ｃ．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求項の番号
A	小幡元樹　外５名，Ｊａｖａにおける明示的メモリ管理，情報処理学会論文誌　論文誌ジャーナル，2009.07.15，第50巻，第7号，p.1693-1715（特に、p.1713 左欄第23行目乃至右欄第28行目）	１－７
A	US 2007/0180002 A1 (SUN MICROSYSTEMS, INC.) 2007.08.02, 全頁,全図（ファミリーなし）	１－７

☑　Ｃ欄の続きにも文献が列挙されている。

☐　パテントファミリーに関する別紙を参照。

＊　引用文献のカテゴリー
「Ａ」特に関連のある文献ではなく、一般的技術水準を示すもの
「Ｅ」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「Ｌ」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「Ｏ」口頭による開示、使用、展示等に言及する文献
「Ｐ」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「Ｔ」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「Ｘ」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Ｙ」特に関連のある文献であって、当該文献と他の１以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「＆」同一パテントファミリー文献

国際調査を完了した日 ３０．０６．２０１０	国際調査報告の発送日 １３．０７．２０１０
国際調査機関の名称及びあて先 日本国特許庁（ＩＳＡ／ＪＰ） 郵便番号１００－８９１５ 東京都千代田区霞が関三丁目４番３号	特許庁審査官（権限のある職員）　５Ｎ　９３６７ 多賀　実 電話番号　０３－３５８１－１１０１　内線　３５８６

Ｃ（続き）．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求項の番号
Ａ	US 6681234 B2 (SUN MICROSYSTEMS, INC.) 2004.01.20, [ABSTRACT], FIG.2,3 & US 2002/0073404 A1 & WO 2002/057913 A2	１－７
Ａ	JP 2003-241967 A (松下電器産業株式会社) 2003.08.29, 【要約】, 【0054】-【0066】, 【0091】-【0095】, 図２ (ファミリーなし)	１－７
Ｐ，Ａ	JP 2010-044532 A (株式会社日立製作所) 2010.02.25, 全頁, 全図 & US 2010/0049938 A1	１－７