(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2020년04월13일
(11) 등록번호 10-2100531
(24) 등록일자 2020년04월07일

(51) 국제특허분류(Int. Cl.)
G06F 9/44 (2018.01) *G06F 8/40* (2018.01)
(52) CPC특허분류
G06F 8/315 (2013.01)
G06F 8/436 (2013.01)
(21) 출원번호 10-2015-7021105
(22) 출원일자(국제) 2014년01월03일
심사청구일자 2018년12월06일
(85) 번역문제출일자 2015년08월04일
(65) 공개번호 10-2015-0104156
(43) 공개일자 2015년09월14일
(86) 국제출원번호 PCT/US2014/010112
(87) 국제공개번호 WO 2014/107539
국제공개일자 2014년07월10일
(30) 우선권주장
13/734,750 2013년01월04일 미국(US)
(56) 선행기술조사문헌
KR1020070040376 A

(73) 특허권자
마이크로소프트 테크놀로지 라이센싱, 엘엘씨
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
더피 존 제이.
미국 워싱턴주 98052-6399 레드몬드 원 마이크로소프트 웨이 엘씨에이 - 인터내셔널 패턴츠 마이크로소프트 코포레이션 내
파슨스 재러드 포터
미국 워싱턴주 98052-6399 레드몬드 원 마이크로소프트 웨이 엘씨에이 - 인터내셔널 패턴츠 마이크로소프트 코포레이션 내
(뒷면에 계속)
(74) 대리인
김태홍, 김진희

전체 청구항 수 : 총 21 항 심사관 : 지정훈

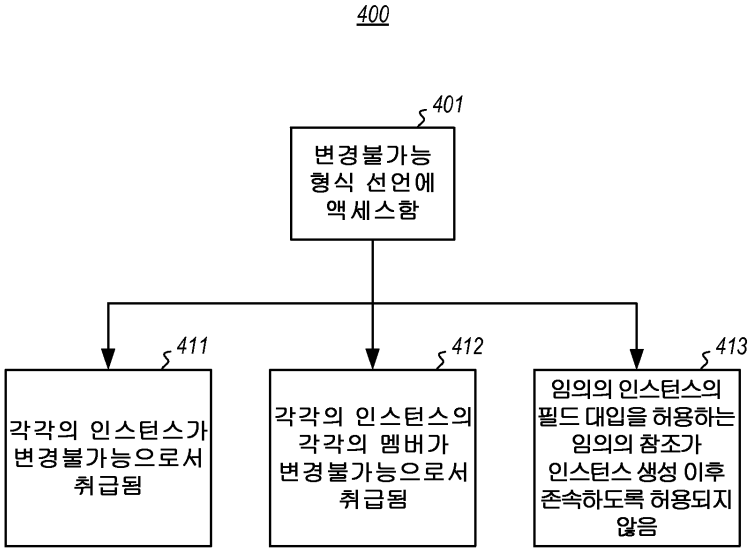
(54) 발명의 명칭 **변경불가 객체 형식**

(57) 요 약

본 발명은 어떤 형식의 인스턴스들 모두가 변경불가능(immutable)인 경우에, 그 형식 전체가 변경불가능인 것으로 선언될 수 있다는 점에서 시스템 프로그래밍에서의 안전성을 향상시키는 언어 확장에 관한 것이다. 변경불가능 형식 선언은 자동으로 그 형식의 임의의 인스턴스들을 변경불가능인 것으로 취급되게 하고, 자동으로 그 인스

(뒷면에 계속)

대 표 도 - 도4



턴스의 모든 직간접적으로 도달가능한 멤버들(예컨대, 필드들, 메서드들, 속성들)을 또한 변경불가능인 것으로 취급되게 한다. 게다가, 인스턴스의 필드 할당을 가능하게 하는 임의의 생성시 참조(construction time reference)는 인스턴스가 그의 작성자(creator)에 대해 액세스가능하게 되는 시점을 넘어 존속하도록 허용되지 않는다. 그에 따라, 이 인스턴스, 및 그 동일한 형식의 임의의 다른 인스턴스는 생성 시부터 변경불가능일 것이다. 이러한 인스턴스들 모두를 변경불가능으로서 분류할 수 있다는 것은 유익한데, 그 이유는 변경불가능 특성이 보통 리소스 액세스 안전성으로 인해 허용되지 않을 동작들을 가능하게 하기 때문이다.

(52) CPC특허분류
G06F 8/437 (2013.01)

(72) 발명자
신즈 마이클
미국 워싱턴주 98052-6399 레드몬드 원 마이크로소프트 웨이 엘씨에이 - 인터내셔널 패턴츠 마이크로소프트 코포레이션 내
브롬필드 알렉산더 다니엘
미국 워싱턴주 98052-6399 레드몬드 원 마이크로소프트 웨이 엘씨에이 - 인터내셔널 패턴츠 마이크로소프트 코포레이션 내
크왈리나 크지슈토프 제이.
미국 워싱턴주 98052-6399 레드몬드 원 마이크로소프트 웨이 엘씨에이 - 인터내셔널 패턴츠 마이크로소프트 코포레이션 내

명 세 서

청구범위

청구항 1

객체 형식(object type)의 복수의 인스턴스들이 변경불가능(immutable)이도록 보장(ensure)하기 위한 컴퓨터 구현 방법에 있어서,

소프트웨어 소스 코드 내의 선언(declaration)에 액세스하는 단계 - 상기 선언은 객체 형식에 대한 명시적 주석을 포함하고, 상기 선언은 상기 객체 형식이 변경불가능이라고 지정함 - 와,

상기 선언에 액세스하는 것에 응답하여, 이하의 단계,

상기 객체 형식의 모든 인스턴스가 심지어 변경가능 퍼미션(mutable permission)이 할당되더라도 변경불가능이 되도록 상기 객체 형식에 대한 임의의 사용자 지정 퍼미션을 무시하는 컴퓨터 실행가능 명령어들을 작성(formulating)하는 것을 포함하여, 객체 형식의 각 인스턴스를 일단 이러한 인스턴스가 생성(construct)되면 변경불가능으로서 분류되도록 시행(enforce)하는 컴퓨터 실행가능 명령어들을 자동으로 작성하는 단계와,

상기 객체 형식의 인스턴스의 모든 직간접적으로 도달가능한 멤버들이 변경불가능으로서 분류되도록 시행하는 컴퓨터 실행가능 명령어들을 자동으로 작성하는 단계와,

생성 시에 상기 객체 형식의 인스턴스의 필드 할당(field assignment)을 허용하는 어떠한 참조도, 상기 인스턴스가 상기 인스턴스의 작성자(creator)에 대해 액세스가능하게 되는 시점(point)을 넘어 존속하는 것이 허용되지 않도록 시행하는 컴퓨터 실행가능 명령어들을 자동으로 작성하는 단계

를 수행하는 단계를 포함하는 컴퓨터 구현 방법.

청구항 2

제1항에 있어서, 상기 변경불가능 객체 형식의 적어도 하나의 인스턴스는, 하나 이상의 기록가능(writable) 입력 값을 취하는(take) 것인 컴퓨터 구현 방법.

청구항 3

제2항에 있어서, 상기 하나 이상의 기록가능 입력 값 중 어느 것도 상기 적어도 하나의 인스턴스 중 임의의 인스턴스의 필드에 저장되지 않는 것인 컴퓨터 구현 방법.

청구항 4

제1항에 있어서, 상기 참조의 인스턴스의 하나 이상의 멤버가 변경불가능 퍼미션(immutable permission)이 아닌 퍼미션을 포함하는 것인 컴퓨터 구현 방법.

청구항 5

제1항에 있어서, 상기 객체 형식의 상기 인스턴스들 중 임의의 인스턴스를 변경가능(mutable)으로 만드는 메커니즘이 존재하지 않는 것인 컴퓨터 구현 방법.

청구항 6

제1항에 있어서, 상기 인스턴스의 멤버들은 상기 인스턴스의 모든 필드, 메서드(method), 및 속성(property)을 포함하는 것인 컴퓨터 구현 방법.

청구항 7

제1항에 있어서, 필드 할당을 허용하는 상기 참조는 상기 객체 형식의 인스턴스를 생성하는 인스턴스 생성자(instance constructor)의 "this" 참조인 것인 컴퓨터 구현 방법.

청구항 8

제1항에 있어서, 상기 필드 할당을 허용하는 상기 참조에 대해 수행될 수 있는 일 세트의 하나 이상의 동작은, 상기 참조가 필드로서 저장되지 않도록 제한되는 것인 컴퓨터 구현 방법.

청구항 9

제8항에 있어서, 상기 일 세트의 하나 이상의 동작은, 판독 및 기록을 포함하는 것인 컴퓨터 구현 방법.

청구항 10

제1항에 있어서,

객체 그래프 내의 각 참조에 대한 퍼미션을 할당하는 것을 포함하여, 객체들에 대한 참조들을 포함하는 객체 그래프를 작성하는 단계를 더 포함하며,

상기 객체 형식의 인스턴스의 모든 직간접적으로 도달가능한 멤버들이 변경불가능으로서 분류되도록 시행하는 컴퓨터 실행가능 명령어들을 작성하는 것은, 각각의 직간접적으로 도달가능한 멤버에 대한 각각의 참조를 상기 객체 그래프에서 변경불가능으로서 표시하는 것을 포함하는 것인 컴퓨터 구현 방법.

청구항 11

컴퓨팅 시스템의 하나 이상의 프로세서에 의해 실행될 때, 상기 컴퓨팅 시스템으로 하여금 객체 형식의 복수의 인스턴스들이 변경불가능이도록 보장하기 위한 방법을 수행하게 하는 컴퓨터 실행가능 명령어들이 저장되어 있는 하나 이상의 하드웨어 저장 디바이스에 있어서, 상기 방법은,

소프트웨어 소스 코드 내의 선언에 액세스하는 단계 - 상기 선언은 객체 형식에 대한 명시적 주석을 포함하고, 상기 선언은 상기 객체 형식이 변경불가능이라고 지정함 - 와,

상기 선언에 액세스하는 것에 응답하여, 이하의 단계,

상기 객체 형식의 모든 인스턴스가 심지어 변경가능 퍼미션이 할당되더라도 변경불가능이 되도록 상기 객체 형식에 대한 임의의 사용자 지정 퍼미션을 무시하는 컴퓨터 실행가능 명령어들을 작성하는 것을 포함하여, 상기 객체 형식의 각 인스턴스를 일단 이러한 인스턴스가 생성되면 변경불가능으로서 분류되도록 시행하는 컴퓨터 실행가능 명령어들을 자동으로 작성하는 단계와,

상기 객체 형식의 인스턴스의 모든 직간접적으로 도달가능한 멤버들이 변경불가능으로서 분류되도록 시행하는 컴퓨터 실행가능 명령어들을 자동으로 작성하는 단계와,

생성 시에 상기 객체 형식의 인스턴스의 필드 할당을 허용하는 어떠한 참조도, 상기 인스턴스가 상기 인스턴스의 작성자에 대해 액세스가능하게 되는 시점을 넘어 존속하는 것이 허용되지 않도록 시행하는 컴퓨터 실행가능 명령어들을 자동으로 작성하는 단계

를 수행하는 단계를 포함하는 것인 하나 이상의 하드웨어 저장 디바이스.

청구항 12

제11항에 있어서, 상기 변경불가능 객체 형식의 적어도 하나의 인스턴스는, 하나 이상의 기록가능 입력 값을 취하는 것인 하나 이상의 하드웨어 저장 디바이스.

청구항 13

제12항에 있어서, 상기 하나 이상의 기록가능 입력 값 중 어느 것도 상기 적어도 하나의 인스턴스 중 임의의 인스턴스의 필드에 저장되지 않는 것인 하나 이상의 하드웨어 저장 디바이스.

청구항 14

제11항에 있어서, 상기 참조의 인스턴스의 하나 이상의 멤버가 변경불가능 퍼미션이 아닌 퍼미션을 포함하는 것인 하나 이상의 하드웨어 저장 디바이스.

청구항 15

제11항에 있어서, 상기 객체 형식의 상기 인스턴스들 중 임의의 인스턴스를 변경가능으로 만드는 메커니즘이 존

재하지 않는 것인 하나 이상의 하드웨어 저장 디바이스.

청구항 16

제11항에 있어서, 상기 인스턴스의 멤버들은 상기 인스턴스의 모든 필드, 메서드, 및 속성을 포함하는 것인 하나 이상의 하드웨어 저장 디바이스.

청구항 17

제11항에 있어서, 필드 할당을 허용하는 상기 참조는, 상기 객체 형식의 인스턴스를 생성하는 인스턴스 생성자의 "this" 참조인 것인 하나 이상의 하드웨어 저장 디바이스.

청구항 18

제11항에 있어서, 상기 필드 할당을 허용하는 상기 참조에 대해 수행될 수 있는 일 세트의 하나 이상의 동작은, 상기 참조가 필드로서 저장되지 않도록 제한되는 것인 하나 이상의 하드웨어 저장 디바이스.

청구항 19

시스템에 있어서,

하나 이상의 하드웨어 프로세서와,

상기 하나 이상의 하드웨어 프로세서에 의해 실행될 때, 상기 시스템으로 하여금 적어도 이하의 단계를 수행하도록 구성된 컴파일러를 구현하게 하는 컴퓨터 실행가능 명령어들이 저장되어 있는 하나 이상의 하드웨어 저장 디바이스

를 포함하며,

상기 이하의 단계는,

소프트웨어 소스 코드 내의 선언에 액세스하는 단계 - 상기 선언은 객체 형식에 대한 명시적 주석을 포함하고, 상기 선언은 상기 객체 형식이 변경불가능이라고 지정함 - 와,

상기 선언에 액세스하는 것에 응답하여, 적어도 이하의 단계,

상기 객체 형식의 모든 인스턴스가 심지어 변경가능 퍼미션이 할당되더라도 변경불가능이 되도록 상기 객체 형식에 대한 임의의 사용자 지정 퍼미션을 무시하는 컴퓨터 실행가능 명령어들을 작성하는 것을 포함하여, 객체 형식의 각 인스턴스를 일단 이러한 인스턴스가 생성되면 변경불가능으로서 분류되도록 시행하는 컴퓨터 실행가능 명령어들을 자동으로 작성하는 단계와,

상기 객체 형식의 인스턴스의 모든 직간접적으로 도달가능한 멤버들이 변경불가능으로서 분류되도록 시행하는 컴퓨터 실행가능 명령어들을 자동으로 작성하는 단계와,

생성 시에 상기 객체 형식의 인스턴스의 필드 할당을 허용하는 어떠한 참조도, 상기 인스턴스가 상기 인스턴스의 작성자에 대해 액세스가능하게 되는 시점을 넘어 존속하는 것이 허용되지 않도록 시행하는 컴퓨터 실행가능 명령어들을 자동으로 작성하는 단계

를 수행하는 단계를 포함하는 것인 시스템.

청구항 20

제19항에 있어서, 상기 변경불가능 객체 형식의 적어도 하나의 인스턴스는, 하나 이상의 기록가능 입력 값을 취하는 것인 시스템.

청구항 21

제20항에 있어서, 상기 하나 이상의 기록가능 입력 값 중 어느 것도 상기 적어도 하나의 인스턴스 중 임의의 인스턴스의 필드에 저장되지 않는 것인 시스템.

발명의 설명

기 술 분 야

배 경 기 술

[0001] 컴퓨팅 시스템들은 소프트웨어 프로그램들을 실행하는 것에 의해 고도의 기능을 달성한다. 프로그램들은 하드 드라이브, 콤팩트 디스크, 썸 드라이브(thumbdrive), 플래시 메모리 등에 어떤 영속적 형태로 유지되는 컴퓨터 실행가능 명령어들로 이루어져 있다. 실행 동안, 이러한 컴퓨터 실행가능 명령어들은 종종 랜덤 액세스 메모리에 로드되고, 컴퓨팅 시스템의 하나 이상의 프로세서들에 의해 실행되어, 컴퓨팅 시스템으로 하여금 작업들을 수행하게 할 수도 있다.

[0002] 객체 지향 프로그래밍(object-oriented programming)에서, 이 컴퓨터 실행가능 명령어들은 함수 호출을 통해 상호작용하고 하나 이상의 속성(property)들을 가질 수 있는 개체들로 구성된다. 관리 코드(managed code)는 형식 안전성(type safety)을 제공하지만, 메모리 관리 및 예외 처리도 제공할 수 있는 관리 환경(managed environment)에서 실행하는 코드이다. 관리 코드에서, 객체들은 무한한 수명을 갖고, 저장에 대한 제한이 없으며, 액세스 제한의 방법들은 간접 지정(indirection) 또는 추가 리소스를 필요로 한다.

[0003] 시스템 레벨 프로그래밍은 시스템 전체에 걸쳐 리소스(객체 등)에의 액세스 및 리소스의 수명의 엄격하고 효율적인 관리에 기초하고 있다. 이 엄격한 관리를 제공하는 하나의 통상의 방식은 리소스의 수명 및 액세스를 관리하기 위해 API(Application Program Interface)를 사용하는 것이다.

발명의 내용

[0004] 본 명세서에 설명된 적어도 일부 실시예들은 시스템 프로그래밍에서의 안전성을 향상시키는 언어 확장(language extension)에 관한 것이다. 언어 확장에 따라, 어떤 형식의 인스턴스들 모두가 변경불가능(immutable)인 경우에, 그 형식 전체가 변경불가능인 것으로 선언될 수도 있다. 변경불가능 형식 선언은 자동으로 그 형식의 임의의 인스턴스들을 변경불가능인 것으로 취급되게 하고, 자동으로 그 인스턴스의 모든 직간접적으로 도달가능한 멤버들(예컨대, 필드들, 메서드들, 속성들)을 또한 변경불가능인 것으로 취급되게 한다. 게다가, 인스턴스의 필드 할당(field assignment)을 가능하게 하는 임의의 생성시 참조(construction time reference)는 인스턴스가 작성자(creator)에 의해 액세스가능하게 되는 시점을 넘어 존속하도록 허용되지 않는다. 그에 따라, 이 인스턴스, 및 그 동일한 형식의 임의의 다른 인스턴스는 생성 시부터 변경불가능일 것이다.

[0005] 이러한 인스턴스들 모두를 변경불가능으로서 분류할 수 있다는 것은 유익한데, 그 이유는 변경불가능 특성(immutable characteristic)이 보통 리소스 액세스 안전성으로 인해 허용되지 않을 동작들을 가능하게 하기 때문이다. 예를 들어, 인스턴스가 다수의 컴포넌트들 및 다수의 스레드들 간에 공유될 수 있는데, 그 이유는 이 컴포넌트들 및 스레드들이 충돌하는 동작들을 인스턴스에 대해 어떻게든 수행할 위험이 없기 때문이다.

[0006] 이러한 개요는 청구된 주제의 핵심적인 특징들 또는 필수적인 특징들을 확인하기 위한 것이 아니며, 청구된 주제의 범위를 정하는 데 보조 수단으로 사용되기 위한 것도 아니다.

도면의 간단한 설명

[0007] 앞서 언급된 장점들 및 특징들과 기타 장점들 및 특징들이 달성될 수 있는 방식을 설명하기 위해, 다양한 실시예들의 보다 상세한 설명이 첨부 도면을 참조하여 이루어질 것이다. 이들 도면이 예시적인 실시예만을 도시하고 있고 따라서 본 발명의 범위를 제한하는 것으로 간주되어서는 안된다는 것을 이해하면서, 첨부 도면을 사용하여 실시예들이 보다 구체적이고 상세하게 기술되고 설명될 것이다.

도 1은 본 명세서에 설명된 일부 실시예들이 이용될 수 있는 컴퓨팅 시스템을 추상적으로 나타낸 도면.

도 2는 본 명세서에 설명된 실시예들이 이용될 수 있는 예시적인 환경을 나타내는 관리 코드 시스템(managed code system)을 추상적으로 나타낸 도면.

도 3은 본 명세서에 설명된 원리들에 따른, 변경불가능 형식 선언들이 행해질 수 있는 소스 코드의 저작 환경(authoring environment)을 포함하는 환경을 나타낸 도면.

도 4는 본 명세서에 설명된 실시예들에 따른, 객체 형식의 다수의 인스턴스들이 변경불가능하도록 보장하는 방법의 플로우차트를 나타낸 도면.

발명을 실시하기 위한 구체적인 내용

[0008] 본 명세서에 설명된 실시예들에 따르면, 형식 전체가 변경불가능인 것으로 선언될 수 있는 시스템 프로그래밍에서의 안전성을 향상시키는 언어 확장이 설명된다. 변경불가능 형식 선언은 자동으로 그 형식의 임의의 인스턴스들을 변경불가능인 것으로 취급되게 하고, 자동으로 그 인스턴스의 모든 직간접적으로 도달가능한 멤버들(예컨대, 필드들, 메서드들, 속성들)을 또한 변경불가능인 것으로 취급되게 한다. 게다가, 인스턴스에서의 필드 할당을 가능하게 하는 임의의 생성시 참조는 인스턴스가 작성자에 의해 액세스가능하게 되는 시점을 넘어 존속하도록 허용되지 않는다. 그에 따라, 이 인스턴스, 및 그 동일한 형식의 임의의 다른 인스턴스는 생성 시부터 변경불가능일 것이다. 이러한 인스턴스들 모두를 변경불가능으로서 분류할 수 있다는 것은 유익한데, 그 이유는 변경불가능 특성이 보통 리소스 액세스 안전성으로 인해 허용되지 않을 동작들을 가능하게 하기 때문이다. 예를 들어, 인스턴스가 다수의 컴포넌트들 및 다수의 스레드들 간에 공유될 수 있는데, 그 이유는 이 컴포넌트들 및 스레드들이 충돌하는 동작들을 인스턴스에 대해 어떻게든 수행할 위험이 없기 때문이다.

[0009] 컴퓨팅 시스템에 대한 어떤 서론적 논의가 도 1과 관련하여 설명될 것이다. 이어서, 관리 코드 시스템의 원리들이 도 2와 관련하여 설명될 것이다. 마지막으로, 변경불가능 형식들을 선언하는 언어 확장의 원리들이 도 3 및 도 4와 관련하여 설명될 것이다.

[0010] 컴퓨팅 시스템은 이제 점점 더 다양한 형태들을 취하고 있다. 컴퓨팅 시스템은, 예를 들어, 핸드헬드 디바이스, 가전 제품, 랩톱 컴퓨터, 데스크톱 컴퓨터, 메인 프레임, 분산 컴퓨팅 시스템, 또는 심지어 종래에 컴퓨팅 시스템으로 간주되지 않았던 디바이스일 수 있다. 이 설명 및 청구 범위에서, "컴퓨팅 시스템"이라는 용어는 적어도 하나의 물리적 및 유형적 프로세서, 그리고 프로세서에 의해 실행될 수 있는 컴퓨터 실행가능 명령어들을 가질 수 있는 물리적 및 유형적 메모리를 포함하는 임의의 디바이스 또는 시스템(또는 이들의 조합)을 포함하는 것으로 광의적으로 정의된다. 메모리는 임의의 형태를 취할 수 있고, 컴퓨팅 시스템의 성질 및 형태에 의존할 수 있다. 컴퓨팅 시스템은 네트워크 환경에 걸쳐 분산되어 있을 수 있고 다수의 구성 컴퓨팅 시스템(constituent computing system)들을 포함할 수 있다.

[0011] 도 1에 예시된 바와 같이, 가장 기본적인 구성에서, 컴퓨팅 시스템(100)은 전형적으로 적어도 하나의 처리 유닛(102) 및 메모리(104)를 포함한다. 메모리(104)는 휘발성, 비휘발성, 또는 이 둘의 어떤 조합일 수 있는 물리적 시스템 메모리일 수 있다. "메모리"라는 용어는 또한 본 명세서에서 물리적 저장 매체와 같은 비휘발성 대용량 저장소를 지칭하기 위해 사용될 수 있다. 컴퓨팅 시스템이 분산되어 있는 경우, 처리, 메모리 및/또는 저장 기능도 분산되어 있을 수 있다. 본 명세서에서 사용되는 바와 같이, "실행가능 모듈" 또는 "실행가능 컴포넌트"라는 용어는 컴퓨팅 시스템 상에서 실행될 수 있는 소프트웨어 객체(software object), 루틴 또는 메서드를 지칭할 수 있다. 본 명세서에 설명된 상이한 컴포넌트들, 모듈들, 엔진들 및 서비스들이 컴퓨팅 시스템 상에서 (예컨대, 개별적인 스레드들로서) 실행되는 객체들 또는 프로세스들로서 구현될 수 있다.

[0012] 이하의 설명에서, 하나 이상의 컴퓨팅 시스템들에 의해 수행되는 동작들을 참조하여 실시예들이 설명된다. 이러한 동작들이 소프트웨어로 구현되는 경우, 동작을 수행하는 관련 컴퓨팅 시스템의 하나 이상의 프로세서들은 컴퓨터 실행가능 명령어들을 실행한 것에 응답하여 컴퓨팅 시스템의 동작을 지시한다. 예를 들어, 이러한 컴퓨터 실행가능 명령어들은 컴퓨터 프로그램 제품을 형성하는 하나 이상의 컴퓨터 판독가능 매체 상에 구현될 수 있다. 이러한 동작의 일례는 데이터의 조작을 포함한다. 컴퓨터 실행가능 명령어들(및 조작된 데이터)는 컴퓨팅 시스템(100)의 메모리(104)에 저장될 수 있다. 컴퓨팅 시스템(100)은 또한 컴퓨팅 시스템(100)이, 예를 들어, 네트워크(110)를 통해 다른 메시지 프로세서들과 통신할 수 있게 하는 통신 채널들(108)을 포함할 수 있다.

[0013] 본 명세서에 설명된 실시예들은, 예를 들어, 이하에서 더 상세히 논의되는 바와 같이, 하나 이상의 프로세서들 및 시스템 메모리와 같은 컴퓨터 하드웨어를 포함하는 특수 목적 또는 범용 컴퓨터를 포함하거나 이용할 수 있다. 본 명세서에 설명된 실시예들은 또한 컴퓨터 실행가능 명령어들 및/또는 데이터 구조들을 전달 또는 저장하는 물리적 및 기타 컴퓨터 판독가능 매체를 포함한다. 이러한 컴퓨터 판독가능 매체는 범용 또는 특수 목적 컴퓨터 시스템에 의해 액세스될 수 있는 임의의 이용가능한 매체일 수 있다. 컴퓨터 실행가능 명령어들을 저장하는 컴퓨터 판독가능 매체는 물리적 저장 매체이다. 컴퓨터 실행가능 명령어들을 전달하는 컴퓨터 판독가능 매체는 전송 매체이다. 이와 같이, 제한이 아닌 예로서, 본 발명의 실시예들은 적어도 2 개의 뚜렷하게 상이한 종류의 컴퓨터 판독가능 매체 - 즉, 물리적 저장 매체 및 전송 매체 - 를 포함할 수 있다.

[0014] 컴퓨터 저장 매체는 컴퓨터 실행가능 명령어들 또는 데이터 구조들의 형태로 되어 있는 원하는 프로그램 코드 수단을 저장하는 데 사용될 수 있고 범용 또는 특수 목적 컴퓨터에 의해 액세스될 수 있는 RAM, ROM, EEPROM,

CD-ROM 또는 기타 광학 디스크 저장소, 자기 디스크 저장소 또는 기타 자기 저장 디바이스, 또는 임의의 다른 매체를 포함한다.

[0015] "네트워크"는 컴퓨터 시스템들 및/또는 모듈들 및/또는 기타 전자 디바이스들 간의 전자 데이터의 전송을 가능하게 하는 하나 이상의 데이터 링크들로서 정의된다. 정보가 네트워크 또는 다른 통신 연결(유선, 무선 또는 유선 또는 무선의 조합)을 통해 컴퓨터로 전송되거나 제공될 때, 컴퓨터는 적절하게도 그 연결을 전송 매체로 본다. 전송 매체는 컴퓨터 실행가능 명령어들 또는 데이터 구조들의 형태로 되어 있는 원하는 프로그램 코드 수단을 전달하는 데 사용될 수 있고 범용 또는 특수 목적 컴퓨터에 의해 액세스될 수 있는 네트워크 및/또는 데이터 링크들을 포함할 수 있다. 상기한 것들의 조합들도 역시 컴퓨터 판독가능 매체의 범위 내에 포함되어야 한다.

[0016] 게다가, 다양한 컴퓨터 시스템 구성요소들에 도달할 때, 컴퓨터 실행가능 명령어들 또는 데이터 구조들의 형태로 되어 있는 프로그램 코드 수단이 전송 매체로부터 컴퓨터 저장 매체로(또는 그 반대로) 자동으로 전달될 수 있다. 예를 들어, 네트워크 또는 데이터 링크를 통해 수신되는 컴퓨터 실행가능 명령어들 또는 데이터 구조들이 네트워크 인터페이스 모듈(예컨대, "NIC") 내의 RAM에 버퍼링될 수 있고, 이어서 궁극적으로 컴퓨터 시스템 RAM으로 및/또는 컴퓨터 시스템에 있는 저휘발성(less volatile) 컴퓨터 저장 매체로 전달될 수 있다. 따라서, 컴퓨터 저장 매체가 또한(또는 심지어 주로) 전송 매체를 이용하는 컴퓨터 시스템 구성요소들에 포함될 수 있다는 것을 잘 알 것이다.

[0017] 컴퓨터 실행가능 명령어들은, 프로세서에서 실행될 때, 예를 들어, 범용 컴퓨터, 특수 목적 컴퓨터, 또는 특수 목적 처리 디바이스로 하여금 특정의 기능 또는 일군의 기능들을 수행하게 하는 명령어들 및 데이터를 포함한다. 컴퓨터 실행가능 명령어들은, 예를 들어, 바이너리(binary), 어셈블리 언어와 같은 중간 포맷 명령어(intermediate format instruction), 또는 심지어 소스 코드일 수 있다. 주제가 구조적 특징들 및/또는 방법 동작들에 대한 특유한 언어로 설명되어 있지만, 첨부된 청구범위에서 규정된 주제가 전술한 바와 같이 설명된 특징들 또는 동작들로 꼭 제한되는 것은 아님을 잘 알 것이다. 오히려, 설명된 특징들 및 동작들은 청구항들을 구현하는 예시적인 형태들로서 개시되어 있다.

[0018] 당업자라면 본 발명이 개인용 컴퓨터, 데스크톱 컴퓨터, 랩톱 컴퓨터, 메시지 프로세서, 핸드헬드 디바이스, 멀티프로세서 시스템, 마이크로프로세서 기반 또는 프로그램가능 가전 제품, 네트워크 PC, 미니컴퓨터, 메인프레임 컴퓨터, 이동 전화, PDA, 페이저, 라우터, 스위치 등을 비롯한 많은 유형의 컴퓨터 시스템 구성들을 갖는 네트워크 컴퓨팅 환경들에서 실시될 수 있다는 것을 알 수 있다. 본 발명은 또한 네트워크를 통해 (유선 데이터 링크, 무선 데이터 링크에 의해, 또는 유선 및 무선 데이터 링크들의 조합에 의해) 연결되어 있는 로컬 및 원격 컴퓨터 시스템들 양자가 작업들을 수행하는 분산 시스템 환경들에서 실시될 수 있다. 분산 시스템 환경에서, 프로그램 모듈들은 로컬 및 원격 메모리 저장 디바이스들 양자에 위치될 수 있다.

[0019] 도 2는 본 명세서에 설명된 원리들이 동작할 수 있는 환경(200)을 나타낸 것이다. 환경(200)은 객체 그래프(object graph)들의 격리(isolation) 및 변경가능성(mutability)을 추적하는 프레임워크(framework)(210)를 포함한다. 프레임워크(210)는 참조에 대한 액세스를 제어하는 것에 의해 객체 그래프들에 대한 다양한 참조들(221)을 구성한다. 참조는 지역 변수, 메서드 파라미터, 객체 필드, 또는 객체 그래프에 대한 임의의 다른 참조일 수 있다. 예시된 실시예에서, 참조들(221)은 참조들(221A 내지 221E)을 포함하는 것으로 예시되어 있지만, 생략부호(221F)는 프레임워크(210)가 임의의 수의 참조들(220)에 대한 액세스를 관리할 수 있다는 것을 기호에 의해 나타내고 있다.

[0020] 프레임워크(210)는 참조들에 퍼미션(permission)들을 할당하는 것에 의해 참조들(221)에 대한 액세스를 관리한다. "퍼미션"은 참조가 변경(mutate)될 수 있는지에 관한 어떤 속성을 나타내는 참조에 대한 주석(annotation)이다. 이러한 퍼미션들은 도 2에서의 퍼미션들(211)에 의해 추상적으로 표현된다. 퍼미션들은 판독가능(readable) 퍼미션(211A), 기록가능(writable) 퍼미션(211B), 및 변경불가능(immutable) 퍼미션(211C)을 포함한다.

[0021] "판독가능" 퍼미션(211A)은 대응하는 객체(및 그의 직간접적으로 도달가능한 멤버들 전부)로부터 단지 판독될 수 있을 뿐이라는 것을 의미한다.

[0022] "기록가능" 퍼미션(211B)은 대응하는 객체에 기록할 수 있다는 것을 의미한다.

[0023] "변경불가능" 퍼미션(211C)은 판독가능 퍼미션(211A)과 같지만, 아무도 이 객체에 대한 기록가능 참조를 갖지 않는다는 것을 추가로 보장한다. 변경불가능 퍼미션(211C)은 이 객체에 대한 기록가능 참조가 다시는 없을 것

임을 추가로 보장할 수 있다. 이와 같이, 변경불가능 퍼미션(211C)은 대응하는 객체에 결코 기록하지 않을 것임(및 그의 직간접적으로 도달가능한 멤버들 전부에 결코 기록하지 않을 것임)과, 이와 유사하게, 그의 필드들 전부 및 그의 필드의 필드들 모두, 기타 등등에 결코 기록하지 않을 것임을 의미한다. 객체 내의 모든 정적 필드들은 프레임워크(210)에 의해 변경불가능 퍼미션(211C)을 갖는 것으로 취급된다.

[0024] "프레쉬(fresh)" 퍼미션(211D)은 1) 반환된 참조에 의해 참조되는 특정의 객체 그래프(또는 특정의 객체 그래프 내의 임의의 객체)에 대한 외부 참조(external reference)가 없다는 것과, 2) 객체 그래프 외부에 있는 임의의 객체들에 대한 변경가능 참조(mutable reference)가 객체 그래프 내부에 없다는 것을 의미한다. "생성되지 않음(not constructed)" 퍼미션(211E)은 이하에서 더 상세히 기술될 것이다.

[0025] 프레임워크(210)는 또한 메모리 장소들을 격리된 것으로 주석 첨부하는 격리된 메모리 관리자(isolated memory manager)(212)를 포함한다. 저장 장소에 대한 "격리됨(isolated)" 주석은 그 장소가 외부적으로 고유한(externally unique) 값을 저장한다는 것을 의미한다. 즉, 그 장소에 저장된 임의의 객체에 대해, 시스템에 그 객체에 대한 외부 참조가 없고, 그 장소에 저장된 객체의 직간접적으로 도달가능한 멤버들 중 임의의 것에 대한 어떤 참조도 없다. 예를 들어, 격리된 메모리 관리자(212)는 격리된 메모리 장소들(222)을 관리한다. 격리된 메모리 장소들이 두 개의 격리된 메모리 장소들(222A 및 222B)을 포함하는 것으로 예시되어 있지만, 생략부호(222C)는 격리된 메모리 장소들(222)이 임의의 수의 격리된 메모리 장소들을 포함할 수 있다는 것을 나타낸다.

[0026] 주석 컴포넌트(201)는 프레임워크(210) 상에 구축된다. 다른 대안으로서 또는 그에 부가하여, 주석 컴포넌트(201)는 프레임워크(201)의 일부로서 동작할 수 있다. 주석 컴포넌트(201)는 도 2의 참조들(221) 중 하나와 같은 참조를 반환하는 실행가능 컴포넌트를 평가하도록 구성되어 있다.

[0027] 본 설명은 생성이 완료되면 항상 변경불가능한 형식을 어떻게 표현할지에 관한 참조 퍼미션들을 처리하는 시스템에서의 과제를 다룬다. 이 형식들의 필드들은 변경불가능하고 판독 전용이며, 기록가능 참조를 통해서도 변경될 수 없다. 변경이 행해질 수 있는 메커니즘이 전혀 없다. 이후부터, 이 형식들은 "변경불가능 형식"이라고 지칭될 것이고, 그 외의 모든 것은 '변경가능 형식'이라고 지칭될 것이다.

[0028] 변경불가능 형식은 시스템 프로그래밍에서 흔한데, 그 이유는 상이한 컴포넌트들 및 상이한 스레드들 간에 자유롭게 공유될 수 있는 데이터를 표현하기 때문이다. 하나의 컴포넌트가, 효과가 다른 컴포넌트에게 보이도록, 데이터를 변경할 수 있는 것 이외에는 위험이 없다.

[0029] 변경불가능 형식들을 분류함에 있어서의 제1 문제점은 변경불가능 형식들이 종종 그들의 생성자에 대해 "기록가능"으로 분류된 입력 파라미터들을 받는다는 것이다. 그렇지만, 기록가능 입력은, 생성 중인 인스턴스 외에는, 필드에 저장되지 않고, 따라서 전체로서의 객체 형식이 변경불가능인 것으로 더 이상 간주되지 않을 수 있고, 그 대신에 변경가능 형식으로 될 것이다. 그렇지만, 변경불가능 형식이 기록가능 입력 파라미터에 기초하여 어떤 변경불가능 데이터를 계산하고 이어서 그 최종 변경불가능 값을 필드로서 저장할 수 있다. 이와 같이, 변경불가능 형식은 대응하는 인스턴스가 생성되는 시점부터 그의 변경불가능 특성을 유지한다.

[0030] 생성자에의 입력이 변경불가능 또는 원래 상태 입력만을 포함하는 경우, 객체는 "변경불가능"으로 승격가능(promotable)할 수 있다. 그렇지만, 이 규칙으로 인해, 모든 변경불가능 객체들이 변경불가능으로 분류되지는 않는다. 예를 들어, 이하의 의사 코드 예를 생각해보자:

[0031] public class Container

[0032] {

[0033] readonly immutable string m_name;

[0034] readonly immutable string m_address;

[0035] public Container(writable Student student)

[0036] {

[0037] m_name = student.Name;

[0038] m_address = student.Address;

[0039] }

[0040] }

[0041] 이 예에서, Container는 생성 후에 실제로 변경불가능이지만, 형식 Container는 기록가능인 것으로 선언되는 Student라고 불리우는 입력 파라미터를 받아들인다. 따라서, 규칙(생성자에의 입력이 변경불가능 또는 원래 상태 퍼미션을 갖는 입력 파라미터들만을 포함하는 경우, 객체는 "변경불가능"으로 승격될 수 있음)은, 이 형식이 기록가능 입력을 받아들이기 때문에, 이 형식이 변경불가능으로 승격될 자격을 박탈할 것이다. 이는 판독가능의 최대 퍼미션을 허용할 것이다.

[0042] 제2 문제점은 변경불가능 형식 및 비변경불가능(non-immutable) 형식을 위한 공동 저장소를 정의하는 것이다. 프로그래밍에서, 저장된 값이 원래의 상태로 검색될 수 있도록 임의의 객체에 대한 저장소일 수 있는 장소를 정의하는 것이 유리하다. 저장 장소가 변경가능 형식 및 변경불가능 형식을 받아들일 필요가 있을 때 이것은 깨지는데, 그 이유는 양 형식에 대해 기능하는 가능한 퍼미션이 없기 때문이다. 이하의 의사 코드 예를 생각해보자:

[0043] writable Student student = ... ;

[0044] immutable string name = ... ;

[0045] ??? object storage;

[0046] if (condition) {

[0047] storage = name;

[0048] }

[0049] else {

[0050] storage = student;

[0051] }

[0052] 이 예에서, "student" 및 "name" 양자로부터의 할당(assignment)을 허용하고 그들이 그들의 원래의 퍼미션으로 검색될 수 있게 할 "storage" 로캘(locale)에 대한 퍼미션을 선택할 방법이 없다. 모든 퍼미션이 문제가 있다. 예를 들어, "기록가능" 퍼미션은 "name"이 할당되지 못하게 할 것인데, 그 이유는 변경불가능 참조가 "기록가능"으로 표시된 참조에 저장될 수 없기 때문이다. "변경불가능" 퍼미션은 "student"가 할당되지 못하게 할 것인데, 그 이유는 기록가능 참조가 변경불가능 장소에 저장될 수 없기 때문이다. "판독가능" 퍼미션은 "storage"에의 할당 및 그로부터의 검색을 못하게 할 것인데, 그 이유는 할당된 값 및 검색된 값이 원래 가졌던 것과는 상이한 퍼미션을 가질 것이기 때문이다.

[0053] 본 명세서에 설명된 원리들은 변경불가능 형식의 개념을 도입함으로써 이들 문제점을 해결한다. 이것은, 형식의 모든 인스턴스들이 변경불가능이라는 것을 나타내기 위해, 사용자가 형식 선언(type declaration)에 추가하는 명시적 주석이다. 예를 들어, 본 명세서에 설명된 원리들이 변경불가능 형식이 선언되는 방식으로 제한되지 않지만, 이하의 예시적인 의사 코드를 생각해보자:

[0054] immutable class String {

[0055] ...

[0056] }

[0057] 도 3은 본 명세서에 설명된 원리들이 이용될 수 있는 환경(300)을 나타낸 것이다. 저작 프로세스(authoring process)(310)(저작 프로그램 등)는 프로그래밍 또는 다른 프로그래밍 엔티티(소프트웨어 등)가 변경불가능 형식 선언(312)(그의 일례가 바로 위에 나타내어져 있음)을 포함하는 코드(311)[소스 코드 또는 중간 언어 코드(intermediate language code) 등]를 생성할 수 있게 한다. 컴파일러(320)는 이어서 코드(311)를 컴파일하고, 그러자 곧 변경불가능 형식 선언(312)을 만난다.

[0058] 도 4는 객체 형식의 다수의 인스턴스들이 변경불가능하도록 보장하는 방법(400)의 플로우차트를 나타낸 것이다. 방법(400)은 객체 형식이 변경불가능이라는 선언에 액세스[동작(401)]할 시에 컴파일러에 의해 수행될 수 있다. 예를 들어, 도 3을 참조하면, 방법(400)은 컴파일러(320)가 변경불가능 형식 선언(312)에 액세스할 시에 수행될 수 있다.

[0059] 그 선언에 응답하여, 컴파일러는 그 객체 형식의 각각의 인스턴스가 변경불가능으로서 분류되도록 컴퓨터 실행가능 명령어들을 자동으로 작성한다[동작(411)]. 이것은 변경불가능 형식의 다른 인스턴스들 모두를 암시적으로 "변경불가능"으로 되게 한다. 게다가, 그 객체 형식의 인스턴스의 모든 직간접적으로 도달가능한 멤버들이 변경불가능으로서 분류된다[동작(412)]. 예를 들어, 인스턴스에 의해 직간접적으로 도달가능한 모든 형식들, 필드들, 메서드들 및 속성들이 암시적으로 변경불가능 퍼미션으로 태깅된다. 그에 부가하여, 생성 시에 그 객체의 인스턴스의 필드 할당을 가능하게 하는 임의의 참조는 인스턴스가 작성자에 의해 액세스가능하게 되는 시점을 넘어 존속하도록 허용되지 않는다[동작(413)]. 일례로서, 생성자에서의 "this"의 퍼미션은 암시적으로 "생성되지 않음"이라고 불리우는 퍼미션을 할당받는다. 이 퍼미션은 도 2에서 "생성되지 않음" 퍼미션(211E)으로서 예시되어 있다.

[0060] 변경불가능 형식들의 주된 속성은 그들이 생성 동안 그들의 상태를 변경하도록 허용되어 있지만, 일단 생성이 완료되면, 변경불가능 형식으로부터 생성된 임의의 인스턴스가 다시는 변경될 수 없다(따라서 변경불가능임)는 것이다. 이하의 생성후 규칙(post construction rule)들이 이어서 시행된다: 1) 필드들이 변경불가능이고, 따라서 그 자체가 변할 수 없고, 2) 변경불가능 형식의 모든 생성된 인스턴스들이 변경불가능이고, 따라서 그들의 필드들을 재할당할 수 없다(필드를 재할당 또는 변경하려고 시도하는 임의의 멤버는 컴파일 시에 포착될 것임).

[0061] 생성 동안의 변경은 "생성되지 않음" 퍼미션(211E)(도 2 참고)으로 달성된다. 이 퍼미션은 필드들을 목표 참조(targeted reference)로부터 읽는 것 및 그에 쓰는 것을 가능하게 한다. 불변성(immutability)을 갖는 불변식(invariant)들을 유지하기 위해, 필드 할당을 가능하게 하는 "this" 참조가 생성자보다 오래 존속하도록 허용될 수 없다. 이는, 참조가 기록가능 입력에 대한 "생성되지 않음" 퍼미션으로 저장되고 생성자가 완료된 후에 객체를 추가로 변경하기 위해 사용될 수 있는 경우, 문제를 야기할 것이다. "this" 값이, 생성자가 완료될 때까지, 완전히 생성된 Address 객체로서 보이지 않을 수 있다. 이는 Address 객체가, 여전히 변경되고 있더라도, 변경불가능인 것처럼 가장할 수 있게 할 것이다. 이하의 의사 코드 예를 생각해보자:

[0062] public immutable class Address

[0063] {

[0064] public Address(out notconstructed Address p)

[0065] {

[0066] p = this;

[0067] Address address = this;

[0068] }

[0069] }

[0070] "this"로부터의 상기 할당들 중 어느 하나가 적법한 경우, 이는 완전히 생성된 Address 인스턴스가 변경 중인 것으로 보이게 할 수 있다. 이 "생성되지 않음" 퍼미션은 어느 하나의 할당을 못하게 한다. 이 퍼미션은 사용자에 의해 서술가능(statable)하지 않고, 따라서 어떤 저장 장소에도 나타날 수 없다. 의사 코드 예에서의 상기 파라미터 선언이 컴파일러 관점에서는 완전히 불법적인 것이다. "생성되지 않음" 퍼미션은 또한 임의의 다른 퍼미션 형식으로(심지어 판독가능으로도) 변환가능하지 않다. 이것은 그것이 생성자보다 오래 존속하게 하지 않을 것임을 보장하는데, 그 이유는 그것을 저장하는 장소를 정의하는 것이 가능하지 않기 때문이다.

[0071] 이 제한들은, 변경불가능 형식이 생성을 끝내면, 그가 항상 변경불가능이라는 것을 의미한다. 따라서, 생성된 변경불가능 형식의 모든 인스턴스들은 암시적으로 변경불가능 퍼미션으로 태깅된다. 기록가능을 비롯한, 사용자가 형식에 부여하는 임의의 퍼미션이 형식 시스템에 의해 완전히 무시된다. 변경이 전혀 가능하지 않으며, 따라서 퍼미션이 더 이상 의미가 없다.

[0072] 인스턴스가 결코 변경을 야기할 수 없기 때문에, 변경불가능 형식의 인스턴스들은 이제 변경을 위한 것으로 표시되어 있는 장소들에 저장되도록 허용될 수 있다. 이하의 의사 코드 예를 생각해보자:

[0073] string s1 = "test";

[0074] immutable string s2 = s1;

[0075] writable string s3 = s1;

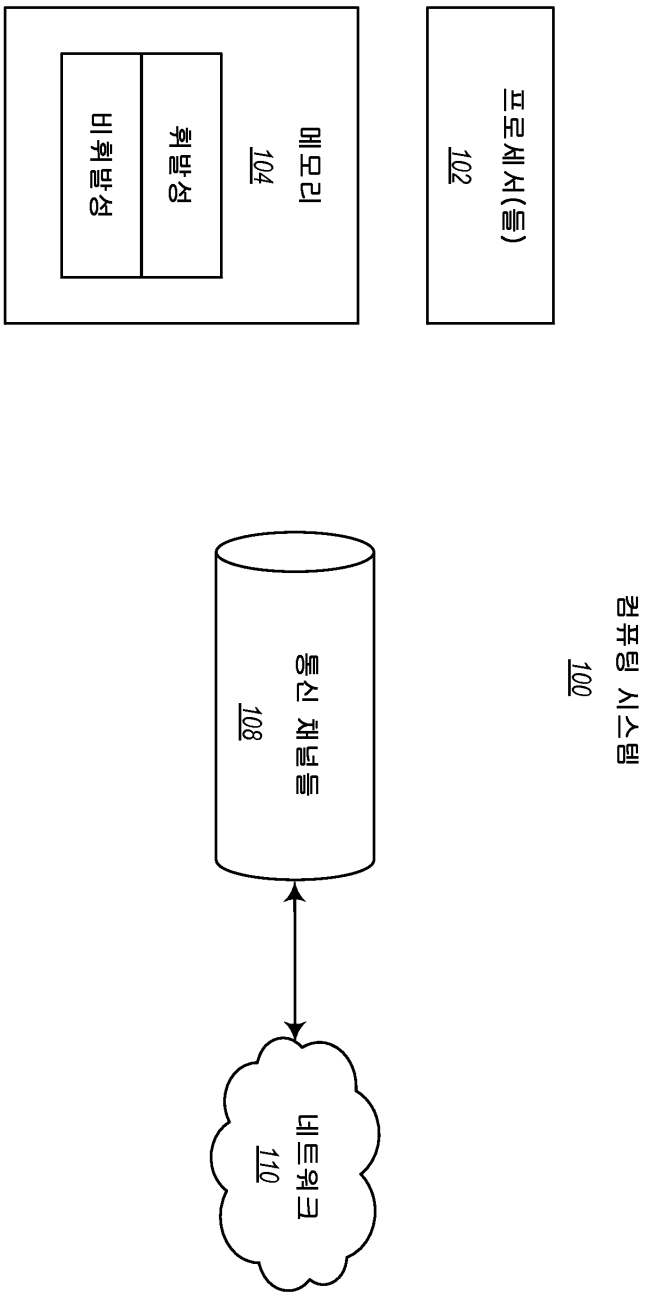
[0076] writable object o1 = s1;

[0077] 이와 같이, 이것은, 다시 말하지만, 변경불가능 형식 및 변경가능 형식의 공동 저장을 가능하게 한다. 게다가, 그렇지 않았으면 변경불가능인 것으로 간주되지 않을 수 있는 객체들이 이제 안전하게 변경불가능인 것으로 분류될 수 있어, 인스턴스들이 상이한 프로세스들 간에, 그 프로세스들이 변경불가능 객체의 사용과 관련하여 서로를 방해하는 것에 관한 걱정 없이, 안전하게 공유될 수 있게 한다.

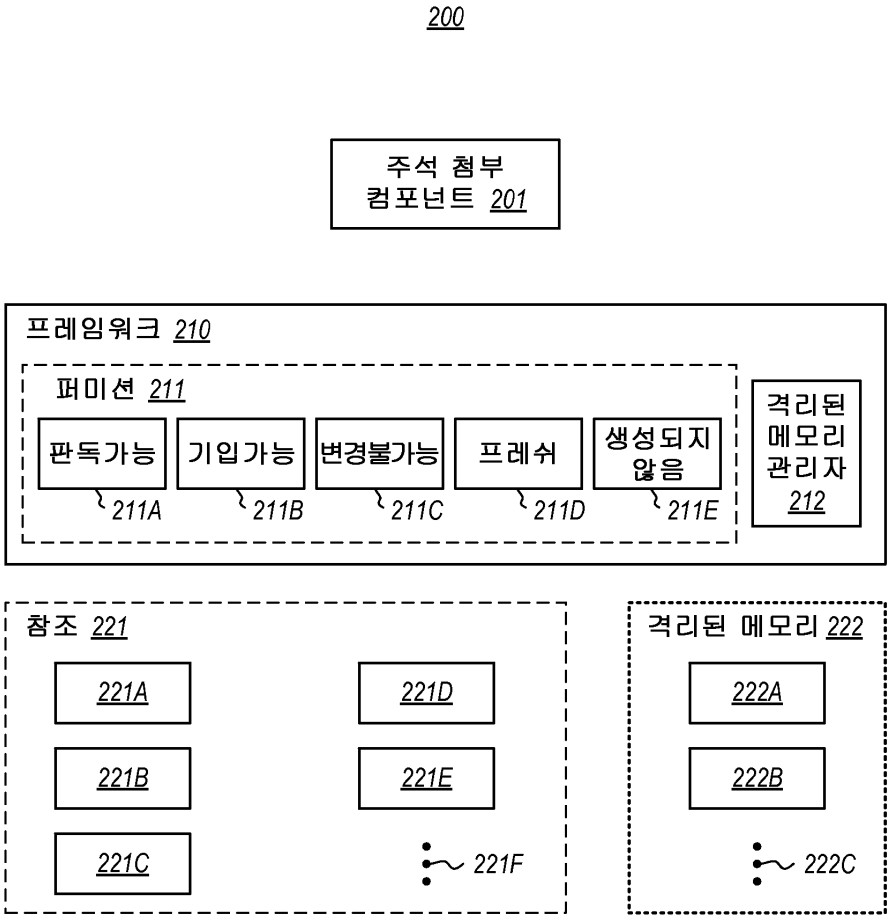
[0078] 본 발명은 그의 사상 또는 본질적인 특성들을 벗어나지 않고 다른 구체적인 형태들로 구현될 수도 있다. 설명된 실시예들이 모든 점에서 제한적이 아니라 단지 예시적인 것으로 간주되어야 한다. 따라서, 본 발명의 범위는 이상의 설명이 아니라 첨부된 청구범위에 의해 나타내어진다. 청구범위의 등가성의 의미 및 범위 내에 속하는 모든 변경들이 청구범위의 범위 내에 포함되어야 한다.

도면

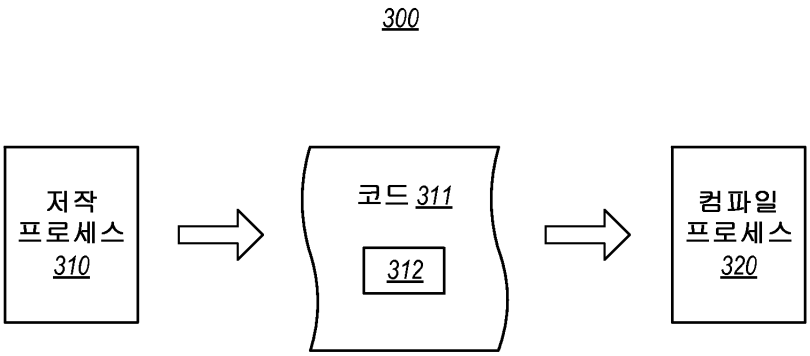
도면1



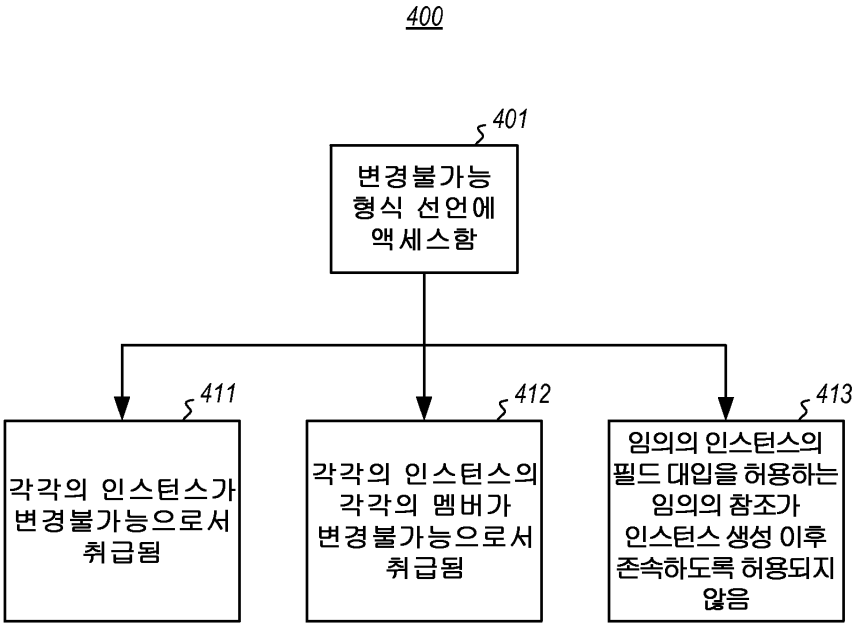
도면2


200
주석 첨부
컴포넌트 201
프레임워크 210
퍼미션 211
판독가능
기입가능
변경불가능
프레쉬
생성되지 않음
211A
211B
211C
211D
211E
격리된 메모리 관리자 212
참조 221
221A
221B
221C
221D
221E
221F
격리된 메모리 222
222A
222B
222C

도면3


300
저작 프로세스 310
코드 311
312
컴파일 프로세스 320

도면4


400
401
변경불가능 형식 선언에 액세스함
411
각각의 인스턴스가 변경불가능으로서 취급됨
412
각각의 인스턴스의 각각의 멤버가 변경불가능으로서 취급됨
413
임의의 인스턴스의 필드 대입을 허용하는 임의의 참조가 인스턴스 생성 이후 존속하도록 허용되지 않음