| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/KeyAgreement.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/crypto/IllegalBlockSizeException.html)   [**NEXT CLASS**](http://docs.google.com/javax/crypto/KeyAgreementSpi.html) | [**FRAMES**](http://docs.google.com/index.html?javax/crypto/KeyAgreement.html)    [**NO FRAMES**](http://docs.google.com/KeyAgreement.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#4d34og8) |

## **javax.crypto**

Class KeyAgreement

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **javax.crypto.KeyAgreement**

public class **KeyAgreement**extends [Object](http://docs.google.com/java/lang/Object.html)

This class provides the functionality of a key agreement (or key exchange) protocol.

The keys involved in establishing a shared secret are created by one of the key generators (KeyPairGenerator or KeyGenerator), a KeyFactory, or as a result from an intermediate phase of the key agreement protocol.

For each of the correspondents in the key exchange, doPhase needs to be called. For example, if this key exchange is with one other party, doPhase needs to be called once, with the lastPhase flag set to true. If this key exchange is with two other parties, doPhase needs to be called twice, the first time setting the lastPhase flag to false, and the second time setting it to true. There may be any number of parties involved in a key exchange.

**Since:** 1.4 **See Also:**[KeyGenerator](http://docs.google.com/javax/crypto/KeyGenerator.html), [SecretKey](http://docs.google.com/javax/crypto/SecretKey.html)

| **Constructor Summary** | |
| --- | --- |
| protected | [**KeyAgreement**](http://docs.google.com/javax/crypto/KeyAgreement.html#KeyAgreement(javax.crypto.KeyAgreementSpi,%20java.security.Provider,%20java.lang.String))([KeyAgreementSpi](http://docs.google.com/javax/crypto/KeyAgreementSpi.html) keyAgreeSpi, [Provider](http://docs.google.com/java/security/Provider.html) provider, [String](http://docs.google.com/java/lang/String.html) algorithm)            Creates a KeyAgreement object. |

| **Method Summary** | |
| --- | --- |
| [Key](http://docs.google.com/java/security/Key.html) | [**doPhase**](http://docs.google.com/javax/crypto/KeyAgreement.html#doPhase(java.security.Key,%20boolean))([Key](http://docs.google.com/java/security/Key.html) key, boolean lastPhase)            Executes the next phase of this key agreement with the given key that was received from one of the other parties involved in this key agreement. |
| byte[] | [**generateSecret**](http://docs.google.com/javax/crypto/KeyAgreement.html#generateSecret())()            Generates the shared secret and returns it in a new buffer. |
| int | [**generateSecret**](http://docs.google.com/javax/crypto/KeyAgreement.html#generateSecret(byte%5B%5D,%20int))(byte[] sharedSecret, int offset)            Generates the shared secret, and places it into the buffer sharedSecret, beginning at offset inclusive. |
| [SecretKey](http://docs.google.com/javax/crypto/SecretKey.html) | [**generateSecret**](http://docs.google.com/javax/crypto/KeyAgreement.html#generateSecret(java.lang.String))([String](http://docs.google.com/java/lang/String.html) algorithm)            Creates the shared secret and returns it as a SecretKey object of the specified algorithm. |
| [String](http://docs.google.com/java/lang/String.html) | [**getAlgorithm**](http://docs.google.com/javax/crypto/KeyAgreement.html#getAlgorithm())()            Returns the algorithm name of this KeyAgreement object. |
| static [KeyAgreement](http://docs.google.com/javax/crypto/KeyAgreement.html) | [**getInstance**](http://docs.google.com/javax/crypto/KeyAgreement.html#getInstance(java.lang.String))([String](http://docs.google.com/java/lang/String.html) algorithm)            Returns a KeyAgreement object that implements the specified key agreement algorithm. |
| static [KeyAgreement](http://docs.google.com/javax/crypto/KeyAgreement.html) | [**getInstance**](http://docs.google.com/javax/crypto/KeyAgreement.html#getInstance(java.lang.String,%20java.security.Provider))([String](http://docs.google.com/java/lang/String.html) algorithm, [Provider](http://docs.google.com/java/security/Provider.html) provider)            Returns a KeyAgreement object that implements the specified key agreement algorithm. |
| static [KeyAgreement](http://docs.google.com/javax/crypto/KeyAgreement.html) | [**getInstance**](http://docs.google.com/javax/crypto/KeyAgreement.html#getInstance(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) algorithm, [String](http://docs.google.com/java/lang/String.html) provider)            Returns a KeyAgreement object that implements the specified key agreement algorithm. |
| [Provider](http://docs.google.com/java/security/Provider.html) | [**getProvider**](http://docs.google.com/javax/crypto/KeyAgreement.html#getProvider())()            Returns the provider of this KeyAgreement object. |
| void | [**init**](http://docs.google.com/javax/crypto/KeyAgreement.html#init(java.security.Key))([Key](http://docs.google.com/java/security/Key.html) key)            Initializes this key agreement with the given key, which is required to contain all the algorithm parameters required for this key agreement. |
| void | [**init**](http://docs.google.com/javax/crypto/KeyAgreement.html#init(java.security.Key,%20java.security.spec.AlgorithmParameterSpec))([Key](http://docs.google.com/java/security/Key.html) key, [AlgorithmParameterSpec](http://docs.google.com/java/security/spec/AlgorithmParameterSpec.html) params)            Initializes this key agreement with the given key and set of algorithm parameters. |
| void | [**init**](http://docs.google.com/javax/crypto/KeyAgreement.html#init(java.security.Key,%20java.security.spec.AlgorithmParameterSpec,%20java.security.SecureRandom))([Key](http://docs.google.com/java/security/Key.html) key, [AlgorithmParameterSpec](http://docs.google.com/java/security/spec/AlgorithmParameterSpec.html) params, [SecureRandom](http://docs.google.com/java/security/SecureRandom.html) random)            Initializes this key agreement with the given key, set of algorithm parameters, and source of randomness. |
| void | [**init**](http://docs.google.com/javax/crypto/KeyAgreement.html#init(java.security.Key,%20java.security.SecureRandom))([Key](http://docs.google.com/java/security/Key.html) key, [SecureRandom](http://docs.google.com/java/security/SecureRandom.html) random)            Initializes this key agreement with the given key and source of randomness. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

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

### KeyAgreement

protected **KeyAgreement**([KeyAgreementSpi](http://docs.google.com/javax/crypto/KeyAgreementSpi.html) keyAgreeSpi,  
 [Provider](http://docs.google.com/java/security/Provider.html) provider,  
 [String](http://docs.google.com/java/lang/String.html) algorithm)

Creates a KeyAgreement object.

**Parameters:**keyAgreeSpi - the delegateprovider - the provideralgorithm - the algorithm

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

### getAlgorithm

public final [String](http://docs.google.com/java/lang/String.html) **getAlgorithm**()

Returns the algorithm name of this KeyAgreement object.

This is the same name that was specified in one of the getInstance calls that created this KeyAgreement object.

**Returns:**the algorithm name of this KeyAgreement object.

### getInstance

public static final [KeyAgreement](http://docs.google.com/javax/crypto/KeyAgreement.html) **getInstance**([String](http://docs.google.com/java/lang/String.html) algorithm)  
 throws [NoSuchAlgorithmException](http://docs.google.com/java/security/NoSuchAlgorithmException.html)

Returns a KeyAgreement object that implements the specified key agreement algorithm.

This method traverses the list of registered security Providers, starting with the most preferred Provider. A new KeyAgreement object encapsulating the KeyAgreementSpi implementation from the first Provider that supports the specified algorithm is returned.

Note that the list of registered providers may be retrieved via the [Security.getProviders()](http://docs.google.com/java/security/Security.html#getProviders()) method.

**Parameters:**algorithm - the standard name of the requested key agreement algorithm. See Appendix A in the  [Java Cryptography Architecture Reference Guide](http://docs.google.com/technotes/guides/security/crypto/CryptoSpec.html#AppA) for information about standard algorithm names. **Returns:**the new KeyAgreement object. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified algorithm is null. [NoSuchAlgorithmException](http://docs.google.com/java/security/NoSuchAlgorithmException.html) - if no Provider supports a KeyAgreementSpi implementation for the specified algorithm.**See Also:**[Provider](http://docs.google.com/java/security/Provider.html)

### getInstance

public static final [KeyAgreement](http://docs.google.com/javax/crypto/KeyAgreement.html) **getInstance**([String](http://docs.google.com/java/lang/String.html) algorithm,  
 [String](http://docs.google.com/java/lang/String.html) provider)  
 throws [NoSuchAlgorithmException](http://docs.google.com/java/security/NoSuchAlgorithmException.html),  
 [NoSuchProviderException](http://docs.google.com/java/security/NoSuchProviderException.html)

Returns a KeyAgreement object that implements the specified key agreement algorithm.

A new KeyAgreement object encapsulating the KeyAgreementSpi implementation from the specified provider is returned. The specified provider must be registered in the security provider list.

Note that the list of registered providers may be retrieved via the [Security.getProviders()](http://docs.google.com/java/security/Security.html#getProviders()) method.

**Parameters:**algorithm - the standard name of the requested key agreement algorithm. See Appendix A in the  [Java Cryptography Architecture Reference Guide](http://docs.google.com/technotes/guides/security/crypto/CryptoSpec.html#AppA) for information about standard algorithm names.provider - the name of the provider. **Returns:**the new KeyAgreement object. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified algorithm is null. [NoSuchAlgorithmException](http://docs.google.com/java/security/NoSuchAlgorithmException.html) - if a KeyAgreementSpi implementation for the specified algorithm is not available from the specified provider. [NoSuchProviderException](http://docs.google.com/java/security/NoSuchProviderException.html) - if the specified provider is not registered in the security provider list. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the provider is null or empty.**See Also:**[Provider](http://docs.google.com/java/security/Provider.html)

### getInstance

public static final [KeyAgreement](http://docs.google.com/javax/crypto/KeyAgreement.html) **getInstance**([String](http://docs.google.com/java/lang/String.html) algorithm,  
 [Provider](http://docs.google.com/java/security/Provider.html) provider)  
 throws [NoSuchAlgorithmException](http://docs.google.com/java/security/NoSuchAlgorithmException.html)

Returns a KeyAgreement object that implements the specified key agreement algorithm.

A new KeyAgreement object encapsulating the KeyAgreementSpi implementation from the specified Provider object is returned. Note that the specified Provider object does not have to be registered in the provider list.

**Parameters:**algorithm - the standard name of the requested key agreement algorithm. See Appendix A in the  [Java Cryptography Architecture Reference Guide](http://docs.google.com/technotes/guides/security/crypto/CryptoSpec.html#AppA) for information about standard algorithm names.provider - the provider. **Returns:**the new KeyAgreement object. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified algorithm is null. [NoSuchAlgorithmException](http://docs.google.com/java/security/NoSuchAlgorithmException.html) - if a KeyAgreementSpi implementation for the specified algorithm is not available from the specified Provider object. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the provider is null.**See Also:**[Provider](http://docs.google.com/java/security/Provider.html)

### getProvider

public final [Provider](http://docs.google.com/java/security/Provider.html) **getProvider**()

Returns the provider of this KeyAgreement object.

**Returns:**the provider of this KeyAgreement object

### init

public final void **init**([Key](http://docs.google.com/java/security/Key.html) key)  
 throws [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html)

Initializes this key agreement with the given key, which is required to contain all the algorithm parameters required for this key agreement.

If this key agreement requires any random bytes, it will get them using the [SecureRandom](http://docs.google.com/java/security/SecureRandom.html) implementation of the highest-priority installed provider as the source of randomness. (If none of the installed providers supply an implementation of SecureRandom, a system-provided source of randomness will be used.)

**Parameters:**key - the party's private information. For example, in the case of the Diffie-Hellman key agreement, this would be the party's own Diffie-Hellman private key. **Throws:** [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html) - if the given key is inappropriate for this key agreement, e.g., is of the wrong type or has an incompatible algorithm type.

### init

public final void **init**([Key](http://docs.google.com/java/security/Key.html) key,  
 [SecureRandom](http://docs.google.com/java/security/SecureRandom.html) random)  
 throws [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html)

Initializes this key agreement with the given key and source of randomness. The given key is required to contain all the algorithm parameters required for this key agreement.

If the key agreement algorithm requires random bytes, it gets them from the given source of randomness, random. However, if the underlying algorithm implementation does not require any random bytes, random is ignored.

**Parameters:**key - the party's private information. For example, in the case of the Diffie-Hellman key agreement, this would be the party's own Diffie-Hellman private key.random - the source of randomness **Throws:** [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html) - if the given key is inappropriate for this key agreement, e.g., is of the wrong type or has an incompatible algorithm type.

### init

public final void **init**([Key](http://docs.google.com/java/security/Key.html) key,  
 [AlgorithmParameterSpec](http://docs.google.com/java/security/spec/AlgorithmParameterSpec.html) params)  
 throws [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html),  
 [InvalidAlgorithmParameterException](http://docs.google.com/java/security/InvalidAlgorithmParameterException.html)

Initializes this key agreement with the given key and set of algorithm parameters.

If this key agreement requires any random bytes, it will get them using the [SecureRandom](http://docs.google.com/java/security/SecureRandom.html) implementation of the highest-priority installed provider as the source of randomness. (If none of the installed providers supply an implementation of SecureRandom, a system-provided source of randomness will be used.)

**Parameters:**key - the party's private information. For example, in the case of the Diffie-Hellman key agreement, this would be the party's own Diffie-Hellman private key.params - the key agreement parameters **Throws:** [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html) - if the given key is inappropriate for this key agreement, e.g., is of the wrong type or has an incompatible algorithm type. [InvalidAlgorithmParameterException](http://docs.google.com/java/security/InvalidAlgorithmParameterException.html) - if the given parameters are inappropriate for this key agreement.

### init

public final void **init**([Key](http://docs.google.com/java/security/Key.html) key,  
 [AlgorithmParameterSpec](http://docs.google.com/java/security/spec/AlgorithmParameterSpec.html) params,  
 [SecureRandom](http://docs.google.com/java/security/SecureRandom.html) random)  
 throws [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html),  
 [InvalidAlgorithmParameterException](http://docs.google.com/java/security/InvalidAlgorithmParameterException.html)

Initializes this key agreement with the given key, set of algorithm parameters, and source of randomness.

**Parameters:**key - the party's private information. For example, in the case of the Diffie-Hellman key agreement, this would be the party's own Diffie-Hellman private key.params - the key agreement parametersrandom - the source of randomness **Throws:** [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html) - if the given key is inappropriate for this key agreement, e.g., is of the wrong type or has an incompatible algorithm type. [InvalidAlgorithmParameterException](http://docs.google.com/java/security/InvalidAlgorithmParameterException.html) - if the given parameters are inappropriate for this key agreement.

### doPhase

public final [Key](http://docs.google.com/java/security/Key.html) **doPhase**([Key](http://docs.google.com/java/security/Key.html) key,  
 boolean lastPhase)  
 throws [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html),  
 [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html)

Executes the next phase of this key agreement with the given key that was received from one of the other parties involved in this key agreement.

**Parameters:**key - the key for this phase. For example, in the case of Diffie-Hellman between 2 parties, this would be the other party's Diffie-Hellman public key.lastPhase - flag which indicates whether or not this is the last phase of this key agreement. **Returns:**the (intermediate) key resulting from this phase, or null if this phase does not yield a key **Throws:** [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html) - if the given key is inappropriate for this phase. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this key agreement has not been initialized.

### generateSecret

public final byte[] **generateSecret**()  
 throws [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html)

Generates the shared secret and returns it in a new buffer.

This method resets this KeyAgreement object, so that it can be reused for further key agreements. Unless this key agreement is reinitialized with one of the init methods, the same private information and algorithm parameters will be used for subsequent key agreements.

**Returns:**the new buffer with the shared secret **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this key agreement has not been completed yet

### generateSecret

public final int **generateSecret**(byte[] sharedSecret,  
 int offset)  
 throws [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html),  
 [ShortBufferException](http://docs.google.com/javax/crypto/ShortBufferException.html)

Generates the shared secret, and places it into the buffer sharedSecret, beginning at offset inclusive.

If the sharedSecret buffer is too small to hold the result, a ShortBufferException is thrown. In this case, this call should be repeated with a larger output buffer.

This method resets this KeyAgreement object, so that it can be reused for further key agreements. Unless this key agreement is reinitialized with one of the init methods, the same private information and algorithm parameters will be used for subsequent key agreements.

**Parameters:**sharedSecret - the buffer for the shared secretoffset - the offset in sharedSecret where the shared secret will be stored **Returns:**the number of bytes placed into sharedSecret **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this key agreement has not been completed yet [ShortBufferException](http://docs.google.com/javax/crypto/ShortBufferException.html) - if the given output buffer is too small to hold the secret

### generateSecret

public final [SecretKey](http://docs.google.com/javax/crypto/SecretKey.html) **generateSecret**([String](http://docs.google.com/java/lang/String.html) algorithm)  
 throws [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html),  
 [NoSuchAlgorithmException](http://docs.google.com/java/security/NoSuchAlgorithmException.html),  
 [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html)

Creates the shared secret and returns it as a SecretKey object of the specified algorithm.

This method resets this KeyAgreement object, so that it can be reused for further key agreements. Unless this key agreement is reinitialized with one of the init methods, the same private information and algorithm parameters will be used for subsequent key agreements.

**Parameters:**algorithm - the requested secret-key algorithm **Returns:**the shared secret key **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this key agreement has not been completed yet [NoSuchAlgorithmException](http://docs.google.com/java/security/NoSuchAlgorithmException.html) - if the specified secret-key algorithm is not available [InvalidKeyException](http://docs.google.com/java/security/InvalidKeyException.html) - if the shared secret-key material cannot be used to generate a secret key of the specified algorithm (e.g., the key material is too short)

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/KeyAgreement.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/crypto/IllegalBlockSizeException.html)   [**NEXT CLASS**](http://docs.google.com/javax/crypto/KeyAgreementSpi.html) | [**FRAMES**](http://docs.google.com/index.html?javax/crypto/KeyAgreement.html)    [**NO FRAMES**](http://docs.google.com/KeyAgreement.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#4d34og8) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).