(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 November 2013 (14.11.2013)

WIPO | PCT

(10) International Publication Number
WO 2013/169554 A1

(51) International Patent Classification:
G06F 9/44 (2006.01)

(21) International Application Number:
PCT/US2013/039256

(22) International Filing Date:
2 May 2013 (02.05.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/643,826 7 May 2012 (07.05.2012) US
13/783,024 1 March 2013 (01.03.2013) US

(71) Applicant: **ORACLE INTERNATIONAL CORPORATION** [US/US]; 500 Oracle Parkway, M/S 5op7, Redwood Shores, CA 94065 (US).

(72) Inventor: **OLIVER, Brian**; 11 Minot Avenue, Acton, MA 01720 (US).

(74) Agents: **MEYER, Sheldon, R.** et al.; Fliesler Meyer LLP, 650 California Street, Fourteenth Floor, San Francisco, CA 94108 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, 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, JP, 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, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, 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) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, 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, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

(54) Title: SYSTEM AND METHOD FOR SUPPORTING A DEFERRED REFERENCE TO AN OBJECT IN AN OBJECTED-ORIENTED PROGRAMMING LANGUAGE ENVIRONMENT

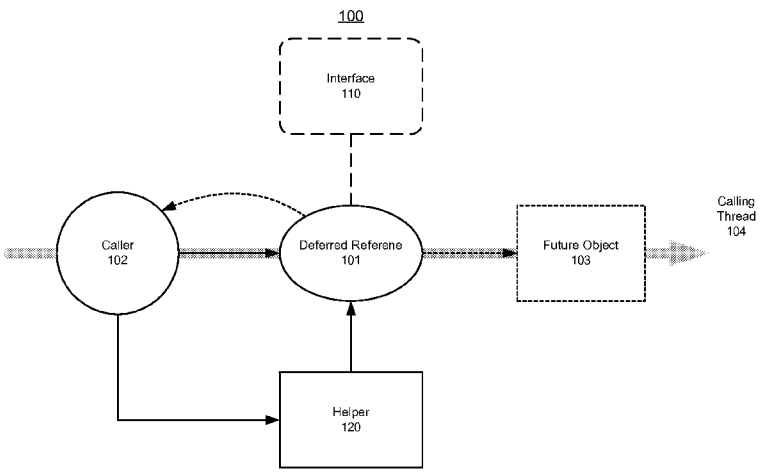

FIGURE 1

(57) Abstract: A system and method can support a deferred reference in an object-oriented programming language environment. The object-oriented programming language environment can include an interface that can provide a deferred reference to an object that will be available at a future time. Furthermore, after receiving a request from a caller to get the object based on the deferred reference, the interface can return the object to the caller when the object is available or can indicate to the caller when the object is not currently available and/or when the object will never become available.

WO 2013/169554 A1

SYSTEM AND METHOD FOR SUPPORTING A DEFERRED REFERENCE TO AN OBJECT IN AN OBJECTED-ORIENTED PROGRAMMING LANGUAGE ENVIRONMENT

Copyright Notice:

[0001] A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

Field of Invention:

[0002] The present invention is generally related to computer systems and software, and is particularly related to supporting an object-oriented programming language environment.

Background:

[0003] Object-oriented programming (OOP) can be used to design applications and computer programs based on objects, which are usually instances of a class. The OOP techniques can include features such as data abstraction, encapsulation, messaging, modularity, polymorphism, and inheritance. Each object, which can have a distinct role or responsibility, is capable of receiving messages, processing data, and sending messages to other objects. Additionally, these objects can be closely associated with various actions, or methods. This is the general area that embodiments of the invention are intended to address.

Summary:

[0004] Described herein are systems and methods that can support a deferred reference in an object-oriented programming language environment. The object-oriented programming language environment can include an interface that can provide a deferred reference to an object that will be available at a future time. Furthermore, after receiving a request from a caller to get the object based on the deferred reference, the interface can return the object to the caller when the object is available or can indcate to the caller when the object is not currently available and/or when the object will never become available.

[0005] Also described herein is a system for supporting a deferred reference in an object-oriented programming language environment operating on one or more microprocessors. The system comprises means for providing a deferred reference to an object that will be available at a future time, and means for performing, after receiving a request from a caller to get the object based on the deferred reference, one of: returning the object to the caller when the object is available, and indicating to the caller at least one of when the object is not currently available and when the object will never become available.

[0006] Furthermore, described herein is a system for supporting a deferred reference in an

object-oriented programming language environment. The system comprises an interface configured to provide a deferred reference to an object that will be available at a future time and a caller configured to send a request to the interface to get the object based on the deferred reference. The interface, after receiving the request from the caller, is further configured to perform one of: returning the object to the caller when the object is available, and indicating to the caller at least one of when the object is not currently available and when the object will never become available.

Brief Description of the Figures:

[0007] **Figure 1** shows an illustration of supporting a deferred reference in an object-oriented programming (OOP) environment in accordance with various embodiments of the invention.

[0008] **Figure 2** shows an illustration of an exemplary interface for defining a deferred type in an OOP environment in accordance with an embodiment of the invention.

[0009] **Figure 3** shows an illustration of an exemplary software pseudo code for acquiring management bean information from a server, in accordance with an embodiment of the invention.

[00010] **Figure 4** illustrates an exemplary flow chart for supporting a deferred reference in an OOP environment in accordance with an embodiment of the invention.

[00011] **Figure 5** illustrates an exemplary system for supporting a deferred reference in an OOP environment in accordance with an embodiment of the invention.

[00012] **Figure 6** illstrates a functional block diagram to show features in accordance with an embodiment of the invention.

Detailed Description:

[00013] Described herein are systems and methods that can support a deferred reference in an object-oriented programming (OOP) environment.

[00014] A reference object can encapsulate a reference to another object, or a referent, so that the reference itself can be examined and manipulated like the referent object. A deferred reference represents a referent that may not exist yet, may not be available or will only become available at some point in the future.

[00015] **Figure 1** shows an illustration of supporting a deferred reference in an OOP environment in accordance with various embodiments of the invention. As shown in Figure 1, an OOP environment 100, e.g. a JAVA programming environment, can provide a differed reference 101 to a future object 103. The differed reference 101 can be based on an interface 110, which can define a deferred type in the OOP environment 100. The future object 103 can be an object, e.g. a Server, a Member, a Connection, an Mbean, a Value, and a Condition, that will be available at a future time.

[00016] In the OOP environment 100, a caller 102 can send a request, e.g. a get method, to

obtain the future object 103 based on the deferred reference 101. The computation to resolve the referent, or the future object 102, may not start until the get method is called. On the other hand, the return of the deferred reference 101 to the caller 102 happens immediately.

[00017] In accordance with the various embodiments of the invention, the caller 102 can be implemented on a general purpose computer which includes well-known hardware elements such as a processor, a memory and a communication interface. The caller 102 is realized when instructions of a software program are executed by the processor.

[00018] After receiving the request from the caller 102, the OOP environment 100 can provide an indication to the caller 102 that the future object 103 is not currently available. Alternatively, the OOP environment can return the future object 103 to the caller 102 as soon as it becomes available.

[00019] In accordance with various embodiments of the invention, the OOP environment can avoid blocking a calling thread 104 that the caller 102 uses to request for the future object 102, while waiting for the future object 102 to become available. Such an asynchronized model can be beneficial, since the calling thread 104 can be responsible for other important tasks.

[00020] Furthermore, the OOP environment 100 can use one or more helpers 120 to extend the functionalities of a deferred reference 101. Also, the OOP environment 100 allows for nesting another deferred reference into the deferred reference 101.

[00021] In accordance with various embodiments of the invention, the OOP environment can provide a future class definition, e.g. a JAVA Future class, which can represent the result of an asynchronous operation that has been requested.

[00022] Using the deferred reference 101 of a future object 102, the OOP environment can avoid various problems associated with the JAVA Future class. For example, the computation to resolve the value may start when the future class is created. Furthermore, the future classes may force the calling threads to wait. For example, the JAVA Future classes can provide a Future.get() method that must block the calling thread to wait forever until a value is produced, or a Future.get(time) method that blocks calling thread for a specified amount of time. Both methods force developers to write blocking algorithms, which are not asynchronous.

[00023] In accordance with various embodiments of the invention, the deferred reference 102 can be enabled as part of a software package, such as part of a com.oracle.tools.deferred package in Oracle Coherence Incubator Common, or be enabled in a separate jar for wider uses.

[00024] Additionally, the OOP environment 100 can provide a mechanism for an application to be notified when a deferred object becomes available. This allows developers to define one or more deferred references together with callbacks to be notified when the underlying objects become available. Thus, the underlying object-oriented programming language can inform the application developer when objects are no longer "deferred".

[00025] **Figure 2** shows an illustration of an exemplary interface for defining a deferred type in an OOP environment in accordance with an embodiment of the invention. As shown in Figure 2, an interface Deferred can be used, or implemented, to provide a deferred reference to an object that will be available at a future time (Line 1). The interface Deferred includes a get() method that can returns the deferred reference (Line 3), and a getDeferredclass() method that can determine the class of the referent without actually existing (Line 5).

[00026] When the get() method is invoked by a caller, the object-oriented programming language environment can avoid blocking the calling thread, while waiting for the future object to become available. Unlike the JAVA Future classes, which waits for the result value to be produced, the get() method in the interface Deferred returns a reference to the future object imediately.

[00027] Furthermore, the OOP environment can return a NULL value or throw a runtime exception if the object is not currently available. Also, the object-oriented programming language environment can throw an object not available exception (e.g. an ObjectNotAvailableException) if the object may never become available.

[00028] In accordance with various embodiments of the invention, the OOP environment can provide different types of deferred helpers for extending the functionalities of a deferred reference.

[00029] For example, a deferred helper, which is shown in the following, can ensure that the caller waits a certain period of time for the object to become available, or retries after a certain period of time.

```
Deferred<T> deferred = ...
Deferred<T> ensured = new Ensured(deferred, timeout);
```

[00030] In the OOP environment, an ensured Deferred type can use a deferred implementation (wrapper) that waits a certain amount of time for an object in a Deferred type to become available. Here, any Deferred type can be ensured. The ensured Deferred type allows the nested Deferred types without a need of nested timeouts.

[00031] Furthermore, the Deferred types do not implement blocking semantics, while only Ensured types do. The ensured Deferred allows "waiting" with customizable semantics, while the Deferred types may never block the calling thread. For example, the Deferred types can throw an (runtime) ObjectNotAvailableExceptions when the referents are assumed never to become available.

[00032] As shown in the following, the OOP environment can resolve a deferred value by waiting a maximum (default) period of time, or retrying after a default period of time.

```
public <T> T ensure(Deferred<T> deferred);
```

[00033] Additionally, the object-oriented programming language environment allows the caller to specify an explicit maximum time out.

```
public <T> T ensure(Deferred<T> deferred, long duration, TimeUnit unit);
```

[00034] In accordance with an embodiment of the invention, the OOP environment can support deferred method invocation based on the concept of deferred helpers.

[00035] For example, as shown in the following, a recording dynamic proxy of an interface and/or object can be created.

```
public <T> T invoking(T t);
```

[00036] Thus, the OOP environment can obtain a "recording" dynamic proxy of a Deferred type, deferred.

```
public <T> T invoking(Deferred<T> deferred);
```

[00037] Additionally, the OOP environment can obtain a Deferred type representing the last recorded invoking call as shown in the following.

```
public <T> Deferred<T> eventually(T t);
```

[00038] Finally, the OOP environment can obtain a deferred type, Deferred<Integer>, that represents the last recorded invoking call to list.size.

```
Deferred<Integer> defSize = eventually(invoking(list).size());
```

[00039] Thus, a value concerning the last recorded dynamic proxy of the object can be returned to a caller as a deferred value, e.g. a deferred integer can represent the size of a list at a future time. Additionally, a deferred reference can be nested into another deferred reference.

[00040] Additionally, as shown in the following, the OOP environment can support method chaining.

```
Deferred<Boolean> defHasElements = eventually(invoking(list).getEnumeration().hasElements());
```

[00041] Also, the object-oriented programming language environment can support deferred

method invocation on a deferred type.

```
Deferred<Integer> defSize = eventually(invoking(defClusterMBean).getSize());
```

[00042] The deferred reference can be beneficial in a distributed data environment. For example, instead of using multiple try blocks, a simple test can ensure that a cluster reaches a certain size by simply calling the following to ensure that the virtual machine and Mbean server in the distributed data environment are available.

```
assertthat(eventually(invoking(defClusterMbean).getSize()), is (4))
```

[00043] Furthermore, the OOP environment can provide various other deferred helpers. For example, there can be a deferred helper that can cache a result of a successful get() request.

```
public <T> Deferred<T> cached(Deferred<T> deferred);
```

[00044] Also, a deferred helper can convert a value T into a Deferred<T>.

```
public <T> Deferred<T> deferred(T t);
```

[00045] Additionally, the OOP environment can obtain Java Future representation of a deferred type.

```
public <T> Future<T> future(Deferred<T> deferred);
```

Examples

[00046] **Figure 3** shows an illustration of an exemplary software pseudo code for acquiring management bean information from a server, in accordance with an embodiment of the invention. As shown in Figure 3, a caller can establish a connection to a server, e.g. a JMX MBeanServer instance (Line 5) before retrieving related management bean information, e.g. an MBeanInfo instance, based on the name of an object, objectName (Line 6).

[00047] The OOP environment may need to ensure that various conditions are satisfied before acquiring the management bean information from the server. For example, a virtual machine, e.g. a JAVA virtual machine (JVM), on which the JMX MBeanServer instance is running, may need to be started before the JMX MBeanServer can be created. Furthermore, a server connection, e.g. the MbeanServerConnection may need to be available with the MBean registered. In addition, the object-

oriented programming language environment may also need to handle various other exceptions, such as the IOException, InstanceNotFoundException, ClassCastException.

[00048] In accordance with various embodiments of the invention, the object-oriented programming language environment can use try blocks, as shown in the following, to ensure that these conditions can be satisfied.

```
while(true) {
... try to get reference ...
if (failed) Thread.sleep(10000);
if (succeeded) break;
}
```

[00049] The above approach of using try blocks may cause different problems, e.g., how long should the thread sleep, and how to allow the try block to fail or succeed earlier. Additionally, the code can become overly complicated and difficult to understand when there are too many retry loops.

[00050] Using the ensuring deferred types as defined in **Figure 2**, a DeferredJMXConnector instance can be created in the following.

```
Deferred<JMXConnector> defJMXConnector = new DeferredJMXConnector(jmxConnectionURL, env)
```

[00051] Then, a deferred MBeanInfo can be created.

```
Deferred<MBeanInfo> defMBeanInfo = new DeferredMBeanInfo(defJMXConnector, objectName)
```

[00052] Thus, the code for acquiring management bean information from a server can be simply implemented as:

```
MBeanInfo info = ensure(defMbeanInfo);
```

or

```
MBeanInfo info = ensure(defMbeanInfo, 2, TimeUnit.MINUTES).
```

[00053] As shown in the above, using the ensuring deferred types, the ensuring deferred types can isolate “waiting” logic into the ensuring method that takes a deferred type as a parameter. Furthermore, the ensuring deferred types allow nested deferred types without nested timeouts.

[00054] In the examples as shown in the following, the OOP environment can assert that a

service is running.

```
assertThat(eventually(invoking(CacheFactory.getService("some-service").isRunning()), is(true));
```

[00055] Also, the OOP environment can assert that two named cache, cacheA and cacheB are equal.

```
NamedCache cacheA = ...;
NamedCache cacheB = ...;
assertThat(eventually(cacheA), is(equalTo(cacheB)));
```

[00056] **Figure 4** illustrates an exemplary flow chart for supporting a deferred reference in an OOP environment in accordance with an embodiment of the invention. As shown in Figure 4, at step 401, the OOP environment can provide a deferred reference to an object that will be available at a future time. Furthermore, at step 402, after receiving a request from a caller to get the object based on the deferred reference, the OOP environment can return the object to the caller when the object is available or can indicate to the caller when the object is not currently available and/or when the object will never become available.

[00057] **Figure 5** illustrates an exemplary system for supporting a deferred reference in an OOP environment in accordance with an embodiment of the invention. As shown in Figure 5, the system 500 is shown as comprising an interface 510 and a caller 502. The caller 502 caller can send a request to the interface 510 to get an object based on the deferred reference. The interface 510 can provide a deferred reference to the object that will be available at a future time, and, after receiving the request from the caller, can perform one of: returning the object to the caller when the object is available, and indicating to the caller at least one of when the object is not currently available and when the object will never become available.

[00058] The interface 510 is capable of avoiding blocking a calling thread that the caller 502 uses to request the object based on the deferred reference while waiting for the object to become available. The interface 510 is also capable of informing the caller 502 about which class the object is associated with before the object becomes available. The interface 510 is further capable of throwing a runtime exception or returning a NULL value, when the object is not currently available. The interface 510 is further capable of throwing an object not available exception, if the object will never become available. The interface 510 can allow for nesting a deferred reference into another deferred reference.

[00059] The system 510 can further comprises a helper 520. The helper 520 is capable of ensuring that the caller 502 waits a maximum period of time for the object to become available. The

helper 520 is also capable of creating a deferred reference to an object based on a class definition associated with the object. The helper 520 is further capable of obtaining a recording dynamic proxy of the object, and providing a deferred representation of a last recorded call by the recording dynamic proxy of the object.

[00060] According to one embodiment, there disclosed a system for supporting a deferred reference in an object-oriented programming language environment operating on one or more microprocessors. The system comprises means for providing a deferred reference to an object that will be available at a future time, and means for performing, after receiving a request from a caller to get the object based on the deferred reference, one of: returning the object to the caller when the object is available, and indicating to the caller at least one of when the object is not currently available and when the object will never become available.

[00061] Preferably, the system further comprises means for avoiding blocking a calling thread that the caller uses to request the object based on the deferred reference while waiting for the object to become available.

[00062] Preferably, the system further comprises means for informing the caller about which class the object is associated with before the object becomes available.

[00063] Preferably, the system further comprises means for throwing a runtime exception or returning a NULL value, when the object is not currently available.

[00064] Preferably, the system further comprises means for throwing an object not available exception, if the object will never become available.

[00065] Preferably, the system further comprises means for ensuring that the caller waits a maximum period of time for the object to become available.

[00066] Preferably, the system further comprises means for creating a deferred reference to an object based on a class definition associated with the object.

[00067] Preferably, the system further comprises means for nesting a deferred reference into another deferred reference.

[00068] Preferably, the system further comprises means for obtaining a recording dynamic proxy of the object, and means for providing a deferred representation of a last recorded call by the recording dynamic proxy of the object.

[00069] Preferably, the system further comprises meas for providing a mechanism for an application to be notified when the deferred object becomes available.

[00070] According to one embodiment, there disclosed a system for supporting a deferred reference in an object-oriented programming language environment. The system comprises an interface configured to provide a deferred reference to an object that will be available at a future time and a caller configured to send a request to the interface to get the object based on the deferred reference. The interface, after receiving the request from the caller, is further configured to perform one of: returning the object to the caller when the object is available, and indicating to the

caller at least one of when the object is not currently available and when the object will never become available.

[00071] Preferably, the interface is capable of avoiding blocking a calling thread that the caller uses to request the object based on the deferred reference while waiting for the object to become available.

[00072] Preferably, the interface is capable of informing the caller about which class the object is associated with before the object becomes available.

[00073] Preferably, the interface is capable of throwing a runtime exception or returning a NULL value, when the object is not currently available.

[00074] Preferably, the interface is capable of throwing an object not available exception, if the object will never become available.

[00075] Preferably, the system further comprises a helper that is capable of ensuring that the caller waits a maximum period of time for the object to become available.

[00076] Preferably, the system further comprises a helper that is capable of creating a deferred reference to an object based on a class definition associated with the object.

[00077] Preferably, the interface allows for nesting a deferred reference into another deferred reference.

[00078] Preferably, the system further comprises a helper that is capable of obtaining a recording dynamic proxy of the object, and providing a deferred representation of a last recorded call by the recording dynamic proxy of the object.

[00079] **Figure 6** illustrates a functional block diagram to show the present feature. The present feature may be implemented as a system 600 for supporting a deferred reference in an object-oriented programming language environment. The system 600 includes one or more microprocessors 610 and an interface 110 running on the one or more microprocessors 610. The one or more microprocessors 610 includes: a providing unit 620 configured to provide a deferred reference to an object that will be available at a future time; and a performing unit 630 configured to perform, after receiving a request from a caller to get the object based on the deferred reference, one of: returning the object to the caller when the object is available, and indicating to the caller at least one of when the object is not currently available and when the object will never become available. Performing unit 630 includes returning unit 631 and indicating unit 632. Returning unit 631 is configured to return the object to the caller when the object is available. Indicating unit 632 is configured to indicate to the caller at least one of when the object is not currently available and when the object will never become available.

[00080] The present invention may be conveniently implemented using one or more conventional general purpose or specialized digital computer, computing device, machine, or microprocessor, including one or more processors, memory and/or computer readable storage media programmed according to the teachings of the present disclosure. Appropriate software coding can readily be

prepared by skilled programmers based on the teachings of the present disclosure, as will be apparent to those skilled in the software art.

[00081] In some embodiments, the present invention includes a computer program product which is a storage medium or computer readable medium (media) having instructions stored thereon/in which can be used to program a computer to perform any of the processes of the present invention. The storage medium can include, but is not limited to, any type of disk including floppy disks, optical discs, DVD, CD-ROMs, microdrive, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, DRAMs, VRAMs, flash memory devices, magnetic or optical cards, nanosystems (including molecular memory ICs), or any type of media or device suitable for storing instructions and/or data.

[00082] The foregoing description of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations will be apparent to the practitioner skilled in the art. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various modifications that are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalence.

Claims:

What is claimed is:

1. A method for supporting a deferred reference in an object-oriented programming language environment operating on one or more microprocessors, comprising:

providing a deferred reference to an object that will be available at a future time; and

after receiving a request from a caller to get the object based on the deferred reference, performing one of:

returning the object to the caller when the object is available, and

indicating to the caller at least one of when the object is not currently available and when the object will never become available.

2. The method according to Claim 1, further comprising:

avoiding blocking a calling thread that the caller uses to request the object based on the deferred reference while waiting for the object to become available.

3. The method according to Claim 1 or 2, further comprising:

informing the caller about which class the object is associated with before the object becomes available.

4. The method according to any preceding Claim, further comprising:

throwing a runtime exception or returning a NULL value, when the object is not currently available.

5. The method according to any preceding Claim, further comprising:

throwing an object not available exception, if the object will never become available.

6. The method according to any preceding Claim, further comprising:

ensuring that the caller waits a maximum period of time for the object to become available.

7. The method according to any preceding Claim, further comprising:

creating a deferred reference to an object based on a class definition associated with the object.

8. The method according to any preceding Claim, further comprising:

nesting a deferred reference into another deferred reference.

9. The method according to any preceding Claim, further comprising:

obtaining a recording dynamic proxy of the object, and

providing a deferred representation of a last recorded call by the recording dynamic proxy of the object.

10. The method according to any preceding Claim, further comprising:

providing a mechanism for an application to be notified when the deferred object becomes available.

11. A computer program comprising program instructions for running on one or more microprocessors to perform all the steps of the method of any preceeding claim.

12. A computer program product comprising the computer program of claim 11 provided on a machine-readable medium.

13. A system for supporting a deferred reference in an object-oriented programming language environment, comprising:

one or more microprocessors;

an interface running on the one or more microprocessors, wherein the interface operates to perform steps of:

providing a deferred reference to an object that will be available at a future time; and

after receiving a request from a caller to get the object based on the deferred reference, performing one of:

returning the object to the caller when the object is available, and

indicating to the caller at least one of when the object is not currently available and when the object will never become available.

14. The system according to Claim 13, wherein:

the interface is capable of avoiding blocking a calling thread that the caller uses to request the object based on the deferred reference while waiting for the object to become available.

15. The system according to Claim 13 or 14, wherein:

the interface is capable of informing the caller about which class the object is associated with before the object becomes available.

16. The system according to any of Claims 13 to 15, wherein:

the interface is capable of throwing a runtime exception or returning a NULL value, when the

object is not currently available.

17. The system according to any of Claims 13 to 16, wherein:

the interface is capable of throwing an object not available exception, if the object will never become available.

18. The system according to any of Claims 13 to 17, further comprising:

a helper that is capable of ensuring that the caller waits a maximum period of time for the object to become available.

19. The system according to any of Claims 13 to 18, further comprising:

a helper that is capable of creating a deferred reference to an object based on a class definition associated with the object.

20. The system according to any of Claims 13 to 19, wherein:

the interface allows for nesting a deferred reference into another deferred reference.

21. The system according to any of Claims 13 to 20, further comprising:

a helper that is capable of

obtaining a recording dynamic proxy of the object, and

providing a deferred representation of a last recorded call by the recording dynamic proxy of the object.

22. A non-transitory machine readable storage medium having instructions stored thereon that when executed cause a system to perform the steps comprising:

providing a deferred reference to an object that will be available at a future time; and

after receiving a request from a caller to get the object based on the deferred reference, performing one of:

returning the object to the caller when the object is available, and

indicating to the caller at least one of when the object is not currently available and when the object will never become available.

23. A computer program for causing a computer to implement the method recited in any one of claims 1 to 10.

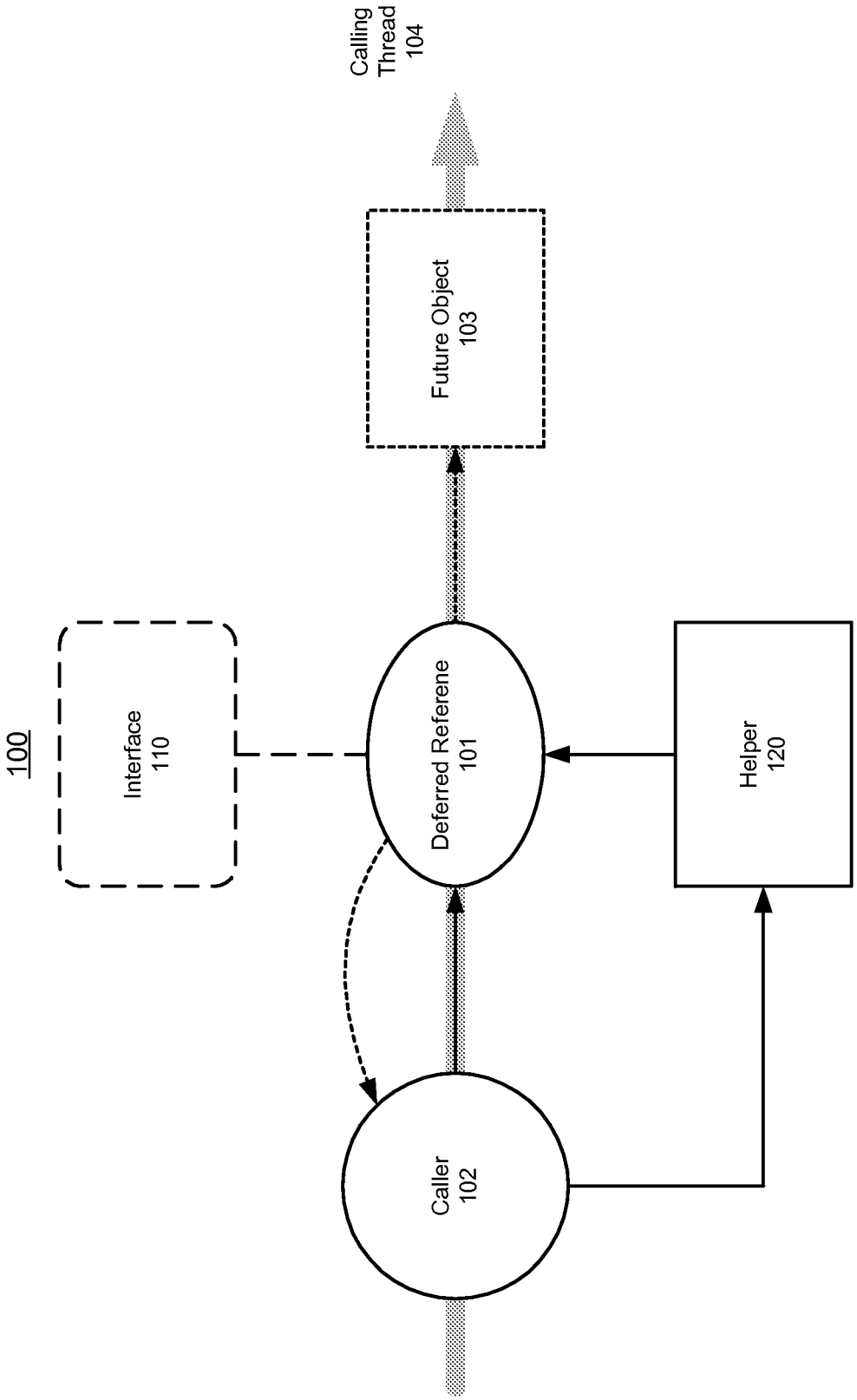
Calling Thread 104
Future Object 103
100
Interface 110
Deferred Referene 101
Helper 120
Caller 102

FIGURE 1

```
interface Deferred<T>
  {
  T get() throws ObjectNotAvailableException;   //a runtime exception

  Class<T> getDeferredClass();   //so we can determine the class of the referent without it actually existing
  }
```

FIGURE 2

```
String sJMXConnectionURL = … ;
Map<String, Object> mapEnv = … ;
ObjectName objectName = … ;
JMXConnector connector = JMXConnectorFactory.newJMXConnector(sJMXConnectionURL, mapEnv);
MBeanServerConnection connection = connector.getMBeanServerConnection();
MBeanInfo info = connection.getMBeanInfo(objectName);
```

FIGURE 3

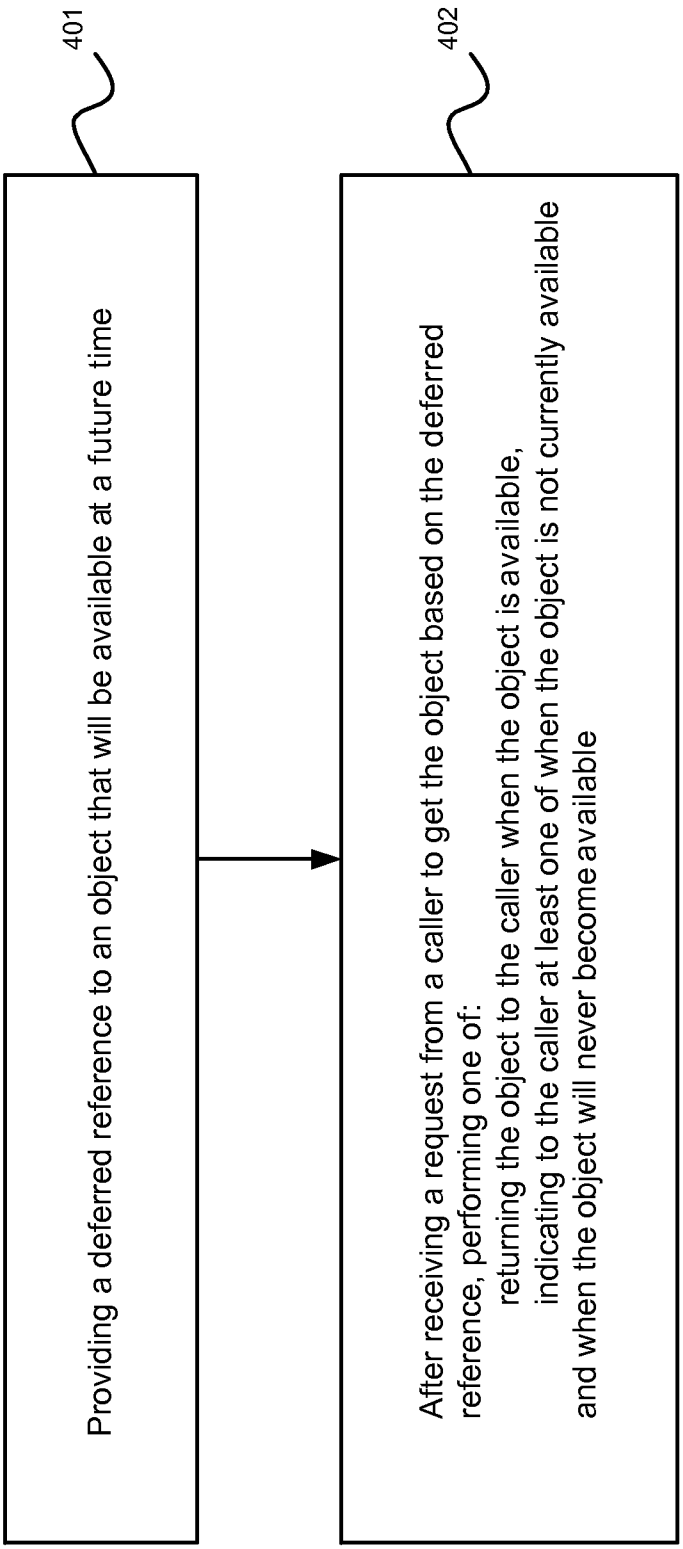
401
Providing a deferred reference to an object that will be available at a future time
402
After receiving a request from a caller to get the object based on the deferred reference, performing one of:
returning the object to the caller when the object is available,
indicating to the caller at least one of when the object is not currently available
and when the object will never become available

FIGURE 4

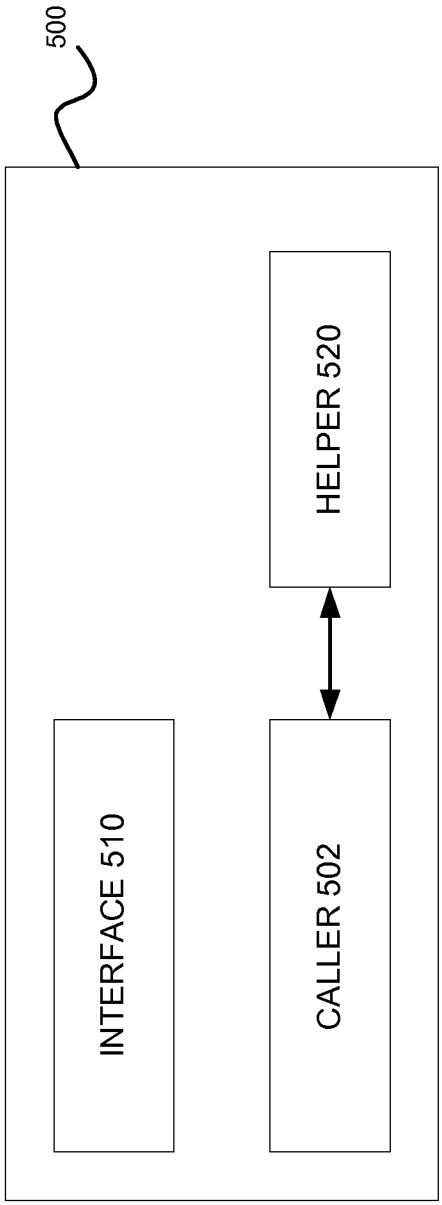

FIGURE 5

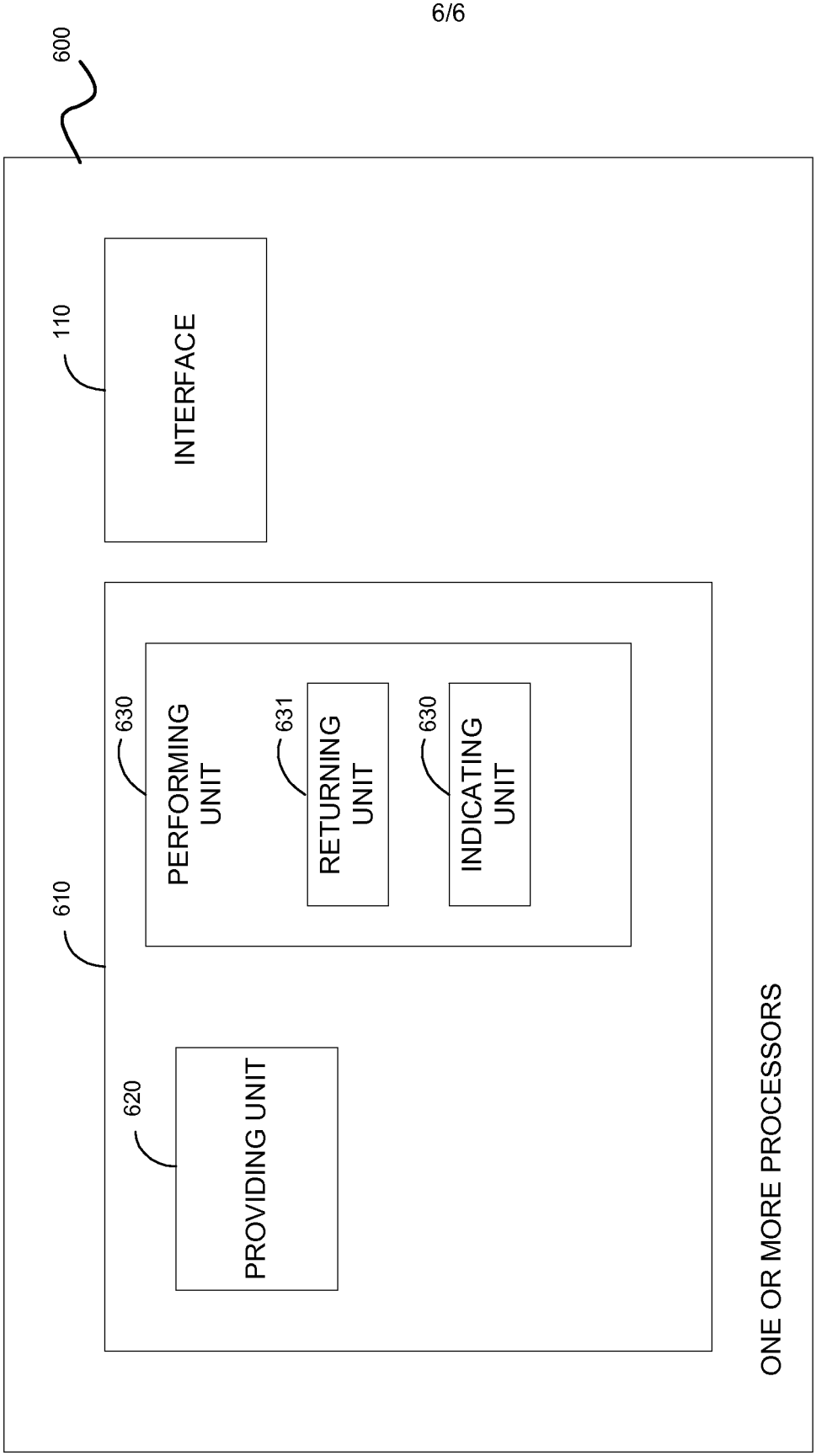
600
110
INTERFACE
610
630
PERFORMING UNIT
631
RETURNING UNIT
630
INDICATING UNIT
620
PROVIDING UNIT
ONE OR MORE PROCESSORS

FIGURE 6

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2013/039256

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/44
ADD.

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)
G06F

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, INSPEC, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	unknown: "Deferred Reference", Twisted Matrix Labs documentation , 6 December 2010 (2010-12-06), XP002711637, Retrieved from the Internet: URL:http://web.archive.org/web/20101206085523/http://twistedmatrix.com/documents/10.1.0/core/howto/defer.html [retrieved on 2013-08-02] the whole document Deferreds, Callbacks; page 1 Multiple callbacks; page 2 Visual Explanation, Errbacks; page 3 Errbacks (continued), Unhandled Errors; page 4 Handling either synchronous or -/--	1-23

[X] Further documents are listed in the continuation of Box C.

[X] 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
26 August 2013	11/09/2013

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Eftimescu, Nicolae

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	asynchronous results; page 5 DeferredList; page 6 - page 7 Class Overview, Basic Callback Functions, Chaining Deferreds; page 7 - page 8 -& Unknown: "Generating Deferreds", Twisted Matrix Labs documentation , 6 December 2010 (2010-12-06), XP002711798, Retrieved from the Internet: URL:http://web.archive.org/web/20101206085 527/http://twistedmatrix.com/documents/10. 1.0/core/howto/gendefer.html [retrieved on 2001-08-02] the whole document Class overview; What Deferreds don't do: make your code asynchronous; page 1 - page 2 Advanced Processing Chain Control; page 2 Returning Deferreds from synchronous functions; Integrating blocking code with Twisted; page 3 - page 4 Possible sources of error; Synchronous callback execution; page 4	

A	The jQuery Project: "Deferred Object", jQuery API , 24 November 2011 (2011-11-24), XP002711799, Retrieved from the Internet: URL:http://web.archive.org/web/20111206101 237/http://api.jquery.com/category/deferre d-object/ [retrieved on 2013-08-07] the whole document	1-23

A	EP 1 235 149 A2 (SUN MICROSYSTEMS INC [US]) 28 August 2002 (2002-08-28) abstract paragraph [0001] - paragraph [0024] paragraph [0026] - paragraph [0030] paragraph [0037] - paragraph [0075]	1-23

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2013/039256

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 1235149 A2	28-08-2002	NONE	