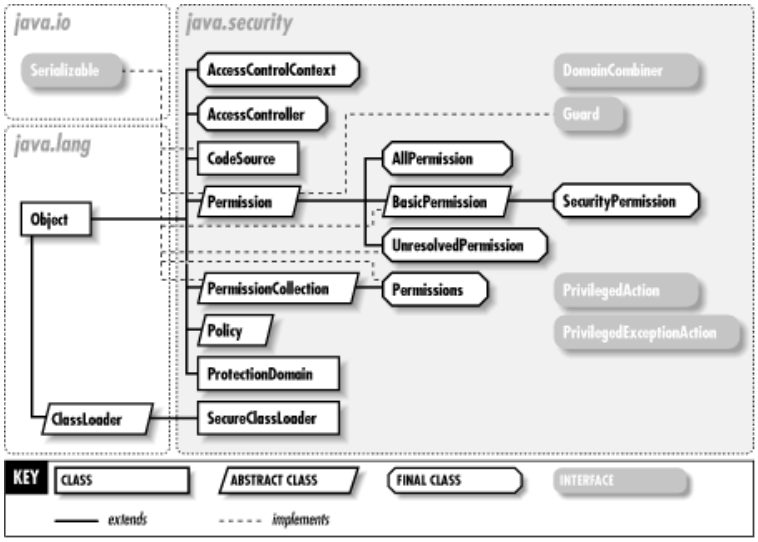
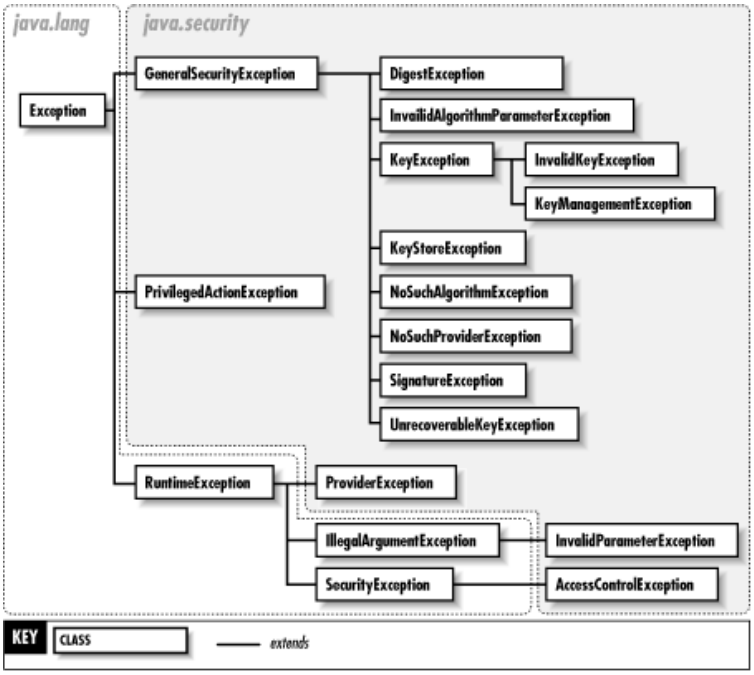
java.security

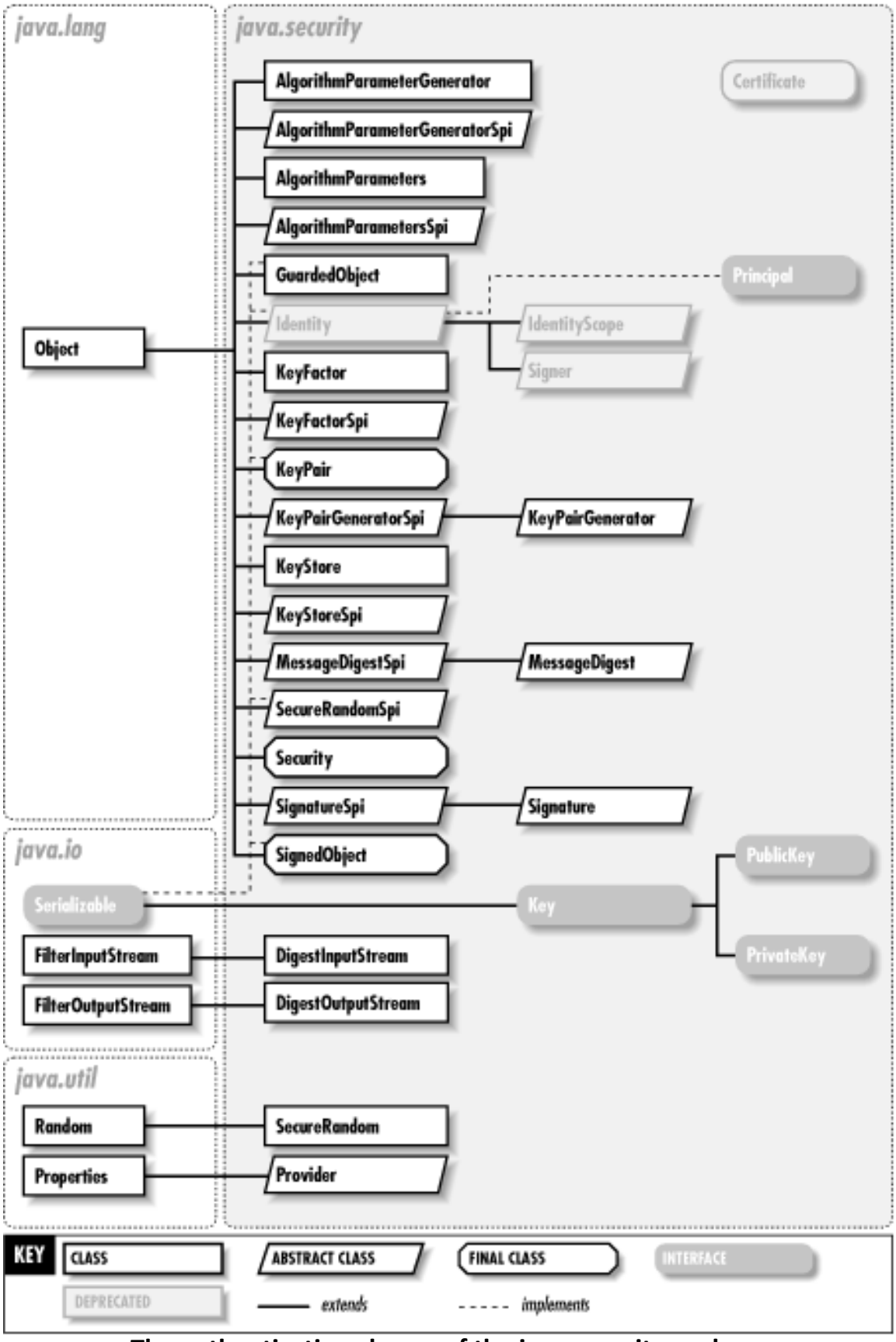
- The java.security package contains the classes and interfaces that implement the Java security architecture. These classes can be divided into two broad categories.
- First, there are classes that implement access control and prevent untrusted code from performing sensitive operations. Second, there are authentication classes that implement message digests and digital signatures and can authenticate Java classes and other objects.
- The central access control class is AccessController; it uses the currently installed Policy object to decide whether a given class has Permission to access a given system resource. The Permissions and ProtectionDomain classes are also important pieces of the Java access control architecture.
- The key classes for authentication are MessageDigest and Signature; they compute and verify cryptographic message digests and digital signatures. These classes use public-key cryptography techniques and rely on the PublicKey and PrivateKey classes. They also rely on an infrastructure of related classes, such as SecureRandom for producing cryptographic-strength pseudo-random numbers, KeyPairGenerator for generating pairs of public and private keys, and KeyStore for managing a collection of keys and certificates.



The access control classes of the java.security package



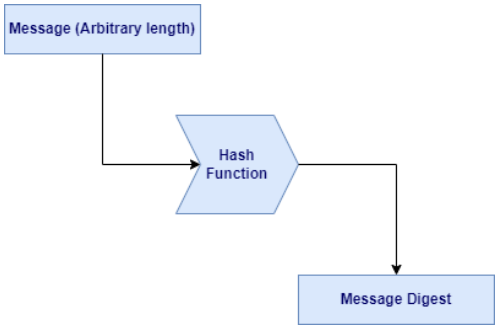
The exception classes of the java.security package



The authentication classes of the java.security package

MessageDigest

- MessageDigest is the returned value of the hash function, which is also known as has values. Hash functions are mostly used in each and every information security application. Hash functions are used for converting numerical values into compressed numerical values. For Hash functions, the length of the user-given input can be arbitrary, but the length of the output is always of fixed length.
- The java.security package provides a class, i.e., MessageDigest, that supports algorithms such as SHA-1, SHA 256, and MD5 etc., for converting a message of arbitrary length to a message digest.




```
import java.security.MessageDigest;
import java.util.Scanner;
public class Test
{
	public static void main(String args[]) throws Exception
	{
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter the message of any arbitrary length:");
		String msg = sc.nextLine();
		sc.close();

		MessageDigest obj = MessageDigest.getInstance("SHA-256");
		obj.update(msg.getBytes());
		byte[] byteArray = obj.digest();
		System.out.println(byteArray);
		StringBuffer hexData = new StringBuffer();

		for (int i = 0; i < byteArray.length; i++)
		{
			hexData.append(Integer.toHexString(0xFF & byteArray[i]));
		}

		System.out.println("Data in Hex format : " + hexData.toString());
   }
}
```

```
C:\Program Files\Java\jdk-11.0.12\bin\Manish>javac Test.java

C:\Program Files\Java\jdk-11.0.12\bin\Manish>java Test
Enter the message of any arbitrary length:
Manish@1
[B@548b7f67
Data in Hex format : e7afba7e19763e54956426cd279a9172c0a465f3402951db311f517a3f3bba68
```

……

```
import java.io.*;
import java.security.*;
class Test
{
	public static void main(String args[])
	{
		try
		{
			MessageDigest md = MessageDigest.getInstance("SHA");
			String s1 = "Manish@1";
			byte[] array = s1.getBytes();
			md.update(array);
			FileOutputStream fos =new FileOutputStream("Test.key");
			ObjectOutputStream oos =new ObjectOutputStream(fos);
			oos.writeObject(md.digest());
			System.out.println(" Hurry.. digest ready!");
		}
		catch(Exception e1)
		{
			System.out.println(""+e1);
}	}	}
```