**java.lang  
Class Object**

**java.lang.Object**

public class **Object**

Class Object is the root of the class hierarchy. Every class has Object as a superclass. All objects, including arrays, implement the methods of this class.

**Since:**

JDK1.0, CLDC 1.0

**See Also:**

[Class](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Class.html)

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**Object**](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#Object())() |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| boolean | [**equals**](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#equals(java.lang.Object))([Object](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html) obj)           Indicates whether some other object is "equal to" this one. |
| [Class](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Class.html) | [**getClass**](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#getClass())()           Returns the runtime class of an object. |
| int | [**hashCode**](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#hashCode())()           Returns a hash code value for the object. |
| void | [**notify**](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#notify())()           Wakes up a single thread that is waiting on this object's monitor. |
| void | [**notifyAll**](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#notifyAll())()           Wakes up all threads that are waiting on this object's monitor. |
| [String](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/String.html) | [**toString**](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#toString())()           Returns a string representation of the object. |
| void | [**wait**](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#wait())()           Causes current thread to wait until another thread invokes the [notify()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#notify()) method or the [notifyAll()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html" \l "notifyAll()) method for this object. |
| void | [**wait**](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#wait(long))(long timeout)           Causes current thread to wait until either another thread invokes the [notify()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#notify()) method or the [notifyAll()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html" \l "notifyAll()) method for this object, or a specified amount of time has elapsed. |
| void | [**wait**](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#wait(long,%20int))(long timeout, int nanos)           Causes current thread to wait until another thread invokes the [notify()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#notify()) method or the [notifyAll()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html" \l "notifyAll()) method for this object, or some other thread interrupts the current thread, or a certain amount of real time has elapsed. |

|  |
| --- |
| **Constructor Detail** |

**Object**

public **Object**()

|  |
| --- |
| **Method Detail** |

**getClass**

public final [Class](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Class.html) **getClass**()

Returns the runtime class of an object. That Class object is the object that is locked by static synchronized methods of the represented class.

**Returns:**

the object of type Class that represents the runtime class of the object.

**hashCode**

public int **hashCode**()

Returns a hash code value for the object. This method is supported for the benefit of hashtables such as those provided by java.util.Hashtable.

The general contract of hashCode is:

* Whenever it is invoked on the same object more than once during an execution of a Java application, the hashCode method must consistently return the same integer, provided no information used in equals comparisons on the object is modified. This integer need not remain consistent from one execution of an application to another execution of the same application.
* If two objects are equal according to the equals(Object) method, then calling the hashCode method on each of the two objects must produce the same integer result.
* It is *not* required that if two objects are unequal according to the [equals(java.lang.Object)](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#equals(java.lang.Object)) method, then calling the hashCode method on each of the two objects must produce distinct integer results. However, the programmer should be aware that producing distinct integer results for unequal objects may improve the performance of hashtables.

As much as is reasonably practical, the hashCode method defined by class Object does return distinct integers for distinct objects. (This is typically implemented by converting the internal address of the object into an integer, but this implementation technique is not required by the JavaTM programming language.)

**Returns:**

a hash code value for this object.

**See Also:**

[equals(java.lang.Object)](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#equals(java.lang.Object)), [Hashtable](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/util/Hashtable.html" \o "class in java.util)

**equals**

public boolean **equals**([Object](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html) obj)

Indicates whether some other object is "equal to" this one.

The equals method implements an equivalence relation:

* It is *reflexive*: for any reference value x, x.equals(x) should return true.
* It is *symmetric*: for any reference values x and y, x.equals(y) should return true if and only if y.equals(x) returns true.
* It is *transitive*: for any reference values x, y, and z, if x.equals(y) returns true and y.equals(z) returns true, then x.equals(z) should return true.
* It is *consistent*: for any reference values x and y, multiple invocations of x.equals(y) consistently return true or consistently return false, provided no information used in equals comparisons on the object is modified.
* For any non-null reference value x, x.equals(null) should return false.

The equals method for class Object implements the most discriminating possible equivalence relation on objects; that is, for any reference values x and y, this method returns true if and only if x and y refer to the same object (x==y has the value true).

**Parameters:**

obj - the reference object with which to compare.

**Returns:**

true if this object is the same as the obj argument; false otherwise.

**See Also:**

[Boolean.hashCode()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Boolean.html#hashCode()), [Hashtable](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/util/Hashtable.html" \o "class in java.util)

**toString**

public [String](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/String.html) **toString**()

Returns a string representation of the object. In general, the toString method returns a string that "textually represents" this object. The result should be a concise but informative representation that is easy for a person to read. It is recommended that all subclasses override this method.

The toString method for class Object returns a string consisting of the name of the class of which the object is an instance, the at-sign character `@', and the unsigned hexadecimal representation of the hash code of the object. In other words, this method returns a string equal to the value of:

getClass().getName() + '@' + Integer.toHexString(hashCode())

**Returns:**

a string representation of the object.

**notify**

public final void **notify**()

Wakes up a single thread that is waiting on this object's monitor. If any threads are waiting on this object, one of them is chosen to be awakened. The choice is arbitrary and occurs at the discretion of the implementation. A thread waits on an object's monitor by calling one of the wait methods.

The awakened thread will not be able to proceed until the current thread relinquishes the lock on this object. The awakened thread will compete in the usual manner with any other threads that might be actively competing to synchronize on this object; for example, the awakened thread enjoys no reliable privilege or disadvantage in being the next thread to lock this object.

This method should only be called by a thread that is the owner of this object's monitor. A thread becomes the owner of the object's monitor in one of three ways:

* By executing a synchronized instance method of that object.
* By executing the body of a synchronized statement that synchronizes on the object.
* For objects of type Class, by executing a synchronized static method of that class.

Only one thread at a time can own an object's monitor.

**Throws:**

[IllegalMonitorStateException](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/IllegalMonitorStateException.html) - if the current thread is not the owner of this object's monitor.

**See Also:**

[notifyAll()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#notifyAll()), [wait()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#wait())

**notifyAll**

public final void **notifyAll**()

Wakes up all threads that are waiting on this object's monitor. A thread waits on an object's monitor by calling one of the wait methods.

The awakened threads will not be able to proceed until the current thread relinquishes the lock on this object. The awakened threads will compete in the usual manner with any other threads that might be actively competing to synchronize on this object; for example, the awakened threads enjoy no reliable privilege or disadvantage in being the next thread to lock this object.

This method should only be called by a thread that is the owner of this object's monitor. See the notify method for a description of the ways in which a thread can become the owner of a monitor.

**Throws:**

[IllegalMonitorStateException](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/IllegalMonitorStateException.html) - if the current thread is not the owner of this object's monitor.

**See Also:**

[notify()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#notify()), [wait()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#wait())

**wait**

public final void **wait**(long timeout)

throws [InterruptedException](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/InterruptedException.html)

Causes current thread to wait until either another thread invokes the [notify()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#notify()) method or the [notifyAll()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html" \l "notifyAll()) method for this object, or a specified amount of time has elapsed.

The current thread must own this object's monitor.

This method causes the current thread (call it *T*) to place itself in the wait set for this object and then to relinquish any and all synchronization claims on this object. Thread *T* becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

* Some other thread invokes the notify method for this object and thread *T* happens to be arbitrarily chosen as the thread to be awakened.
* Some other thread invokes the notifyAll method for this object.
* Some other thread [interrupts](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Thread.html#interrupt()) thread *T*.
* The specified amount of real time has elapsed, more or less. If timeout is zero, however, then real time is not taken into consideration and the thread simply waits until notified.

The thread *T* is then removed from the wait set for this object and re-enabled for thread scheduling. It then competes in the usual manner with other threads for the right to synchronize on the object; once it has gained control of the object, all its synchronization claims on the object are restored to the status quo ante - that is, to the situation as of the time that the wait method was invoked. Thread *T* then returns from the invocation of the wait method. Thus, on return from the wait method, the synchronization state of the object and of thread *T* is exactly as it was when the wait method was invoked.

If the current thread is [interrupted](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Thread.html#interrupt()) by another thread while it is waiting, then an InterruptedException is thrown. This exception is not thrown until the lock status of this object has been restored as described above.

Note that the wait method, as it places the current thread into the wait set for this object, unlocks only this object; any other objects on which the current thread may be synchronized remain locked while the thread waits.

This method should only be called by a thread that is the owner of this object's monitor. See the notify method for a description of the ways in which a thread can become the owner of a monitor.

**Parameters:**

timeout - the maximum time to wait in milliseconds.

**Throws:**

[IllegalArgumentException](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/IllegalArgumentException.html) - if the value of timeout is negative.

[IllegalMonitorStateException](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/IllegalMonitorStateException.html) - if the current thread is not the owner of the object's monitor.

[InterruptedException](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/InterruptedException.html) - if another thread has interrupted the current thread. The *interrupted status* of the current thread is cleared when this exception is thrown.

**See Also:**

[notify()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#notify()), [notifyAll()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html" \l "notifyAll())

**wait**

public final void **wait**(long timeout,

int nanos)

throws [InterruptedException](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/InterruptedException.html)

Causes current thread to wait until another thread invokes the [notify()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#notify()) method or the [notifyAll()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html" \l "notifyAll()) method for this object, or some other thread interrupts the current thread, or a certain amount of real time has elapsed.

This method is similar to the wait method of one argument, but it allows finer control over the amount of time to wait for a notification before giving up. The amount of real time, measured in nanoseconds, is given by:

1000000\*timeout+nanos

In all other respects, this method does the same thing as the method [wait(long)](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#wait(long)) of one argument. In particular, wait(0, 0) means the same thing as wait(0).

The current thread must own this object's monitor. The thread releases ownership of this monitor and waits until either of the following two conditions has occurred:

* Another thread notifies threads waiting on this object's monitor to wake up either through a call to the notify method or the notifyAll method.
* The timeout period, specified by timeout milliseconds plus nanos nanoseconds arguments, has elapsed.

The thread then waits until it can re-obtain ownership of the monitor and resumes execution

This method should only be called by a thread that is the owner of this object's monitor. See the notify method for a description of the ways in which a thread can become the owner of a monitor.

**Parameters:**

timeout - the maximum time to wait in milliseconds.

nanos - additional time, in nanoseconds range 0-999999.

**Throws:**

[IllegalArgumentException](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/IllegalArgumentException.html) - if the value of timeout is negative or the value of nanos is not in the range 0-999999.

[IllegalMonitorStateException](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/IllegalMonitorStateException.html) - if the current thread is not the owner of this object's monitor.

[InterruptedException](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/InterruptedException.html) - if another thread has interrupted the current thread. The *interrupted status* of the current thread is cleared when this exception is thrown.

**wait**

public final void **wait**()

throws [InterruptedException](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/InterruptedException.html)

Causes current thread to wait until another thread invokes the [notify()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#notify()) method or the [notifyAll()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html" \l "notifyAll()) method for this object. In other word's this method behaves exactly as if it simply performs the call wait(0).

The current thread must own this object's monitor. The thread releases ownership of this monitor and waits until another thread notifies threads waiting on this object's monitor to wake up either through a call to the notify method or the notifyAll method. The thread then waits until it can re-obtain ownership of the monitor and resumes execution.

This method should only be called by a thread that is the owner of this object's monitor. See the notify method for a description of the ways in which a thread can become the owner of a monitor.

**Throws:**

[IllegalMonitorStateException](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/IllegalMonitorStateException.html) - if the current thread is not the owner of the object's monitor.

[InterruptedException](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/InterruptedException.html) - if another thread has interrupted the current thread. The *interrupted status* of the current thread is cleared when this exception is thrown.

**See Also:**

[notify()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html#notify()), [notifyAll()](https://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/java/lang/Object.html" \l "notifyAll())