**Java Certificate**

The *Java Certificate* class (java.security.cert.Certificate) represents a cryptographic identity certificate. A Java Certificate class instance contains name plus other details of the entity it identifies, plus possibly a digital signature from a Certificate Authority (CA).

The Java Certificate class is an abstract class, so while you may use Certificate as variable type, your variable will always point to a subclass of Certificate.

The Java Certificate class has one subclass - the X509Certificate class. This class represents an X.509certificate which is used as identity certificate in HTTPS and TLS.

**Obtaining a Certificate Instance**

You can obtain a Certificate instance in the following ways:

* From a [**CertificateFactory**](http://tutorials.jenkov.com/java-cryptography/certificatefactory.html).
* From a [**KeyStore**](http://tutorials.jenkov.com/java-cryptography/keystore.html).

See these two tutorials for more information about obtaining a Certificate instance.

**getEncoded()**

The Java Certificate getEncoded() method returns an encoded version of the Certificate as a byte array. For instance, if the Certificate is an X509Certificate the returned byte array will contain an X.590 (ASN.1 DER) encoded version of the Certificate instance. Here is a getEncoded() example:

byte[] encodedCertificate = certificate.getEncoded();

**getPublicKey()**

The Java Certificate getPublicKey() method returns the PublicKey of this Certificate instance. Here is a getPublicKey() example:

PublicKey certificatePublicKey = certificate.getPublicKey();

**getType()**

The Java Certificate getType() method returns the type of the Certificate instance. Here is a getType()example:

String certificateType = certificate.getType();

**verify()**

The Java Certificate class contains three verify() methods. These methods can be used to verify that the Certificate is really signed with the private key matching the expected public key. Here is a Java Certificate verify() example:

// get expected public key from somewhere else (not Certificate instance !!)

PublicKey expectedPublicKey = ... ;

try{

certificate.verify(expectedPublicKey);

} catch (InvalidKeyException e) {

// certificate was not signed with given public key

} catch (NoSuchAlgorithmException |

NoSuchProviderException |

SignatureException |

CertificateException e){

// something else went wrong

}

The verify() method returns void. If the verification fails, an InvalidKeyException will be thrown. If no exception is thrown the Certificate instance can be considered verified.

## The CertificateFactory Class

The CertificateFactory class is an [engine class](https://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html#Engine) that defines the functionality of a certificate factory, which is used to generate certificate and certificate revocation list (CRL) objects from their encodings.

A certificate factory for X.509 must return certificates that are an instance of java.security.cert.X509Certificate, and CRLs that are an instance of java.security.cert.X509CRL.

### Creating a CertificateFactory Object

CertificateFactory objects are obtained by using one of the [getInstance() static factory methods](https://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html#ProviderImplReq).

### Generating Certificate Objects

To generate a certificate object and initialize it with the data read from an input stream, use the generateCertificate method:

final Certificate generateCertificate(InputStream inStream)

To return a (possibly empty) collection view of the certificates read from a given input stream, use the generateCertificates method:

final Collection generateCertificates(InputStream inStream)

### Generating CRL Objects

To generate a certificate revocation list (CRL) object and initialize it with the data read from an input stream, use the generateCRL method:

final CRL generateCRL(InputStream inStream)

To return a (possibly empty) collection view of the CRLs read from a given input stream, use the generateCRLs method:

final Collection generateCRLs(InputStream inStream)

### Generating CertPath Objects

The certificate path builder and validator for PKIX is defined by the Internet X.509 Public Key Infrastructure Certificate and CRL Profile, [RFC 3280](http://www.ietf.org/rfc/rfc3280.txt).

A certificate store implementation for retrieving certificates and CRLs from Collection and LDAP directories, using the PKIX LDAP V2 Schema is also available from the IETF as [RFC 2587](http://www.ietf.org/rfc/rfc2587.txt).

To generate a CertPath object and initialize it with data read from an input stream, use one of the following generateCertPath methods (with or without specifying the encoding to be used for the data):

final CertPath generateCertPath(InputStream inStream)

final CertPath generateCertPath(InputStream inStream,

String encoding)

To generate a CertPath object and initialize it with a list of certificates, use the following method:

final CertPath generateCertPath(List certificates)

To retrieve a list of the CertPath encodings supported by this certificate factory, you can call the getCertPathEncodings method:

final Iterator getCertPathEncodings()