# Course Name:
# Advanced Java

# Lecture 26
# Topics to be covered

- Digital Signatures
- Code Signing
- Encryption

# JDK 1.1 - JCA

- Introduced Java Cryptography Architecture (JCA)
  - Allows multiple and interoperable cryptography implementations
- Cryptographic Service Provider (CSP) or provider
  - Packages including: concrete implementation of subset of cryptography aspects of JDK security

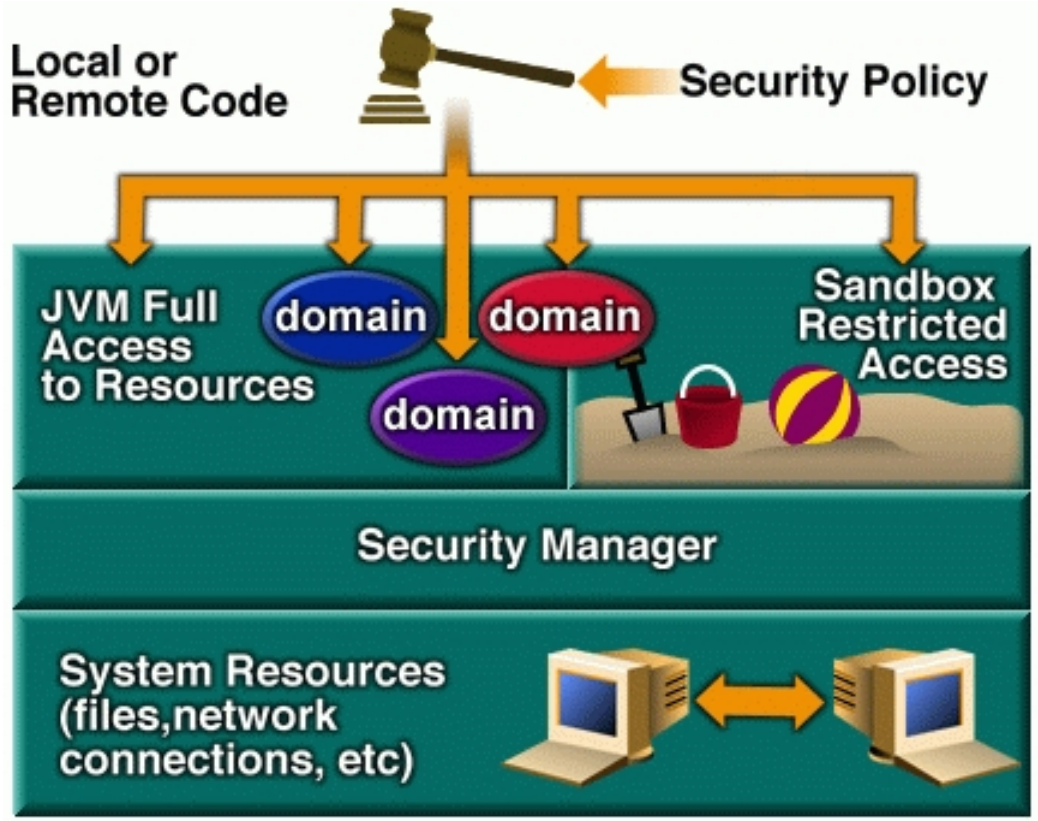
# JDK 1.1 - JCA

- A provider implementation can contain one or more
  - Digital signature algorithms
  - Message digest algorithms
  - Key generation algorithms

# JDK 1.2

- New concept: “Security policy”
  - Applies to all code
  - Defines the permissions for the code
  - Configurable by system administrator and user
- Each permission allows access to specific resource
  - i.e. connection to specific port or read/write access to a particular file

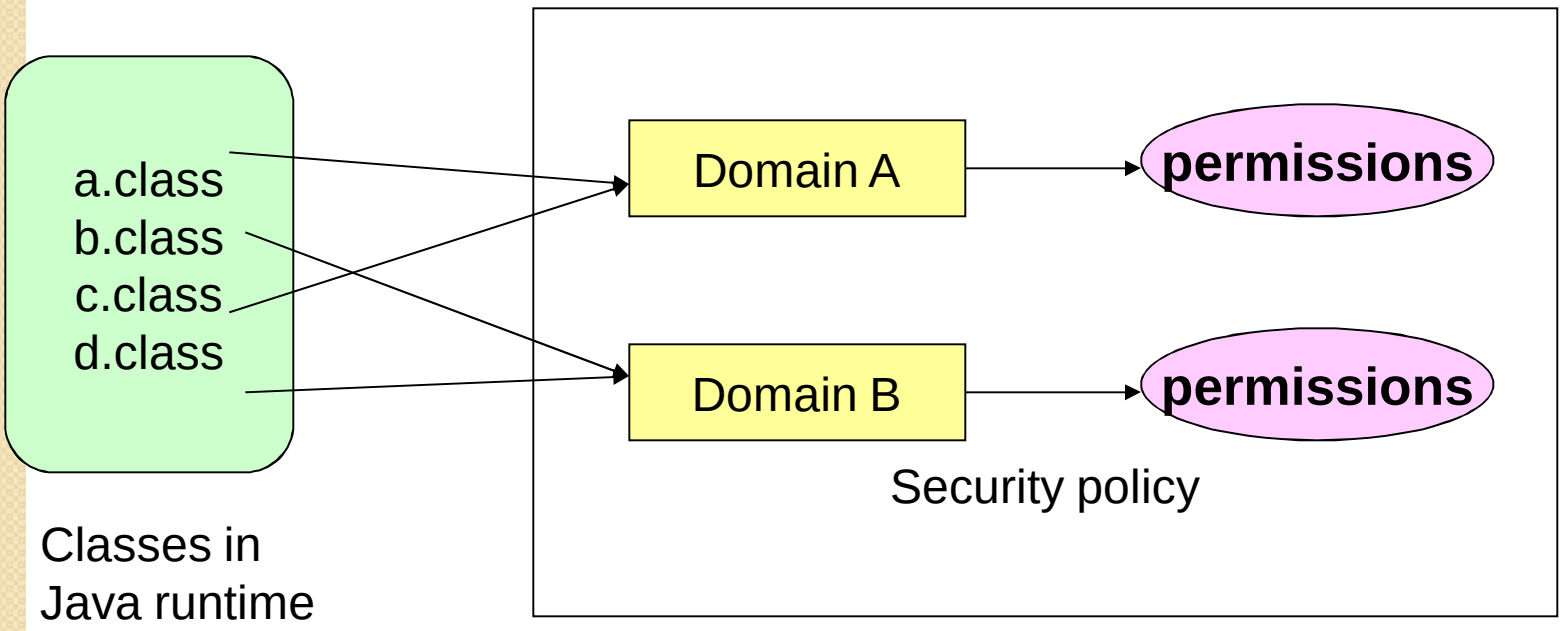
# JDK 1.2


Local or
Remote Code
Security Policy
JVM Full
Access
to Resources
domain
domain
domain
Sandbox
Restricted
Access
Security Manager
System Resources
(files,network
connections, etc)

# JDK 1.2

- Local or remote code's classes are organized into "domain"s
- Each domain has a specific permission
  - Instances of the classes in the same domain have the same permission
- Very restricted domains can be configured
- Applications run unrestricted
  - But the policy can be modified to put some restrictions

# JDK 1.2


a.class
b.class
c.class
d.class
Domain A
Domain B
permissions
permissions
Security policy
Classes in
Java runtime

# JDK 1.2 - JCA

- In addition to JCA1.1 features, it provides
  - Keystore – key creation and management
  - Algorithm parameter generation
  - Algorithm parameter management
  - Key factory support
  - Certificate factory support (generation and revocation)

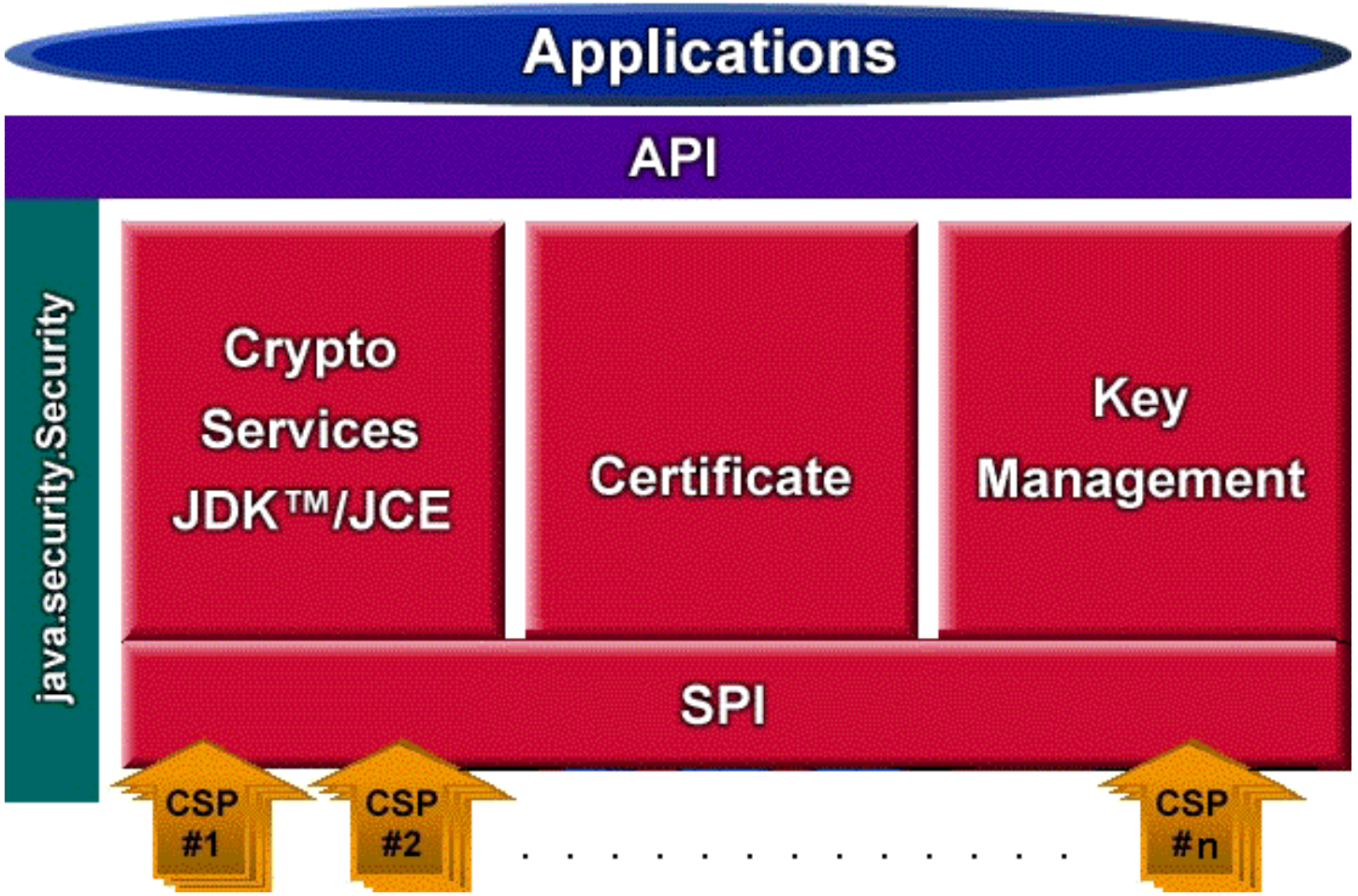
# JDK 1.2 - JCA

- Enables providers with a random-number generation (RNG) algorithm
- Includes a default provider (SUN)
  - DSA (Digital Signature Algorithm)
  - MD5 and SHA-1
  - Certificate factory for x509 certificates

# Cryptography Architecture Extensions (JCE)

- An separate extension to JDK
  - In accordance with US export control laws
- Includes API for
  - Encryption
  - Key exchange
  - MAC – Message Authentication Code

# JCA modules


Applications
API
java.security.Security
Crypto Services JDK™/JCE
Certificate
Key Management
SPI
CSP #1
CSP #2
CSP #n

# Engine Classes

- A class that defines API methods to access specific type of cryptographic service
  - Abstract, no concrete implementation
  - Applications use an instance of Signature engine class and the actual implementation is in Signaturespi class that, for instance, implements SHA-1 with RSA

# Java 1.2 security API

- java.security
  - General classes
- java.security.interface
  - Interfaces for RSA and DSA
- java.security.spec
  - Interfaces and classes for key/algorithm specifications
- java.security.cert
  - Classes for managing certificates
- java.security.acl
  - Classes for managing access control lists

# Key Generation (public key)

- Using RSA algorithm,1024 bit key pair

```
KeyPairGenerator kgen = KeyPairGenerator.getInstance(“RSA”);
//prefer java.security.SecureRandom() to java.util.Random
kgen.initialize(1024, new SecureRandom());
Keypair myPair = kgen.generateKeyPair();
```

# Encryption

- Encrypt using public key

```
String plainText[];
byte cipherText[][];

Cipher encrypt = Cipher.getInstance(“RSA/ECB/PKCS1Padding”);
encrypt.init(Cipher.ENCRYPT_MODE, myPair.getPublic());
int i;
for (i=0; i<plainText.length-1; i+=1) {
    cipherText[i] = encrypt.update(plainText[i].getBytes());
}
cipherText[i] = encrypt.doFinal(plainText[i].getBytes());
```

# Decryption

- Decrypt using private key

```
Cipher decrypt = Cipher.getInstance(“RSA/ECB/PKCS1Padding”);
decrypt.init(Cipher.DECRYPT_MODE, myPair.getPrivate());
int i;
byte[] decrypted;
for (i=0; i<cipherText.length-1; i+=1) {
    decrypted= decrypt.update(cipherText[i]);
}
decrypt = decrypt.doFinal(cipherText[i]);
```

# Message Digest

- Java.security.MessageDigest

```
MessageDigest md = MessageDigest.getInstance(“SHA-1”);
//supply input
md.update(byteArray);
//create message digest
Byte [] mdcode = md.digest();
```

- The message digest object is automatically reset after this call

# Digital Signing

- Sender signs with private key

```
Signature rsa = Signature.getInstance("RSA");
//initialize with private key
PrivateKey privateKey = myKeyPair.getPrivate();
rsa.initSign(privateKey);

//update the signature object with the data to be signed
rsa.update(someData);
//sign the data
byte[] signature = rsa.sign();
```

# Digital Verification

- Receiver verifies data with public key

```
PublicKey pubKey = aKeyPair.getPublic();
rsa.initVerify(pubKey);
//update the signature object with the data to be verified
rsa.update(data);
//verify the signature
Boolean isVerified = rsa.verify(sig);
```