Module java.base
Package java.security

Class KeyPairGenerator

java.lang.Object
    java.security.KeyPairGeneratorSpi
        java.security.KeyPairGenerator

```
public abstract class KeyPairGenerator
extends KeyPairGeneratorSpi
```

The `KeyPairGenerator` class is used to generate pairs of public and private keys. Key pair generators are constructed using the `getInstance` factory methods (static methods that return instances of a given class).

A Key pair generator for a particular algorithm creates a public/private key pair that can be used with this algorithm. It also associates algorithm-specific parameters with each of the generated keys.

There are two ways to generate a key pair: in an algorithm-independent manner, and in an algorithm-specific manner. The only difference between the two is the initialization of the object:

- **Algorithm-Independent Initialization**

  All key pair generators share the concepts of a keysize and a source of randomness. The keysize is interpreted differently for different algorithms (e.g., in the case of the *DSA* algorithm, the keysize corresponds to the length of the modulus). There is an `initialize` method in this `KeyPairGenerator` class that takes these two universally shared types of arguments. There is also one that takes just a `keysize` argument, and uses the `SecureRandom` 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 is used.)

  Since no other parameters are specified when you call the above algorithm-independent `initialize` methods, it is up to the provider what to do about the algorithm-specific parameters (if any) to be associated with each of the keys.

  If the algorithm is the *DSA* algorithm, and the keysize (modulus size) is 512, 768, 1024, or 2048, then the *Sun* provider uses a set of precomputed values for the `p`, `q`, and `g` parameters. If the modulus size is not one of the above values, the *Sun* provider creates a new set of parameters. Other providers might have precomputed parameter sets for more than just the modulus sizes mentioned above. Still others might not have a list of precomputed parameters at all and instead always create new parameter sets.

- **Algorithm-Specific Initialization**

  For situations where a set of algorithm-specific parameters already exists (e.g., so-called *community parameters* in DSA), there are two `initialize` methods that have an `AlgorithmParameterSpec` argument. One also has a `SecureRandom` argument, while the other uses the `SecureRandom` 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 is used.)

In case the client does not explicitly initialize the `KeyPairGenerator` (via a call to an `initialize` method), each provider must supply (and document) a default initialization. See the Keysize Restriction sections of the JDK Providers document for information on the `KeyPairGenerator` defaults used by JDK providers. However, note that defaults may vary across different providers. Additionally, the default value for a provider may change in a future version. Therefore, it is recommended to explicitly initialize the `KeyPairGenerator` instead of relying on provider-specific defaults.

Note that this class is abstract and extends from `KeyPairGeneratorSpi` for historical reasons. Application developers should only take notice of the methods defined in this `KeyPairGenerator` class; all the methods in the superclass are intended for cryptographic service providers who wish to supply their own implementations of key pair generators.

Every implementation of the Java platform is required to support the following standard `KeyPairGenerator` algorithms and keysizes in parentheses:

- `DiffieHellman` (1024, 2048, 4096)
- `DSA` (1024, 2048)
- `RSA` (1024, 2048, 4096)

These algorithms are described in the KeyPairGenerator section of the Java Security Standard Algorithm Names Specification. Consult the release documentation for your implementation to see if any other algorithms are supported.

Since:

1.1

See Also:

`AlgorithmParameterSpec`

Constructor Summary

Constructors

Modifier	Constructor	Description
`protected`	`KeyPairGenerator(String algorithm)`	Creates a `KeyPairGenerator` object for the specified algorithm.

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
`KeyPair`	`generateKeyPair()`	Generates a key pair.
`final KeyPair`	`genKeyPair()`	Generates a key pair.
`String`	`getAlgorithm()`	Returns the standard name of the algorithm for this key pair generator.
`static KeyPairGenerator`	`getInstance(String algorithm)`	Returns a `KeyPairGenerator` object that generates public/private key pairs for the specified algorithm.
`static KeyPairGenerator`	`getInstance(String algorithm, String provider)`	Returns a `KeyPairGenerator` object that generates public/private key pairs for the specified algorithm.
`static KeyPairGenerator`	`getInstance(String algorithm, Provider provider)`	Returns a `KeyPairGenerator` object that generates public/private key pairs for the specified algorithm.
`final Provider`	`getProvider()`	Returns the provider of this key pair generator object.
`void`	`initialize(int keysize)`	Initializes the key pair generator for a certain keysize using a default parameter set and the `SecureRandom` implementation of the highest-priority installed provider as the source of randomness.
`void`	`initialize(int keysize, SecureRandom random)`	Initializes the key pair generator for a certain keysize with the given source of randomness (and

		a default parameter set).
`void`	`initialize(AlgorithmParameterSpec params)`	Initializes the key pair generator using the specified parameter set and the `SecureRandom` implementation of the highest-priority installed provider as the source of randomness.
`void`	`initialize(AlgorithmParameterSpec params, SecureRandom random)`	Initializes the key pair generator with the given parameter set and source of randomness.

Methods declared in class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Details

KeyPairGenerator

`protected KeyPairGenerator(String algorithm)`

Creates a `KeyPairGenerator` object for the specified algorithm.

Parameters:

`algorithm` - the standard string name of the algorithm. See the KeyPairGenerator section in the Java Security Standard Algorithm Names Specification for information about standard algorithm names.

Method Details

getAlgorithm

`public String getAlgorithm()`

Returns the standard name of the algorithm for this key pair generator. See the KeyPairGenerator section in the Java Security Standard Algorithm Names Specification for information about standard algorithm names.

Returns:

the standard string name of the algorithm.

getInstance

```
public static KeyPairGenerator getInstance(String algorithm)
                                    throws NoSuchAlgorithmException
```

Returns a `KeyPairGenerator` object that generates public/private key pairs for the specified algorithm.

This method traverses the list of registered security Providers, starting with the most preferred Provider. A new `KeyPairGenerator` object encapsulating the `KeyPairGeneratorSpi` 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()` method.

Implementation Note:

The JDK Reference Implementation additionally uses the `jdk.security.provider.preferred` Security property to determine the preferred provider order for the specified algorithm. This may be different from the order of providers returned by `Security.getProviders()`.

Parameters:

`algorithm` - the standard string name of the algorithm. See the KeyPairGenerator section in the Java Security Standard Algorithm Names Specification for information about standard algorithm names.

Returns:

the new `KeyPairGenerator` object

Throws:

`NoSuchAlgorithmException` - if no `Provider` supports a `KeyPairGeneratorSpi` implementation for the specified algorithm

`NullPointerException` - if `algorithm` is `null`

See Also:

Provider

getInstance

```
public static KeyPairGenerator getInstance(String algorithm,
                                           String provider)
                                    throws NoSuchAlgorithmException,
                                           NoSuchProviderException
```

Returns a `KeyPairGenerator` object that generates public/private key pairs for the specified algorithm.

A new `KeyPairGenerator` object encapsulating the `KeyPairGeneratorSpi` 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()` method.

Parameters:

`algorithm` - the standard string name of the algorithm. See the KeyPairGenerator section in the Java Security Standard Algorithm Names Specification for information about standard algorithm names.

`provider` - the string name of the provider.

Returns:

the new `KeyPairGenerator` object

Throws:

`IllegalArgumentException` - if the provider name is `null` or empty

`NoSuchAlgorithmException` - if a `KeyPairGeneratorSpi` implementation for the specified algorithm is not available from the specified provider

`NoSuchProviderException` - if the specified provider is not registered in the security provider list

`NullPointerException` - if `algorithm` is `null`

See Also:

`Provider`

getInstance

```
public static KeyPairGenerator getInstance(String algorithm,
                                           Provider provider)
                                    throws NoSuchAlgorithmException
```

Returns a KeyPairGenerator object that generates public/private key pairs for the specified algorithm.

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

Parameters:

algorithm - the standard string name of the algorithm. See the KeyPairGenerator section in the Java Security Standard Algorithm Names Specification for information about standard algorithm names.

provider - the provider.

Returns:

the new KeyPairGenerator object

Throws:

IllegalArgumentException - if the specified provider is null

NoSuchAlgorithmException - if a KeyPairGeneratorSpi implementation for the specified algorithm is not available from the specified Provider object

NullPointerException - if algorithm is null

Since:

1.4

See Also:

Provider

getProvider

```
public final Provider getProvider()
```

Returns the provider of this key pair generator object.

Returns:

the provider of this key pair generator object

initialize

```
public void initialize(int keysize)
```

Initializes the key pair generator for a certain keysize using a default parameter set and the `SecureRandom` 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 is used.)

Parameters:

`keysize` - the keysize. This is an algorithm-specific metric, such as modulus length, specified in number of bits.

Throws:

`InvalidParameterException` - if the `keysize` is not supported by this `KeyPairGenerator` object.

initialize

```
public void initialize(int keysize,
                       SecureRandom random)
```

Initializes the key pair generator for a certain keysize with the given source of randomness (and a default parameter set).

Specified by:

`initialize` in class `KeyPairGeneratorSpi`

Parameters:

`keysize` - the keysize. This is an algorithm-specific metric, such as modulus length, specified in number of bits.

`random` - the source of randomness.

Throws:

`InvalidParameterException` - if the `keysize` is not supported by this `KeyPairGenerator` object.

Since:

1.2

initialize

```
public void initialize(AlgorithmParameterSpec params)
                throws InvalidAlgorithmParameterException
```

Initializes the key pair generator using the specified parameter set and the `SecureRandom` 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 is used.)

This concrete method has been added to this previously-defined abstract class. This method calls the KeyPairGeneratorSpi `initialize` method, passing it `params` and a source of randomness (obtained from the highest-priority installed provider or system-provided if none of the installed providers supply one). That `initialize` method always throws an `UnsupportedOperationException` if it is not overridden by the provider.

Parameters:

`params` - the parameter set used to generate the keys.

Throws:

`InvalidAlgorithmParameterException` - if the given parameters are inappropriate for this key pair generator.

Since:

1.2

initialize

```
public void initialize(AlgorithmParameterSpec params,
                       SecureRandom random)
                throws InvalidAlgorithmParameterException
```

Initializes the key pair generator with the given parameter set and source of randomness.

This concrete method has been added to this previously-defined abstract class. This method calls the KeyPairGeneratorSpi `initialize` method, passing it `params` and `random`. That `initialize` method always throws an `UnsupportedOperationException` if it is not overridden by the provider.

Overrides:

`initialize` in class `KeyPairGeneratorSpi`

Parameters:

`params` - the parameter set used to generate the keys.

`random` - the source of randomness.

Throws:

`InvalidAlgorithmParameterException` - if the given parameters are inappropriate for this key pair generator.

Since:

1.2

genKeyPair

```
public final KeyPair genKeyPair()
```

Generates a key pair.

If this `KeyPairGenerator` has not been initialized explicitly, provider-specific defaults will be used for the size and other (algorithm-specific) values of the generated keys.

This will generate a new key pair every time it is called.

This method is functionally equivalent to `generateKeyPair`.

Returns:

the generated key pair

Since:

1.2

generateKeyPair

```
public KeyPair generateKeyPair()
```

Generates a key pair.

If this `KeyPairGenerator` has not been initialized explicitly, provider-specific defaults will be used for the size and other (algorithm-specific) values of the generated keys.

This will generate a new key pair every time it is called.

This method is functionally equivalent to `genKeyPair`.

Specified by:

`generateKeyPair` in class `KeyPairGeneratorSpi`

Returns:

the generated key pair