Java Card 3 Programming

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Presentation objectives

- Introducing the concepts and the technology of the smart cards
- Describing the protocols between cards and terminals
- Describing how to program the Java Cards
- Exploring the tools and the environments provided by the manufacturers to develop solutions with smart cards

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Presentation content

- Introduction
- ISO7816 Protocol
- Java Card
- The basic rules for Java Card programming
- Cyphering
- SIM Card
- Smart Card Web Server
- Java Card 3.0 Connected Edition
- Conclusion

Introduction

History, technology, standards

Introduction

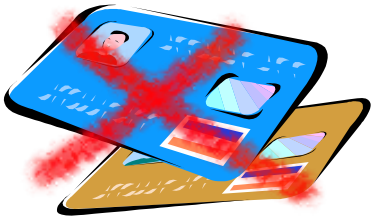
- In this chapter, we'll see
  - A brief history of the smart cards
  - The applications supported by the smart cards
  - The standards supported

Brief history

- Early seventies, first patents
  - Dr Arimura, R Moreno, M Ugon
- Early eighties, first field testing for a memory card
  - Phone card in France
- Mid eighties, large scale introduction of smart cards in banking system
- Mid nineties, SIM card introduced in mobile telephony

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

What is a smart card



- A plastic card like a credit card with an embedded micro chip
  - With or without visible contacts
    - Maybe contactless
- Standardized
  - ISO 7816
    - Mecanical properties
    - Electrical behavior
    - Communication protocol
- Contains a software which
  - Protects internal data
  - Give access to these data in a secure way

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

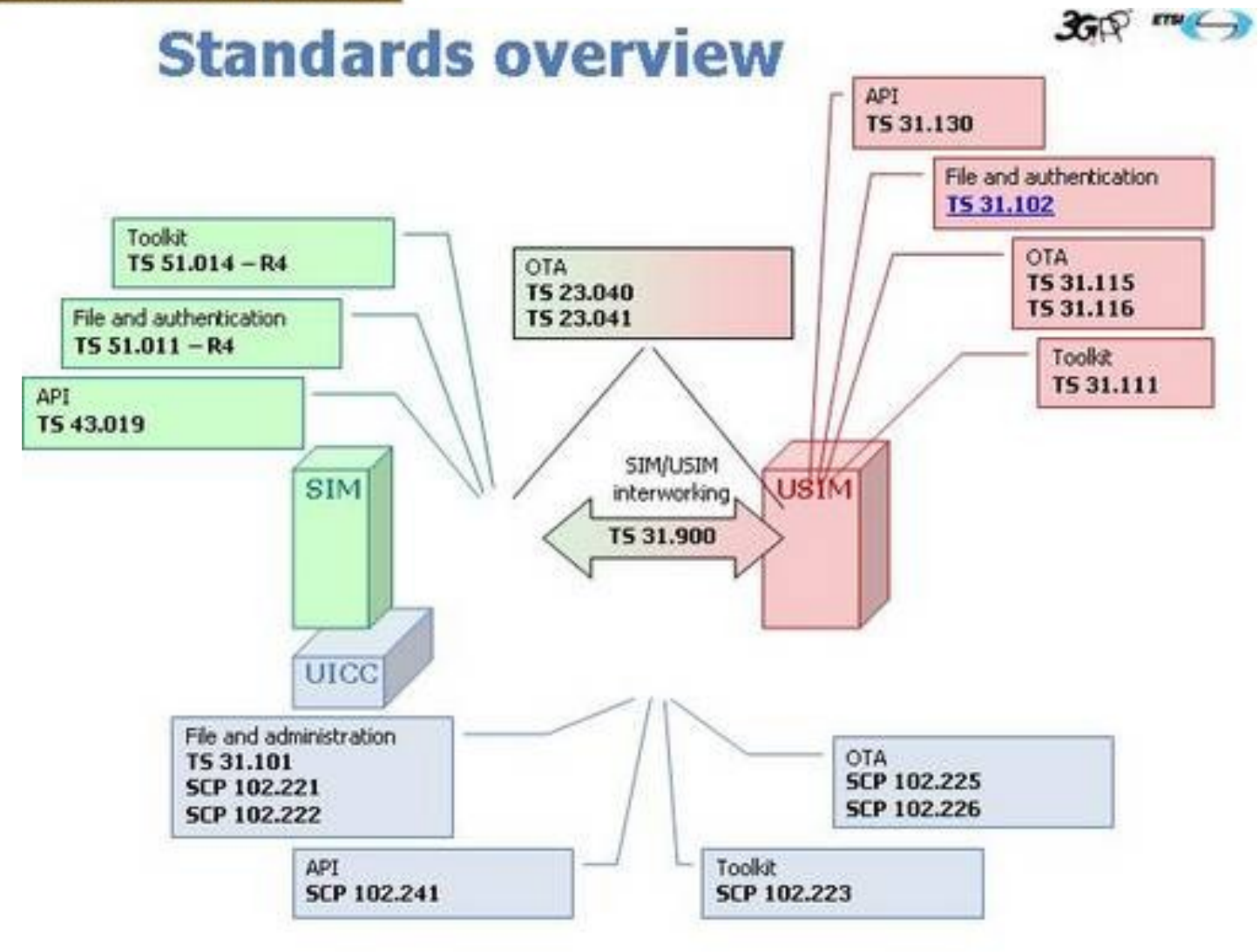
For what applications …

- Payment
- Loyalty systems
- Access systems
- Telephony
  - Mobile (GSM …)
- e-Government
  - ID card, passport
- File system
  - Health
  - Education
  - …

Standards

- **ISO 7816**
- **GSM 11.11 V6.1.0**
- **GSM 11.14 V7.1.0**
  - SIM Toolkit specs
- **GSM 03.19 V1.0.0**
  - SIM API for Javacard
- **ETSI**
  - TS 31 130
  - TS 102 241
  - TS 102 588
- **Java Card**
  - Java Card Forum
- **EMV**
  - Europay, Mastercard, Visa
- **Global Platform**

Standards


Standards overview
API
TS 31.130
File and authentication
TS 31.102
OTA
TS 31.115
TS 31.116
Toolkit
TS 31.111
Toolkit
TS 51.014 – R4
File and authentication
TS 51.011 – R4
API
TS 43.019
OTA
TS 23.040
TS 23.041
SIM/USIM
interworking
TS 31.900
SIM
USIM
UICC
File and administration
TS 31.101
SCP 102.221
SCP 102.222
OTA
SCP 102.225
SCP 102.226
API
SCP 102.241
Toolkit
SCP 102.223

Conclusion

- In this chapter, we have seen
  - A brief history of the smart cards
  - The applications supported by the smart cards
  - The standards supported

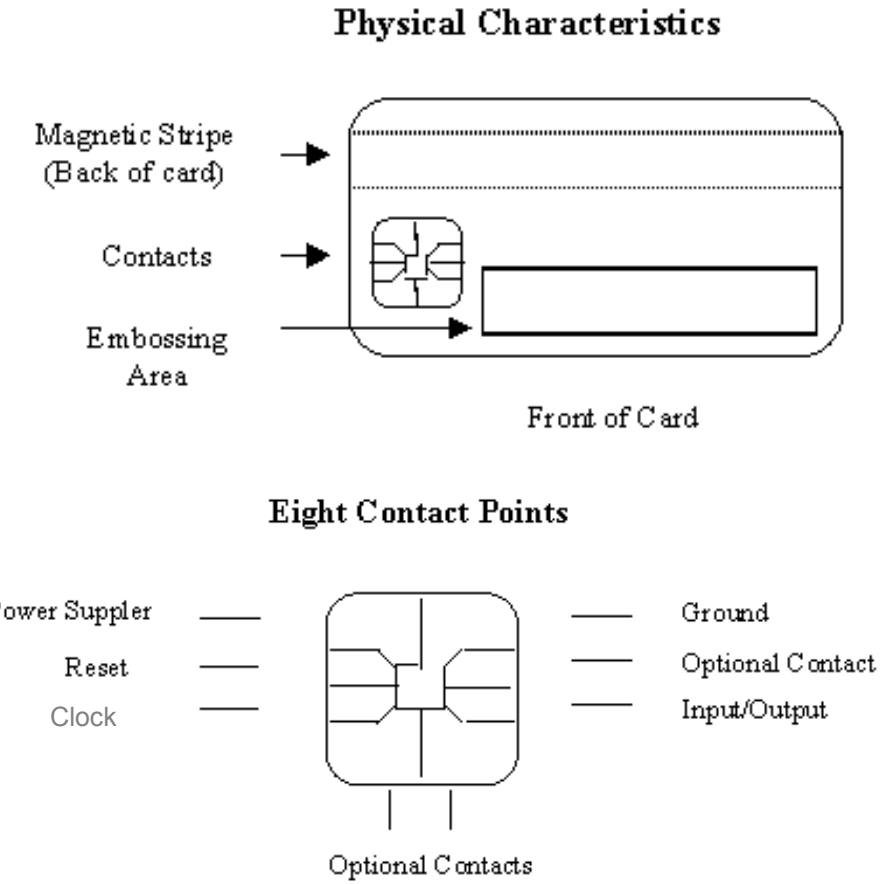
ISO7816 Protocol

Physical description, communication layer, file system

Introduction

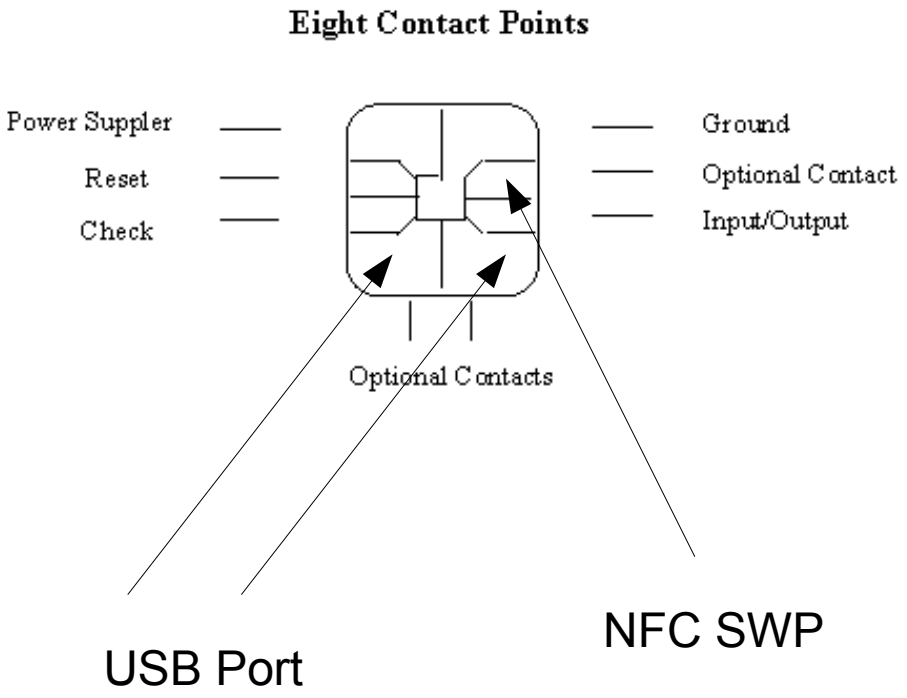
- In this chapter, we'll see
  - An introduction to the ISO7816 standard
  - What is an APDU
  - How to exchange data between the CAD and the smart card

Mechanical and Electrical Aspects



- ISO 7816 standard describes
  - The physical organisation of the plastic card
  - Indicates the various zones
- It specifies also the purpose and the organisation of the contacts
  - For a smart contacted card
- Possible power voltage
  - 3V or 5V
  - Lower maybe in the future

USB and NFC port



- Recent additions to the SIM card had standardized
  - A USB port in place of the two optionnal contacts on the bottom of the circuit
  - A NFC (Single Wire Protocol) port for the last optionnal contact

Half-duplex serial protocol

- Due to the unique pin dedicated to input/output, the first protocol used by the smart cards were
  - Serial
  - Half-duplex
- Communication characteristics:
  - Data: 8 bits
  - Parity: even
  - Stop: 1 bit
- Speed starting at 9600 Bps

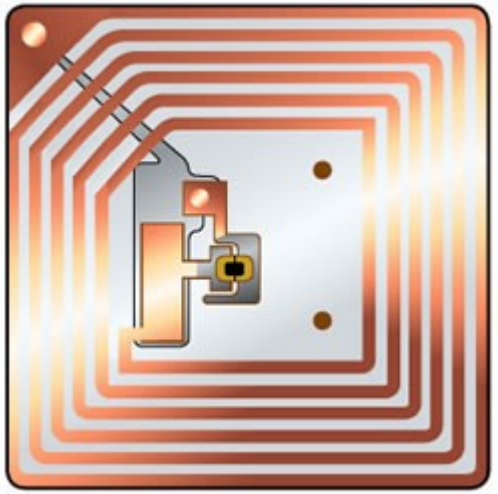
Smart University
State-of-the-Art ICT Knowledge and Training Workshops

USB IC
ETSI TS 102.600

- Max speed 12 Mb/s
- Three flavours:
  - Integrated Circuit Card Devices
    - Compatible with the previous serial protocol
  - Mass Storage
    - Disk emulation
  - Ethernet Emulation Mode
    - To support TCP/IP protocol

NFC SWP

ETSI TS 1002.613 & 622



- Single Wire Protocol
- Full duplex
  - Current and voltage modulation
- Max speed 1.6 Mb/s
- The smart card can act as
  - A RFID tag
  - A RFID tag reader

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Terminology

- The smart card reader powered by
  - a PC
  - A cash register
  - a mobile phone

  is called a **terminal**
- In the standard ISO 7816 it is called :
  - The **C**ard **A**cceptance **D**evice
  - Or CAD

Answer to Reset

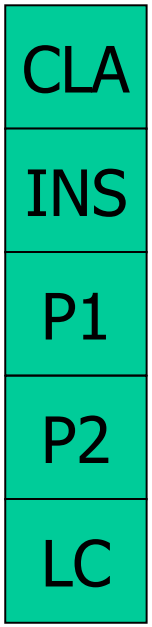
- When a card is inserted into the reader, a micro-switch signals this event to the terminal.
- The terminal powers up the card
  - Using a particular protocol
- When it is properly powered, the card sends back to the terminal a message called "**`Answer to Reset`**"

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

General protocol

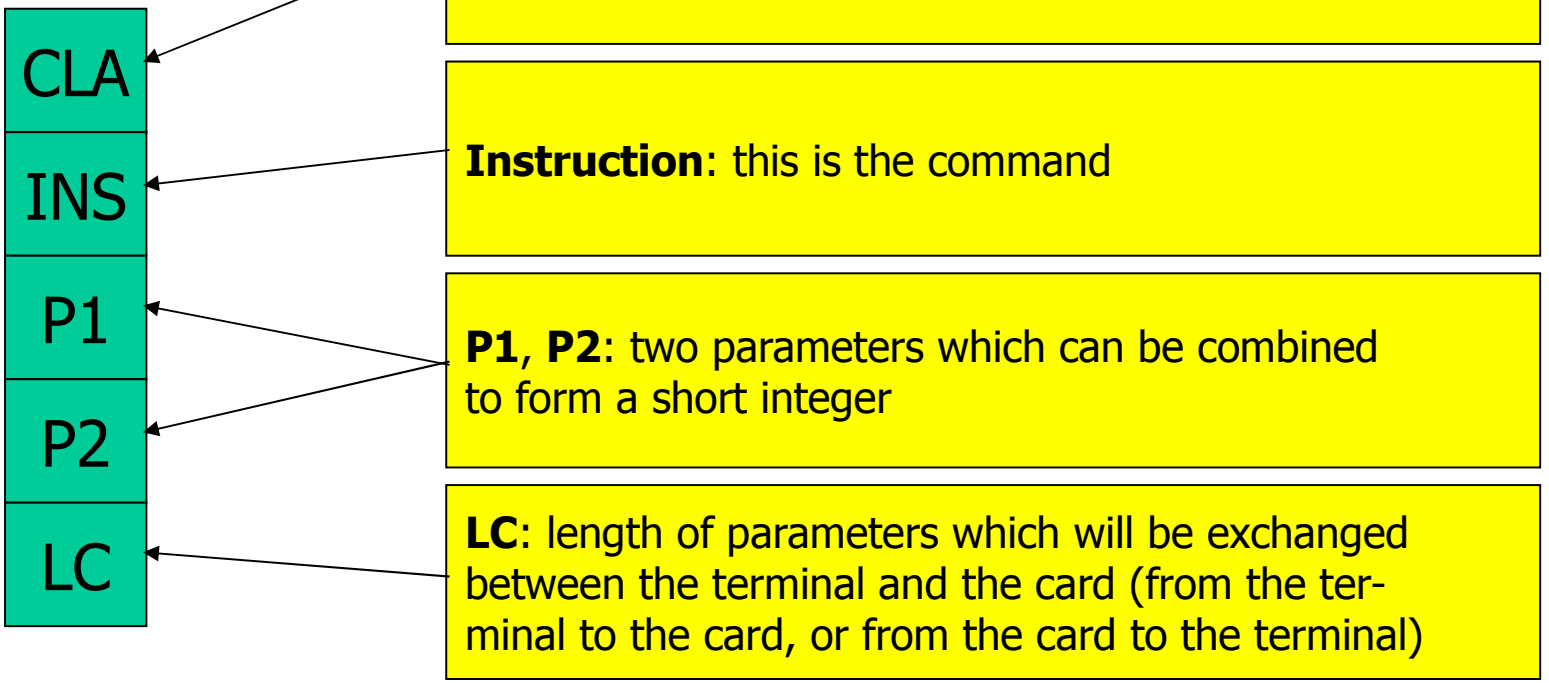
- After sending **`Answer to Reset`**, the card waits until the terminal starts a communication
- The card never starts a communication
- The card answers to a demand coming from the terminal and waits for the next demand

Application Protocol Data Unit

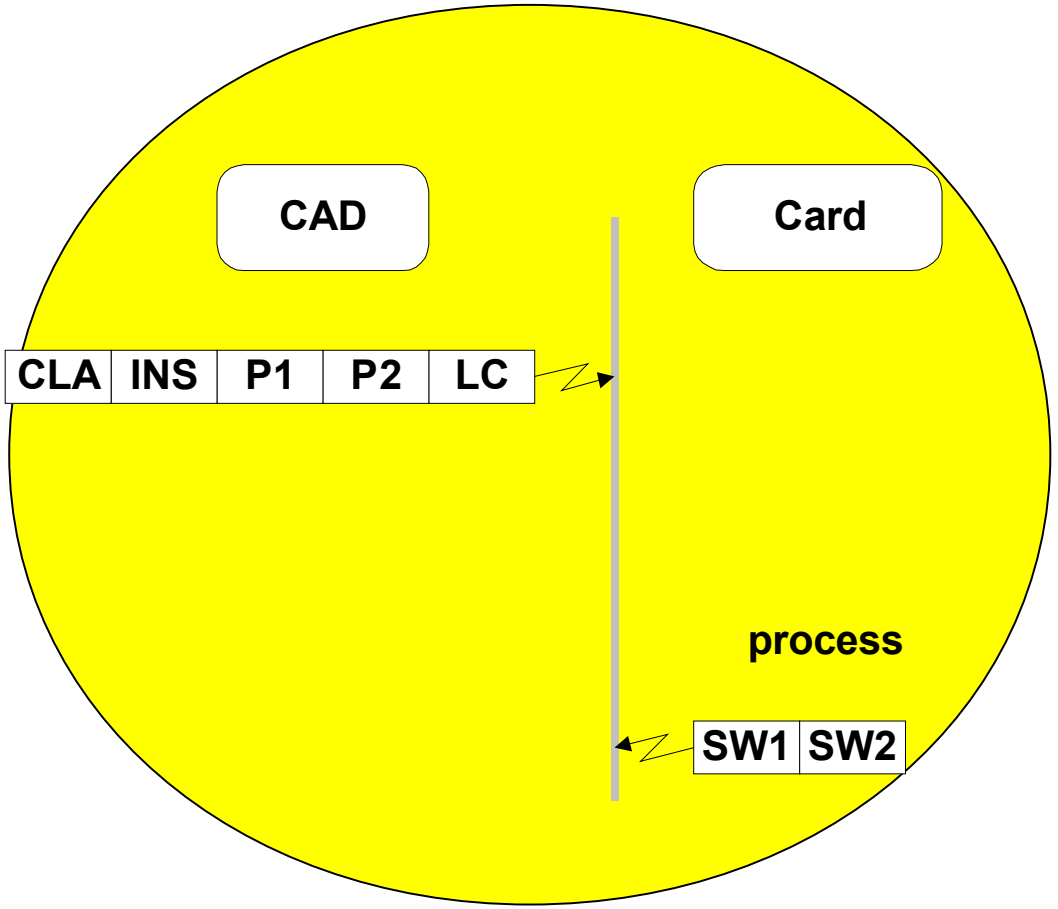


- The APDUs are the commands sent by the terminal to the smart card
- The APDU can
  - Carry parameters to the card
  - Expect results from the card
- Card and terminal must synchronize to
  - The number of bytes to exchange
  - The direction of the exchange
    - This is done by the software embedded in each device

Application Protocol Data Unit

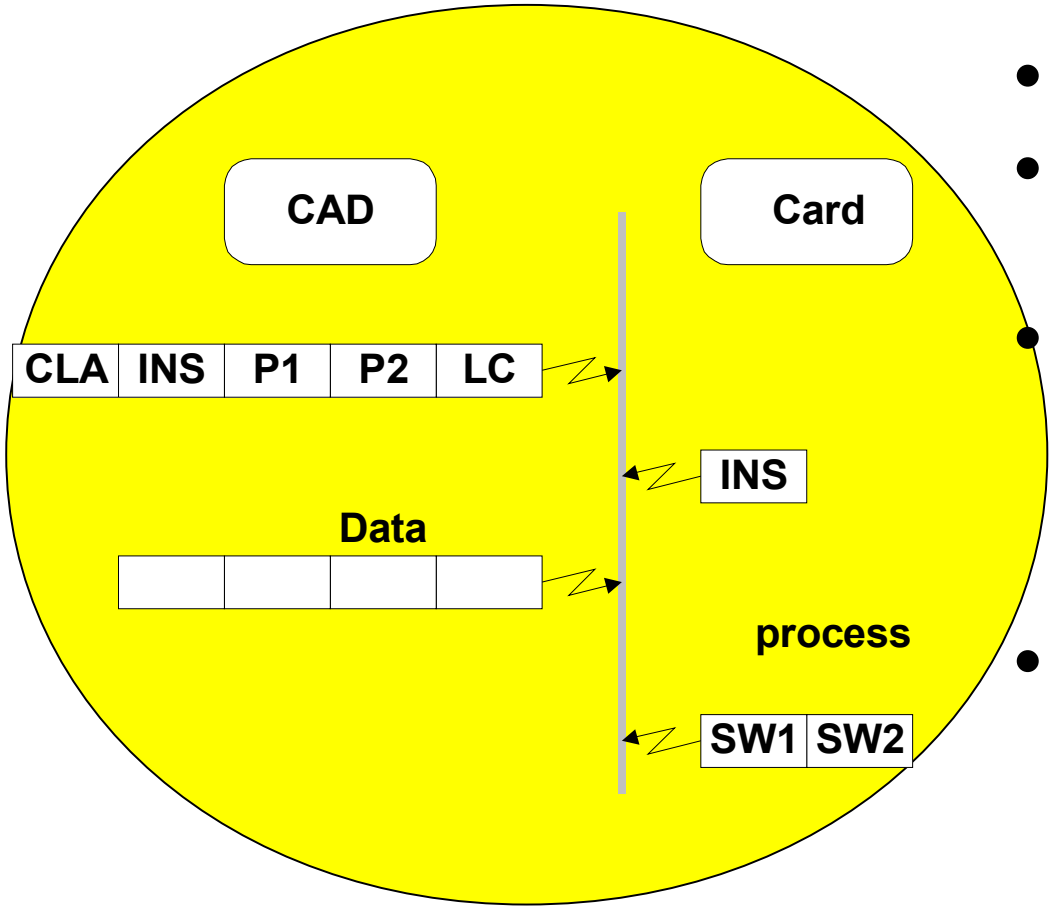


No parameters exchanged



- **LC** ==0
- The card receives the APDU
- It processes it
- It returns a status word
  - Two bytes

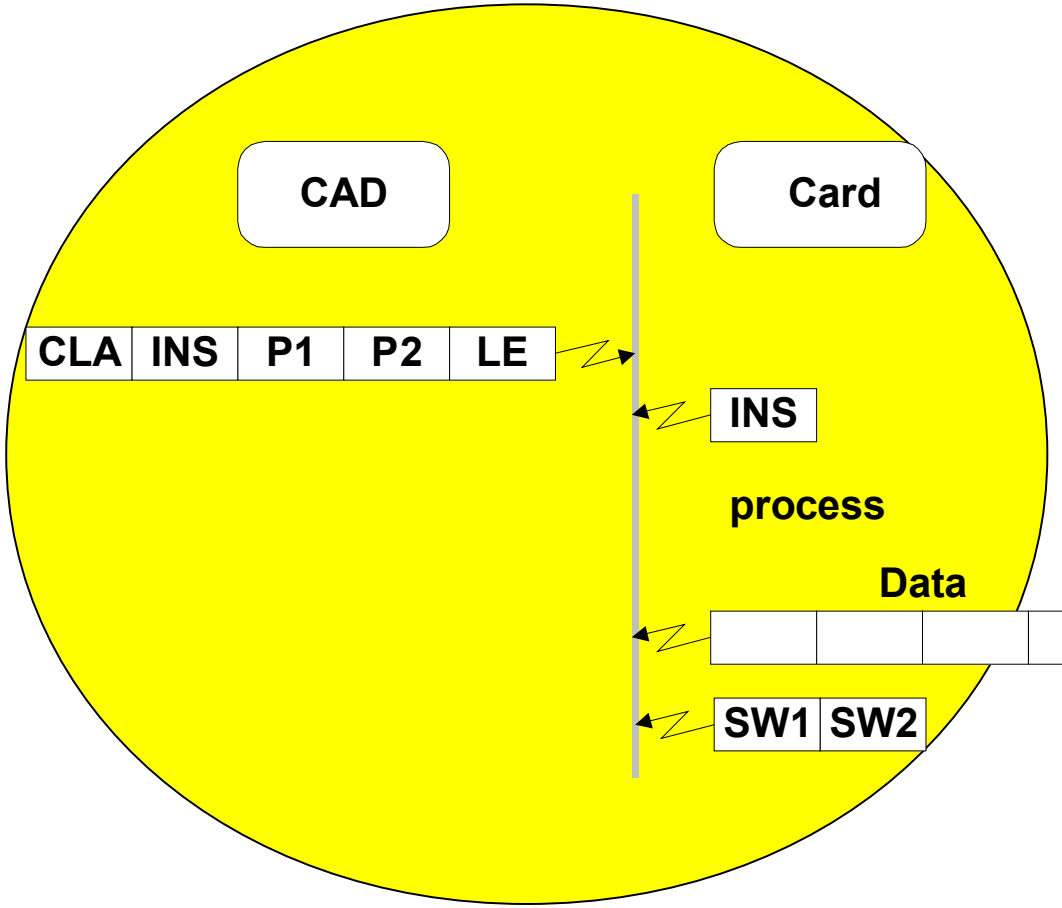
Parameters sent by the terminal



- **LC** ≠ 0
- LC indicates the length of the data in bytes
- The software in the terminal and the software in the card must agree on the direction of the exchange
- The card acknowledges by sending back the **INS** byte
  - Simple case

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Data expected by the terminal



- **LE** ≠ 0
  - The 5th byte is called LE in this case
- The card acknowledges the APDU by sending back the **INS** byte
  - Simple case
- Data are returned by the card, followed by the status word

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Status word

- Status report of the internal operation done by the card
- **0x9000** means success!
- When different, could indicate
  - Denied access
  - File not found
  - No such CLA or INS expected
  - …

Conclusion

- In this chapter, we have seen
  - An introduction to the ISO7816 standard
  - What is an APDU
  - How to exchange data between the CAD and the smart card

Java Card

Java Card Forum, history of the versions, programming aspects

Introduction

- In this chapter, we'll see
  - An introduction to the Java Card system
  - What is a Java Card Applet
  - What is the Java Card Runtime Environment
  - The lifecycle of an Applet
  - How to protect access with an OwnerPIN

Operating systems

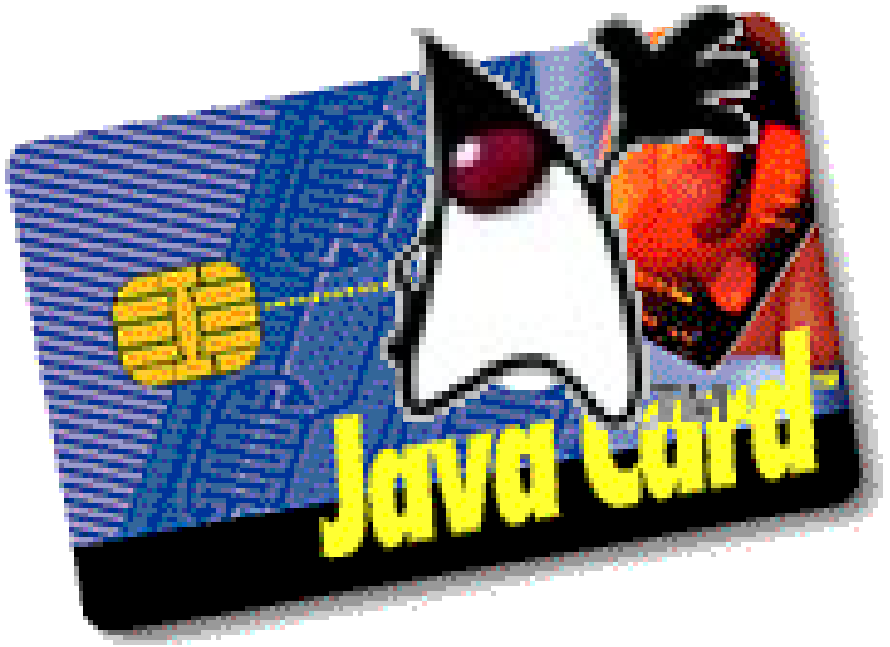
- Beginning: proprietary systems
  - Only the applications were standardized
    - B0' for French banking system
- Now: multi-application systems
  - MULTOS
  - Windows for Smart Card
    - Dead but replaced by .NET for smart cards
  - Java Card

Java Card History

- Early 1996
  - First development
    - Schlumberger, Bull CP8, GemPlus, Sun
  - Schlumberger's Cyberflex
  - Java Card Forum
    - Most of the smart cards manufacturers
    - Sun
      - As a Java guru

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Why Java in a smart card



- Java is an interpreted language
  - Need a Java Virtual Machine to run
- Applications could be portable from one smart card to another
- Applications run securely in a "sand box"
- Small footprint for the applications

Is Java for Java Card pure Java?

- No until Java Card 3.0!
- Roughly:
  - Basic types restricted to
    - Boolean
    - Small integers
      - Byte
      - Short
      - Int (optional)
    - No Strings
  - Arrays restricted to one-dimensional arrays
  - Limited libraries
    - Including java.lang
  - No garbage collector
- Less restrictions for Java Card 3.0

Which version in this course?

Related

Popular Downloads

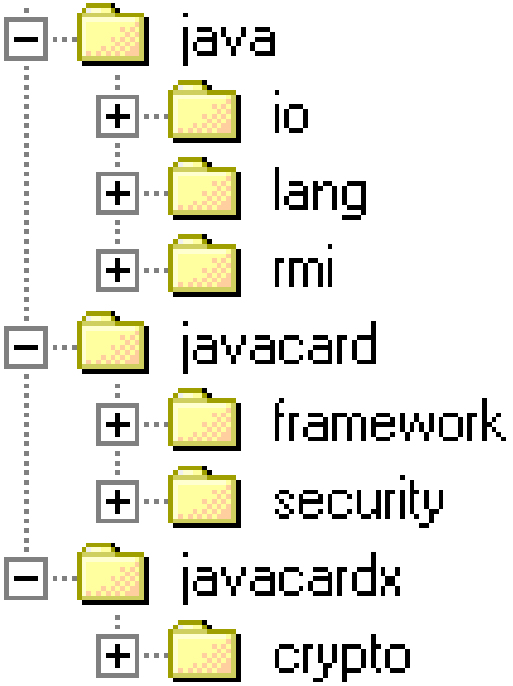
- Java Card Platform Specification 3.0.1
- Java Card Platform Specification 2.2.2
- Java Card Development Kit 2.2.2
- Java Card Protection Profile

Sun Resources

- Developer Technical Support
- Downloads for Application Development

- In this course we will introduce the Java Card 3
  - Classic Edition
    - Java Card 2.2.2
  - Connected Edition

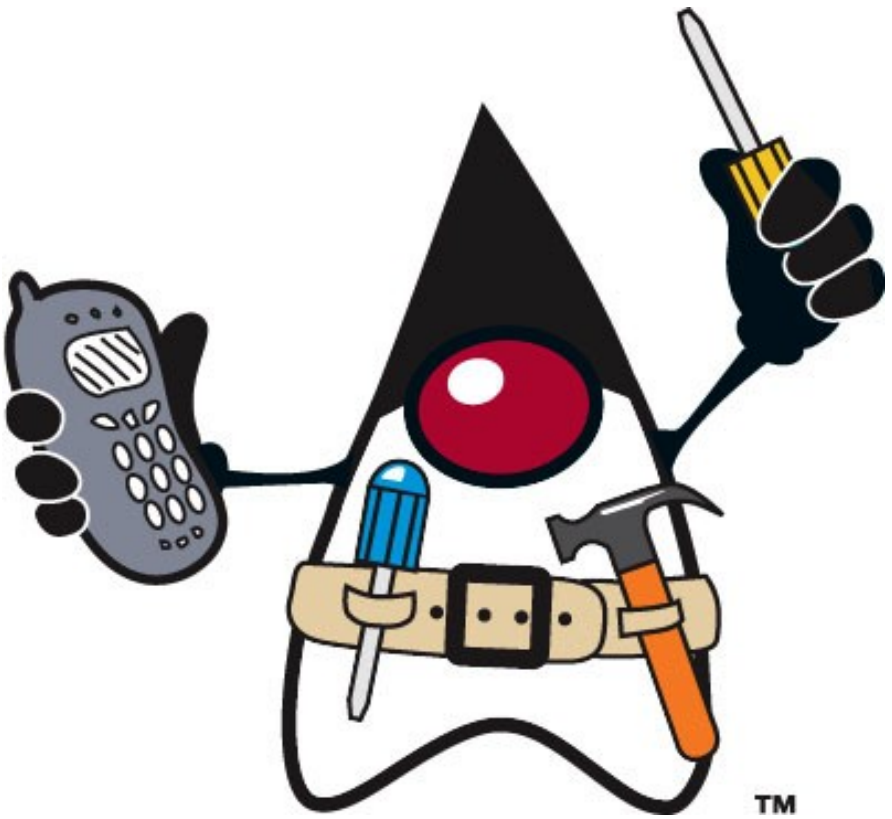
Available libraries



- Basically, **`javacard`** and **`javacardx`** contain the smart card API
  - **`framework`**, **`security`** and **`crypto`**
- **`java.lang`** is reduced mainly to the exception definitions
- **`java.io`** and **`java.rmi`** was introduced in the last 2.2 version
  - **`java.io`** to manage channels
  - **`java.rmi`** to manage remote method invocation

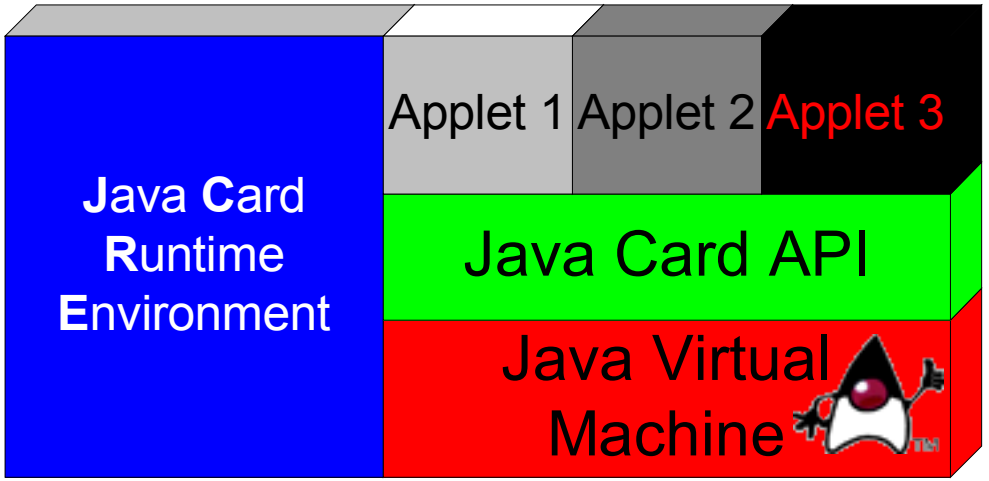
Smart University
State-of-the-Art ICT Knowledge and Training Workshops

SIM Toolkit



- For SIM Toolkit two more packages
  - access
  - toolkit
- Will be detailed later

How Java works in a smart card



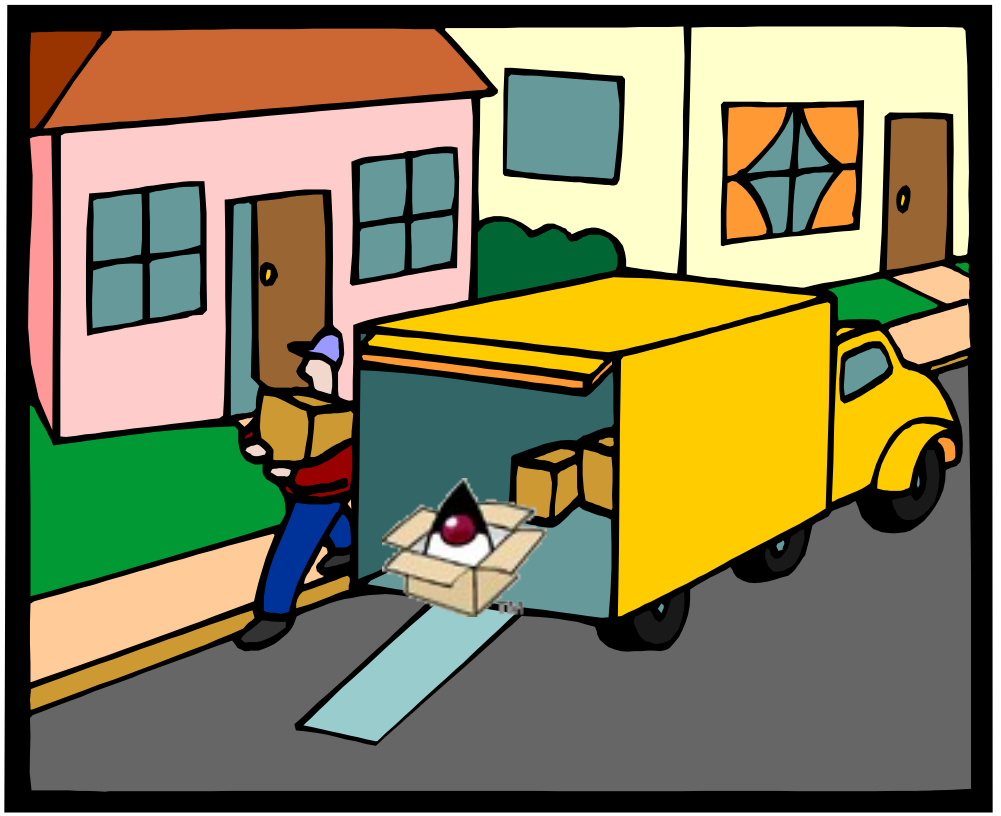
- A Java Virtual Machine is embedded
  - 4 K bytes
  - Basic library
- **J**ava **C**ard **R**untime **E**nvironment
  - In charge of
    - Activation of applications
    - Low level communication protocol
    - Application downloading

Roles of the JCRE

- Downloading a package
- Creating an instance of an applet
- Selecting an applet
- Transmitting an APDU to a selected applet
- Managing the communication protocol with the CAD

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Downloading a package



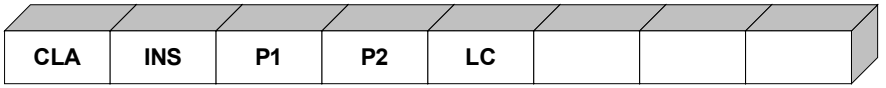
- Applets must be encapsulated in a package
- External processes
  - Compile the applets
  - Verify the bytecode
  - Create a jar-like container
    - CAP file
  - Will be seen later
- Package and applets are associated an identifier for future selection

What is a Java Card Applet

```
package ePurse;
import javacard.framework.*;
class EPurse extends Applet {
    short balance;
    public EPurse(){…}
    public static void install(…){…}
    public boolean select(){…}
    public void process(APDU apdu)
        {…}
}
```

- A java object which is
  - Running using the JVM
  - Controlled by the JCRE
- The class of this object must extend the class **`javacard.framework.Applet`**
- The class must overload several methods

Class APDU



- This class provides the basic features needed to handle the ISO7816 protocol from the applet point of view
- It gives access to the internal buffer dedicated to the communication
- This buffer can be
  - Retrieved by the applet
  - Filled up by the applet and sent to the CAD

Main methods of the APDU

```
byte buffer[] = apdu.getBuffer();


apdu.setIncomingAndReceive();


short le = apdu.setOutgoing();
apdu.setOutgoingLength(le);
apdu.sendBytes(ISO7816.OFFSET_CDATA,
                                    le);


apdu.setOutgoingAndSend(...);
```

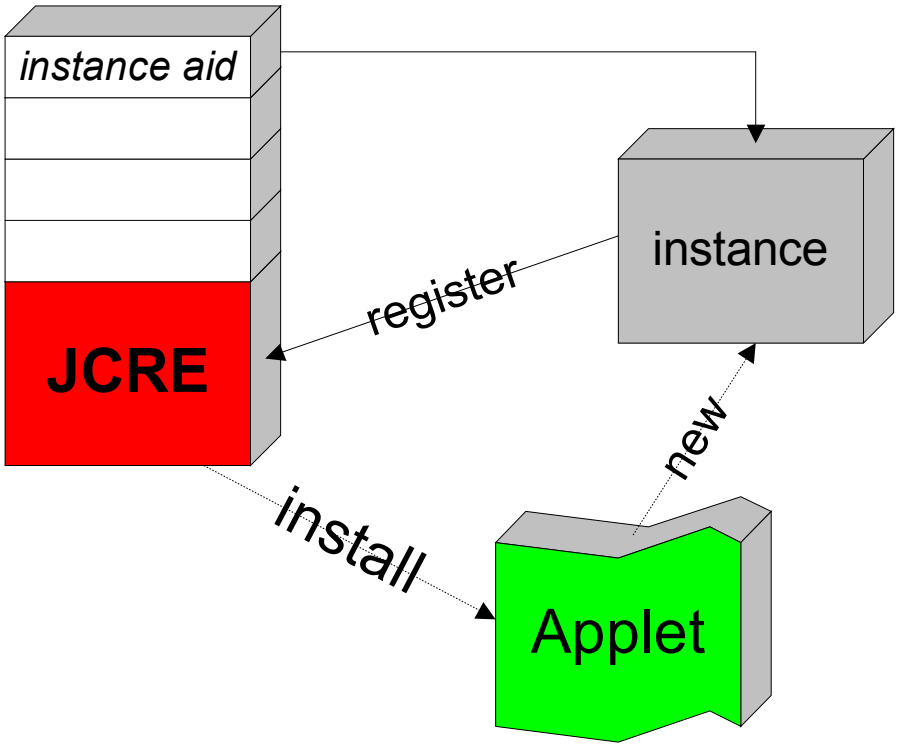
- These methods help to
  - Get the internal buffer
  - Start receiving data
    - Acknowledgement
  - Start transmitting data
- Utilities help to
  - Transform 2 bytes in a short and vice versa
  - Copy buffers
  - Compare buffers

Class ISO7816

Field Summary	
static byte	CLA_ISO7816 APDU command CLA : ISO 7816 = 0x00
static byte	INS_EXTERNAL_AUTHENTICATE APDU command INS : EXTERNAL AUTHENTICATE = 0x82
static byte	INS_SELECT APDU command INS : SELECT = 0xA4
static byte	OFFSET_CDATA APDU command data offset : CDATA = 5
static byte	OFFSET_CLA APDU header offset : CLA = 0
static byte	OFFSET_INS APDU header offset : INS = 1
static byte	OFFSET_LC APDU header offset : LC = 4
static byte	OFFSET_P1 APDU header offset : P1 = 2
static byte	OFFSET_P2 APDU header offset : P2 = 3

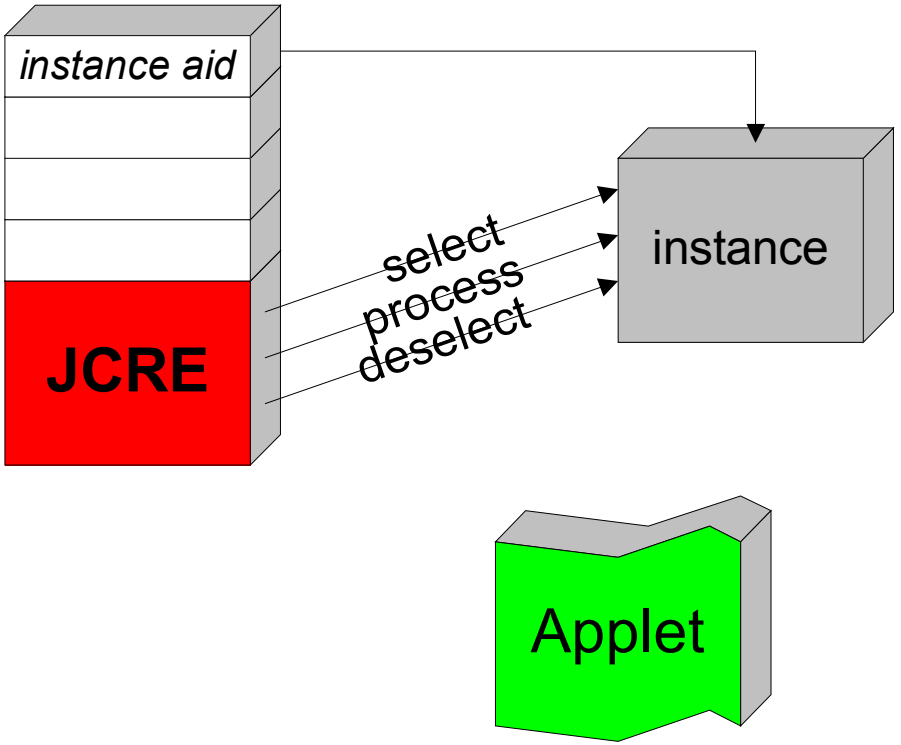
- This class encapsulates most of the ISO7816 constants needed to program the applets
- Constants are prefixed by
  - CLA for class related constants
  - INS for instruction related constants
  - OFFSET for offsets in the buffer
  - SW for status word related constants

Lifecycle of an applet



- The JCRE downloads the package containing the Applet
- It calls the static method **`install`** on the Applet
- This method creates an instance
  - Or more
- And **`register`** this instance using an **`AID`**

Lifecycle of an Applet



- When the instance is created and registered it can be called
- The JCRE can
  - **`select`**
  - **`deselect`**

  the instance
- Can call the instance to **`process`** an APDU

Example of an Applet

EPurse.java - Editor

EPurse.java

```
1   package ePurse;
2
3   import javacard.framework.*;
4
5   public class EPurse extends Applet {
6           public final static byte EPURSE_CLA = (byte)0x80;
7           public final static byte EPURSE_ADD = (byte)0xB0;
8           public final static byte EPURSE_SUB = (byte)0xB2;
9           public final static byte EPURSE_BAL = (byte)0xB4;
10
11          private short balance;
12
13      public static void install(byte[] bArray, short bOffset, byte bLength) {
14          new EPurse();
15      }
16
17      protected EPurse() {
18          balance = (short)0;
19          register();
20      }
```

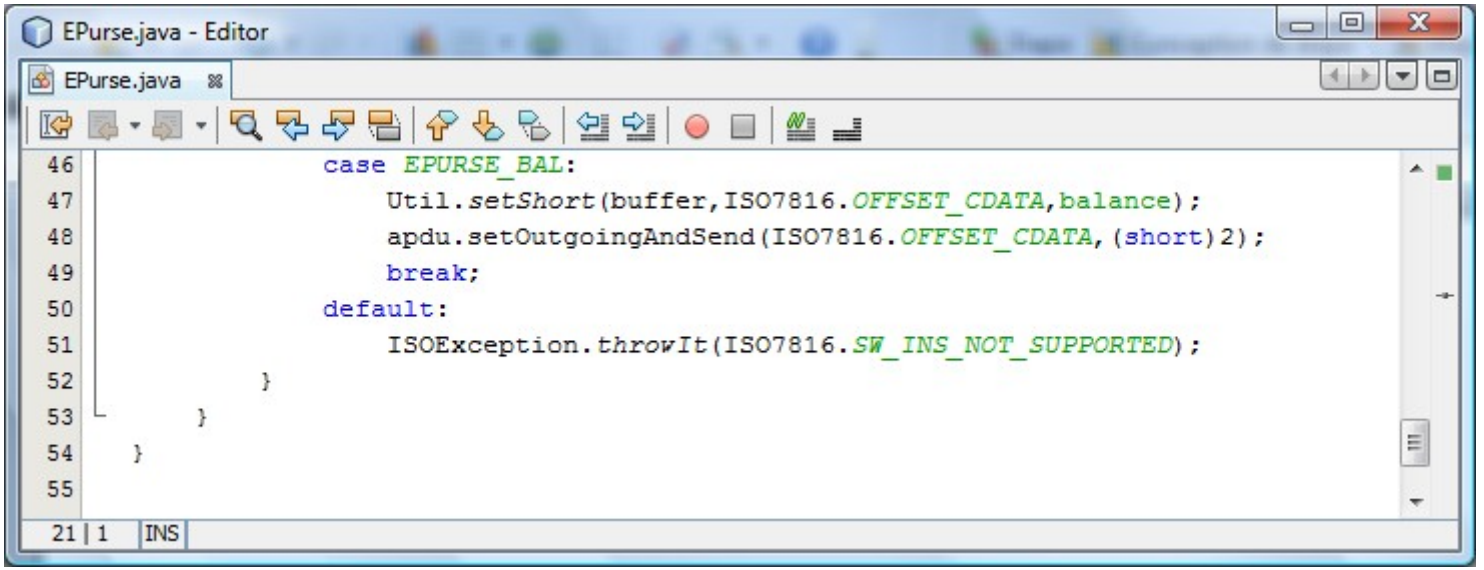
22 | 1 INS

EPurse.java - Editor

EPurse.java

```
    public void process(APDU apdu) {
23      byte []buffer = apdu.getBuffer();
24
25      if(selectingApplet())
26          return;
27
28      if(buffer[ISO7816.OFFSET_CLA] != EPURSE_CLA)
29              ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
30
31      short amount = (short)0;
32      switch(buffer[ISO7816.OFFSET_INS]){
33          case EPURSE_ADD:
34              apdu.setIncomingAndReceive();
35              amount = Util.getShort(buffer,ISO7816.OFFSET_CDATA);
36              balance += amount;
37              break;
38          case EPURSE_SUB:
39              apdu.setIncomingAndReceive();
40              amount = Util.getShort(buffer,ISO7816.OFFSET_CDATA);
41              if(amount<=balance)
42                      balance -= amount;
43              else
44                      ISOException.throwIt(ISO7816.SW_WRONG_DATA);
45              break;
```

21 | 1 INS


EPurse.java - Editor
EPurse.java
46 case EPURSE_BAL:
47 Util.setShort(buffer,ISO7816.OFFSET_CDATA,balance);
48 apdu.setOutgoingAndSend(ISO7816.OFFSET_CDATA,(short)2);
49 break;
50 default:
51 ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
52 }
53 }
54 }
55
21 | 1 INS

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Simulation script

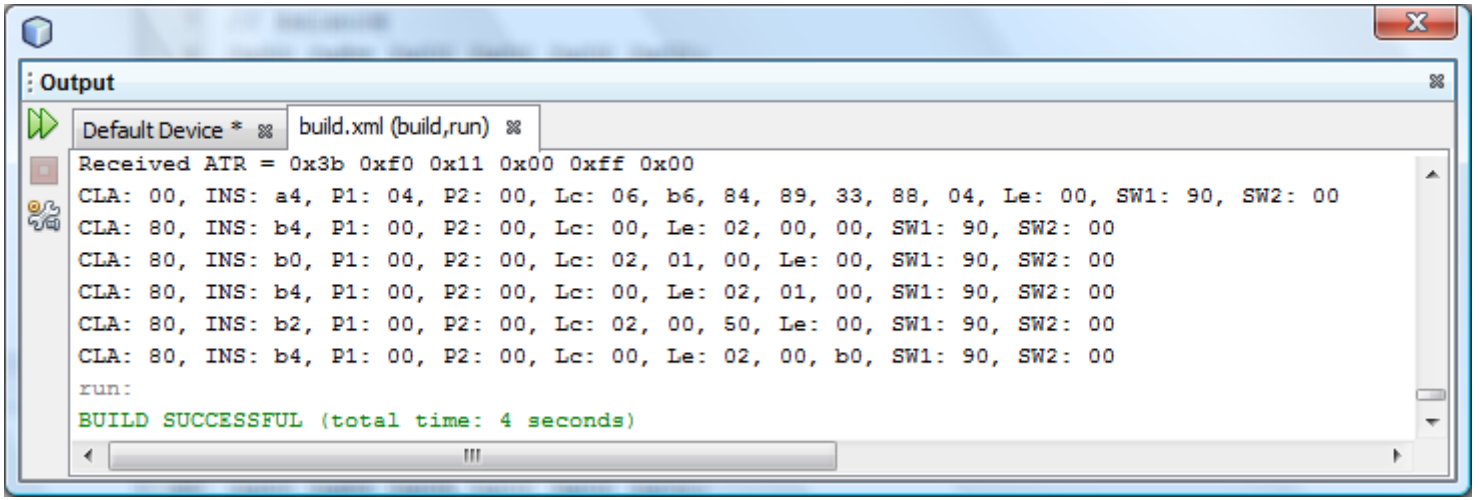
epurse.scr - Editor

epurse.scr

```
1  //Test script for Applet 'EPurse'
2
3  powerup;
4  // Select EPurse //aid/B684893388/03
5  0x00 0xA4 0x04 0x00 0X06 0XB6 0X84 0X89 0X33 0X88 0X04 0x7F;
6
7  // balance
8  0x80 0xB4 0x00 0x00 0x00 0x02;
9
10 // add 0x100
11 0x80 0xB0 0x00 0x00 0x02 0x01 0x00 0x7F;
12
13 // balance
14 0x80 0xB4 0x00 0x00 0x00 0x02;
15
16 // sub 0x50
17 0x80 0xB2 0x00 0x00 0x02 0x00 0x50 0x7F;
18
19 // balance
20 0x80 0xB4 0x00 0x00 0x00 0x02;
21
22 powerdown;
```

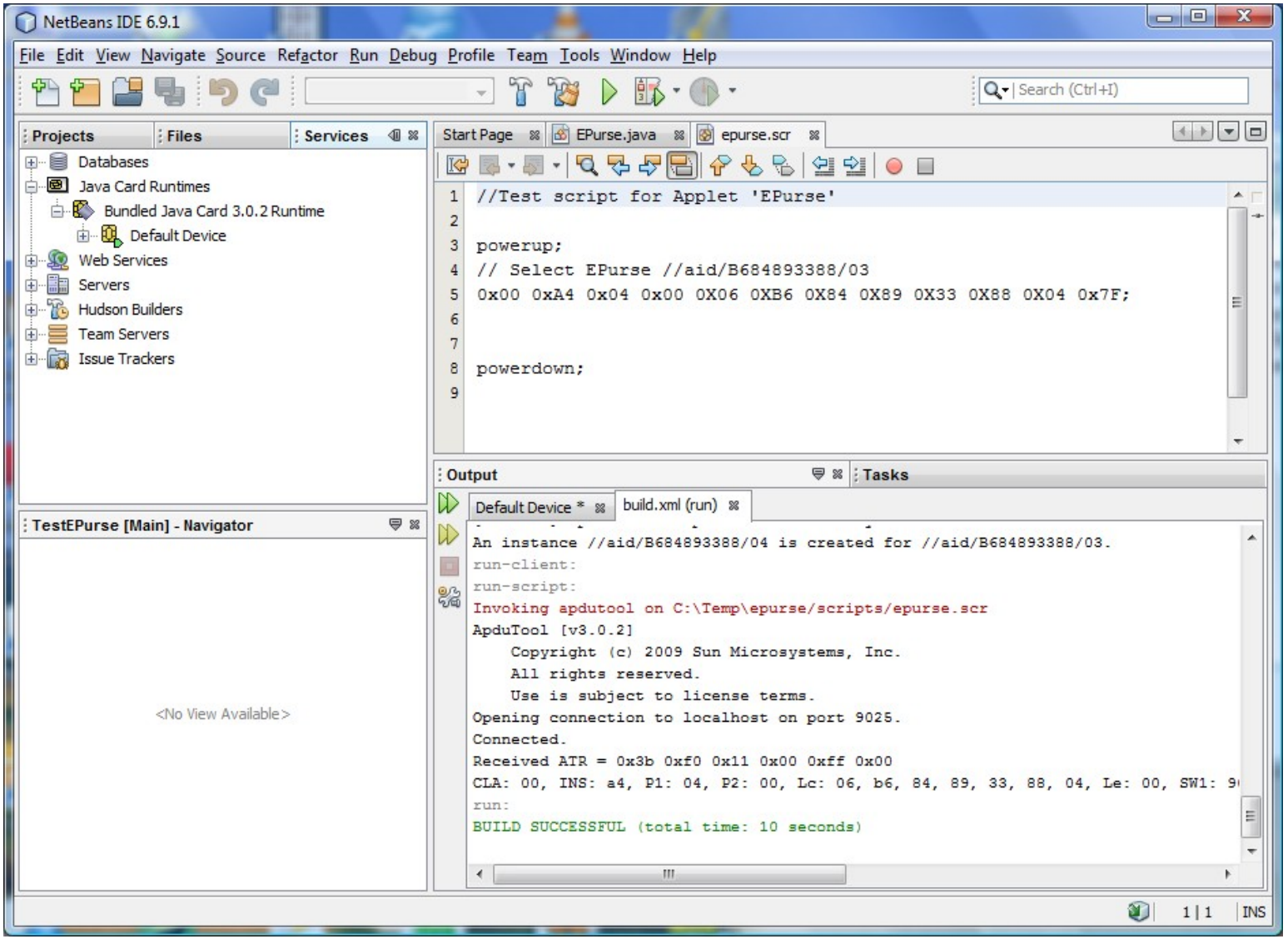
23 | 1 INS

Result

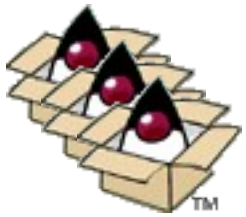

Output
Default Device *
build.xml (build,run)
Received ATR = 0x3b 0xf0 0x11 0x00 0xff 0x00
CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 06, b6, 84, 89, 33, 88, 04, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: b4, P1: 00, P2: 00, Lc: 00, Le: 02, 00, 00, SW1: 90, SW2: 00
CLA: 80, INS: b0, P1: 00, P2: 00, Lc: 02, 01, 00, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: b4, P1: 00, P2: 00, Lc: 00, Le: 02, 01, 00, SW1: 90, SW2: 00
CLA: 80, INS: b2, P1: 00, P2: 00, Lc: 02, 00, 50, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: b4, P1: 00, P2: 00, Lc: 00, Le: 02, 00, b0, SW1: 90, SW2: 00
run:
BUILD SUCCESSFUL (total time: 4 seconds)

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Netbeans 6.9



Other Java Card features



- Many features available
  - PIN code management
  - Transaction handling using **`JCSystem`**
    - Possibility to group together a certain number of actions into a transaction
    - Possibility to **`abort`** or **`commit`** the transaction
  - Shareable applets
  - Possibility to have several applets selected at the same time

OwnerPIN

- This class helps the developer to protect the access to some features of the smart card using a PIN code

```
private OwnerPIN pinCode;


    /** Creates a new instance of EPurse */
public EPurse() {
    balance = (short)0;
    pinCode = new OwnerPIN(EPURSE_PIN_TRY_LIMIT,
                    EPURSE_PIN_MAX_SIZE);
}
```

OwnerPIN

- The CAD must validate the PIN code prior to access the other features

```
case EPURSE_ADD:
    apdu.setIncomingAndReceive();
    if(!pinCode.isValidated())
        ISOException. throwIt(
            ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
    break;
case EPURSE_PIN:
    apdu.setIncomingAndReceive();
    if(!pinCode.check(buffer,
        ISO7816.OFFSET_CDATA, EPURSE_PIN_MAX_SIZE))
    ISOException.throwIt(
            ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
    break;
```

OwnerPIN

- The OwnerPIN proposes a method to unblock a blocked PIN code (after a TRY_LIMIT unsuccessful attempts)

```
case EPURSE_UNBLOCK:
    pinCode.resetAndUnblock();
```

OwnerPIN

- The OwnerPIN proposes a method to reset the `validated` flag

```
public boolean select(){
    pinCode.reset();
}
```

Conclusion

- In this chapter, we have seen
  - An introduction to the Java Card system
  - What is a Java Card Applet
  - What is the Java Card Runtime Environment
  - The lifecycle of an Applet
  - How to protect access with an OwnerPIN

TrUST Me

The key rules for Javacard Programming

Java Card Programming Issues

- Programming a Java Card seems simple
  - Reduced language
  - Reduced library
  - Most exciting features of Java available in Java Card
  - Most difficulties coming from the ISO7816 protocol hidden by the JCRE and the API

Java Card Programming Issues

- Powerful tools help developing applets
  - Basic toolkit available for free from Sun (Oracle)
    - Helps testing and debugging applets
  - Enhanced toolkits provided by most of the manufacturers to
    - Upload applets in target Java Cards
    - Test, on board, the uploaded applets

Java Card Programming Issues

- Most of the trainee's applets suffer from the following drawbacks:
  - No consistency in data when the card is teared suddenly from the reader
  - Poor usability and security
  - Time out and memory issues not taken in account

Tr U S T Me

- A Java Card applet must be
  - **Tr**ansaction aware
  - **U**sable
  - **S**ecure
  - **T**ime-out aware
  - **Me**mory aware

Transaction aware

- Context
  - Memorize the ten last operations for an e-purse
  - Operation is qualified by
    - The type
    - The amount
    - The date

Transaction aware (code example)

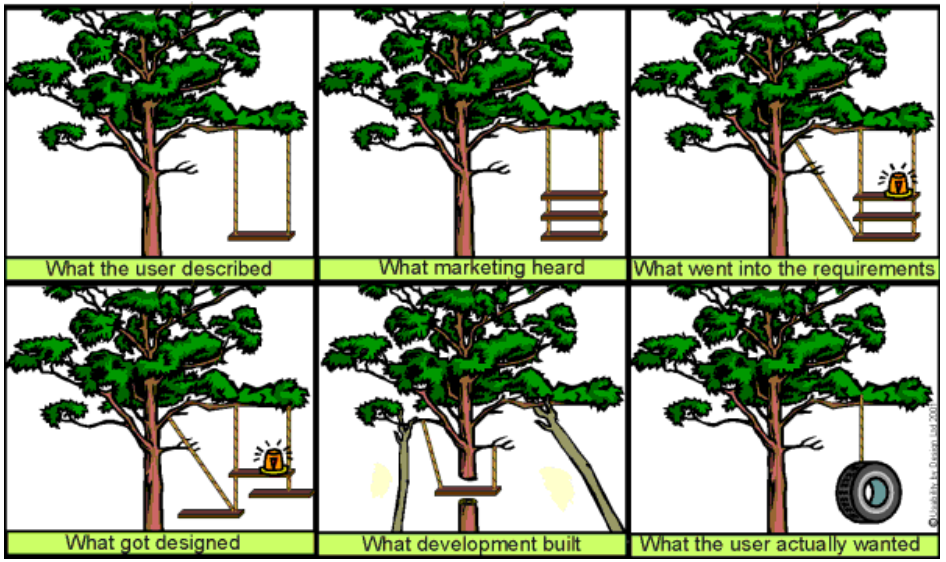
```
case EPURSE_ADD:
   apdu.setIncomingAndReceive();
   if(!pinCode.isValidated())
        ISOException.throwIt(
                ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
   amount = Util.getShort(buffer, ISO7816.OFFSET_CDATA);
   balance = (short)(balance + amount);
   list.add(buffer, ISO7816.OFFSET_INS,
                ISO7816.OFFSET_CDATA, (short)2,
                (short)(ISO7816.OFFSET_CDATA + (short)2),
                (short)8);
   break;
```

Transaction aware
(a better code)

```
…
try{
      JCSystem.beginTransaction();
      amount = Util.getShort(buffer, ISO7816.OFFSET_CDATA);
      balance = (short)(balance + amount);
      list.add(buffer, ISO7816.OFFSET_INS,
              ISO7816.OFFSET_CDATA, (short)2,
              (short)(ISO7816.OFFSET_CDATA + (short)2),
              (short)8);
      JCSystem.commitTransaction();
}catch(TransactionException ex){ … }
break;
```

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Usability



- Context
  - On an e-purse, each operation must be accepted only if the user's PIN code had been validated and if the operation is possible

Usability
(code example)

```
case EPURSE_ADD:
    apdu.setIncomingAndReceive();
    if(! pincode.check(buffer, ISO7816.OFFSET_CDATA, (byte)2))
          ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
    amount =
          Util.getShort(buffer, (short)(ISO7816.OFFSET_CDATA + (short)2));
    balance = (short)(balance + amount);
    list.add(buffer, ISO7816.OFFSET_INS,
                  (short)(ISO7816.OFFSET_CDATA + (short)2), (short)2,
                  (short)(ISO7816.OFFSET_CDATA + (short)4),
                  (short)8);
    break;
```

Usability

- PIN code must not be checked at each operation
  - But at each session starting
- PIN code must be deselected after the applet had been also deselected

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Security

- Context
  - Iris scan security system with the card holder's iris characteristics in a smart card
  - **Problem**:
    - Which part of the system must decide if the iris scanned corresponds to the data stored in the smart card:
      - The card acceptance device?
      - The Java Card?

Security
(proposed answers)

- Answer 1:
  - *The scanned data are passed to the smart card which returns yes or no!*
- Answer 2:
  - *The Card Acceptance Device get the stored data from the card to compare it with the scanned data*

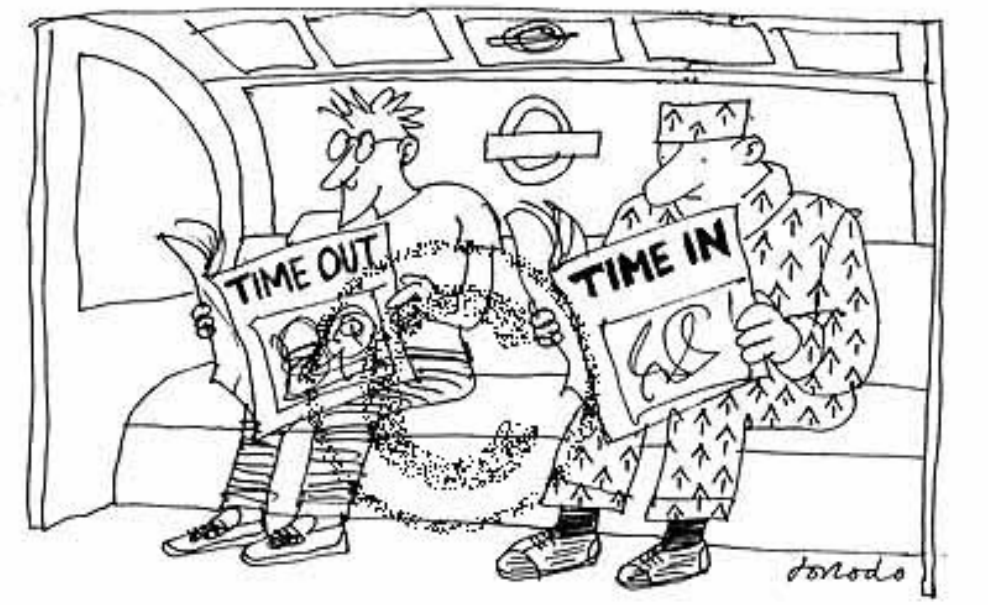
Security

- Mutual authentication is needed prior any data exchange
- Card Acceptance Device
  - which is more difficult to replace by a forged one

must make the comparison between data stored in the card and the data scanned

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Time out



- Context
  - A message is sent to the Java Card to be encrypted using a first command
  - A second command must be issued to get back the encrypted message

Time out
(time issues)

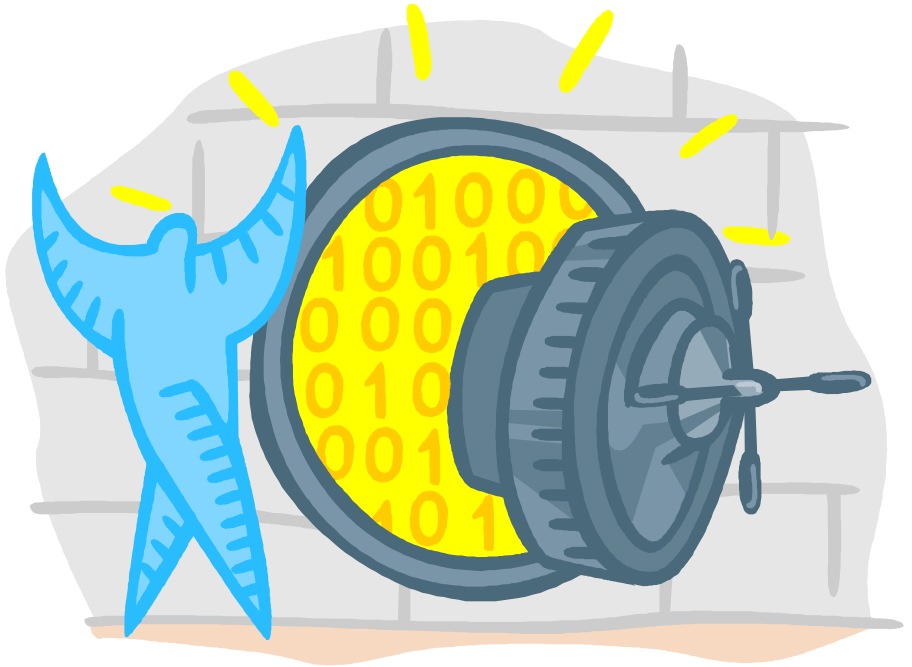
- What if
  - The Java card is teared from the card reader after the first command arrives and before the second command is issued
- Or if
  - The second command arrives before the first one is issued

Time out

- Time out aware applet
  - Must blank the message to be encrypted if deselect and/or select is called before the second command is issued
  - Must refuse the second command if the first was not sent before

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Memory aware



- Context
  - Memorize the ten last operations for an e-purse
  - Operation is qualified by
    - The type
    - The amount
    - The date

Memory aware
(code example)

```
case EPURSE_ADD:
   apdu.setIncomingAndReceive();
    amount = Util.getShort(buffer, ISO7816.OFFSET_CDATA);
    Operation op = new Operation(buffer,
        ISO7816.OFFSET_INS,ISO7816.OFFSET_CDATA, (short)2,
        (short)(ISO7816.OFFSET_CDATA + (short)2), (short)8);
    list.add(op);
    break;
```

Memory aware

- More Memory aware code
  - Avoid creating object on the fly
  - Create all the objects needed during construction phase
  - Recycle already created objects

Conclusion

- At the beginning, smart card programming was done:
  - In assembly language
  - At a low level
  - By engineers aware of the
    - **Tr**ansactions
    - **U**sability
    - **S**ecurity
    - **T**ime-out
    - **Me**mory usage

Conclusion

- Today, thanks to Java Card, applet programming can be done:
  - In Java
  - At a high level
  - By simple Java programmers

Conclusion

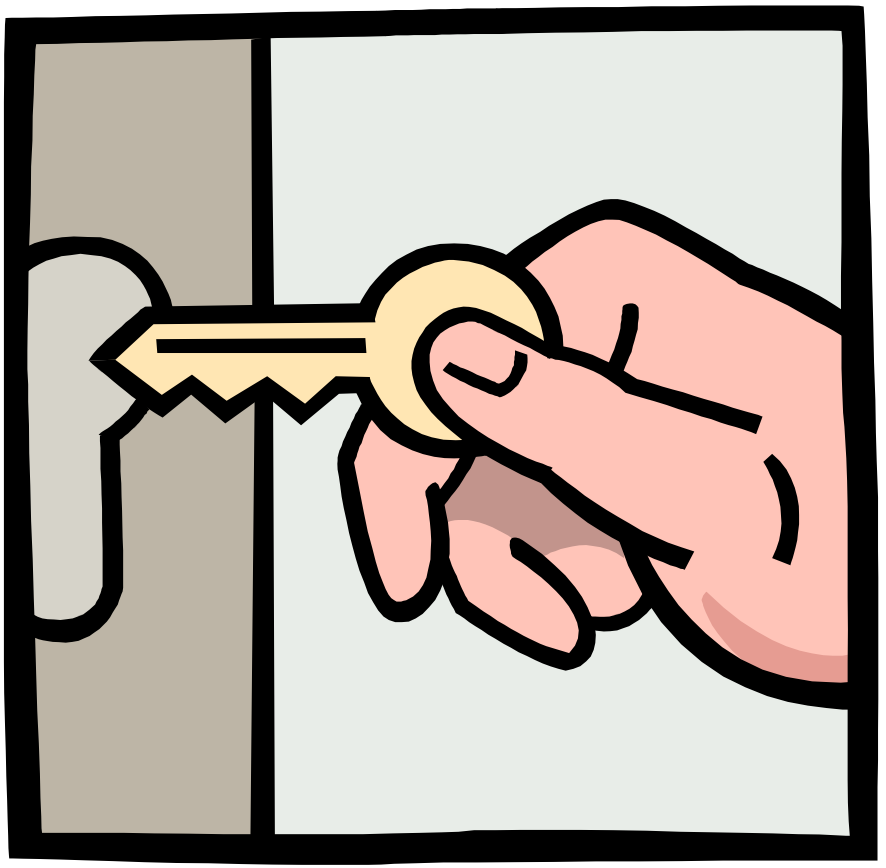
- The Java Card programmers must be aware of:
  - **Tr**ansactions
  - **U**sability
  - **S**ecurity
  - **T**ime out
  - **Me**mory usage

Security

Hardware and software aspects

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

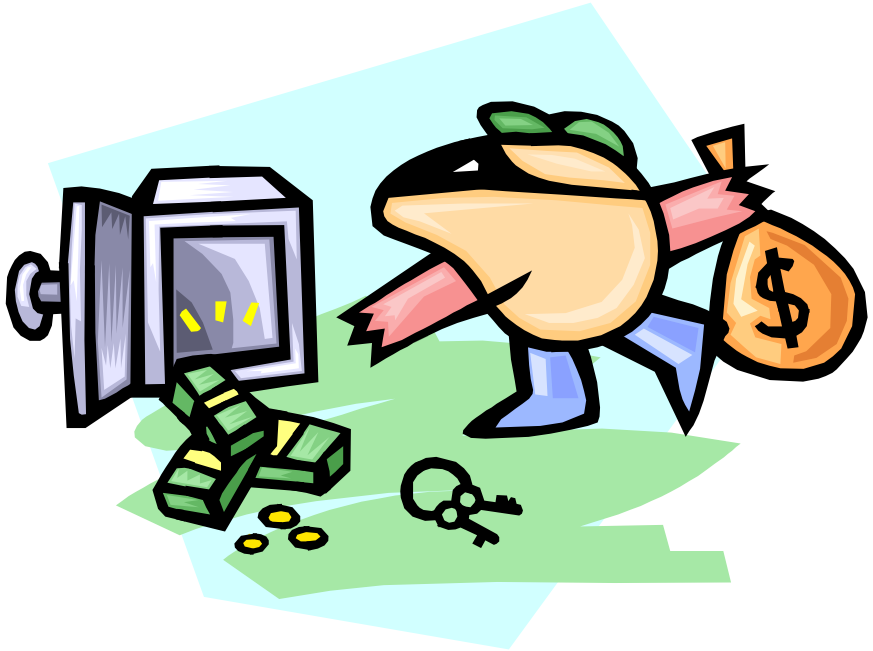
Objectives



- In this chapter, we'll see
  - An introduction about the security aspects of the smart cards
    - From a hardware point of view
    - From a software point of view

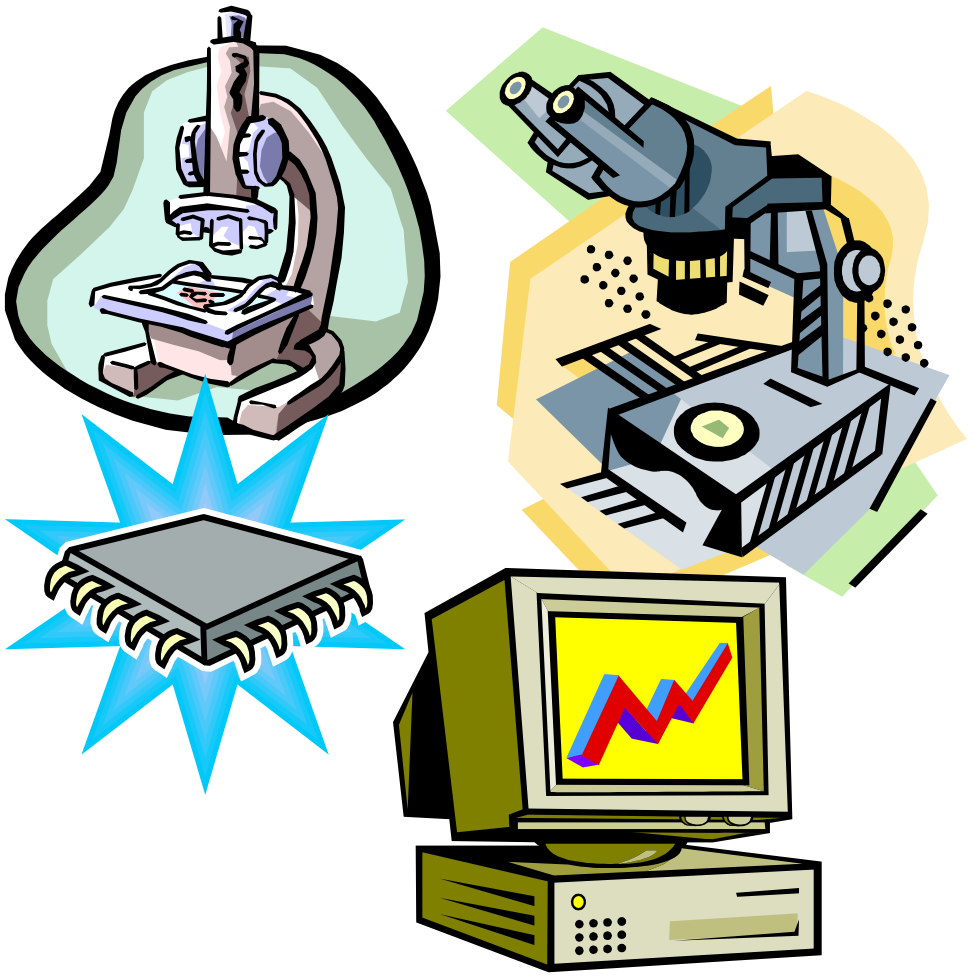
Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Hardware security



- A smart card contains important data
  - It could contain money
    - Electronic purses
- It must be tamper resistant
- "*If you know the attack you can build the shield*"

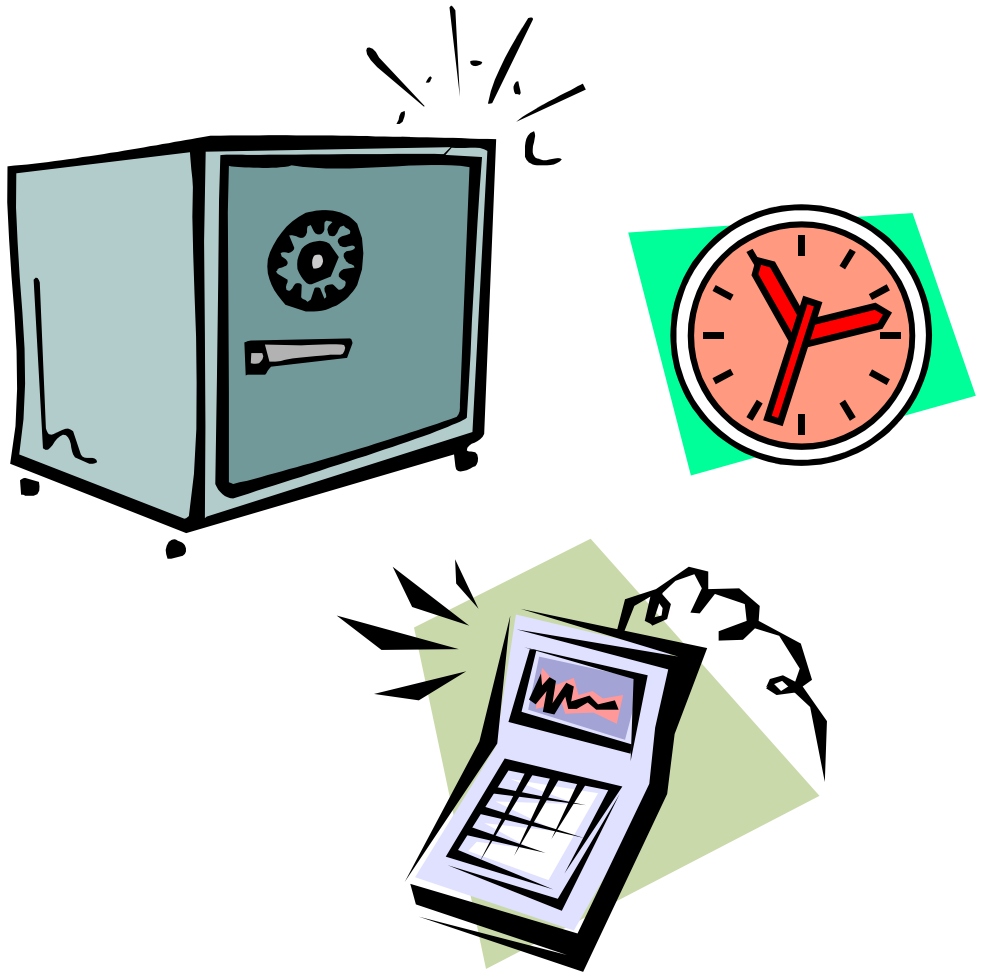
The attacks



- X raying the micro-chip
- Measuring the power consumption variation during critical APDU
  - When the PIN code is transmitted for example
- Measuring the answer delay
  - To try to predict what branches in the program are completed

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

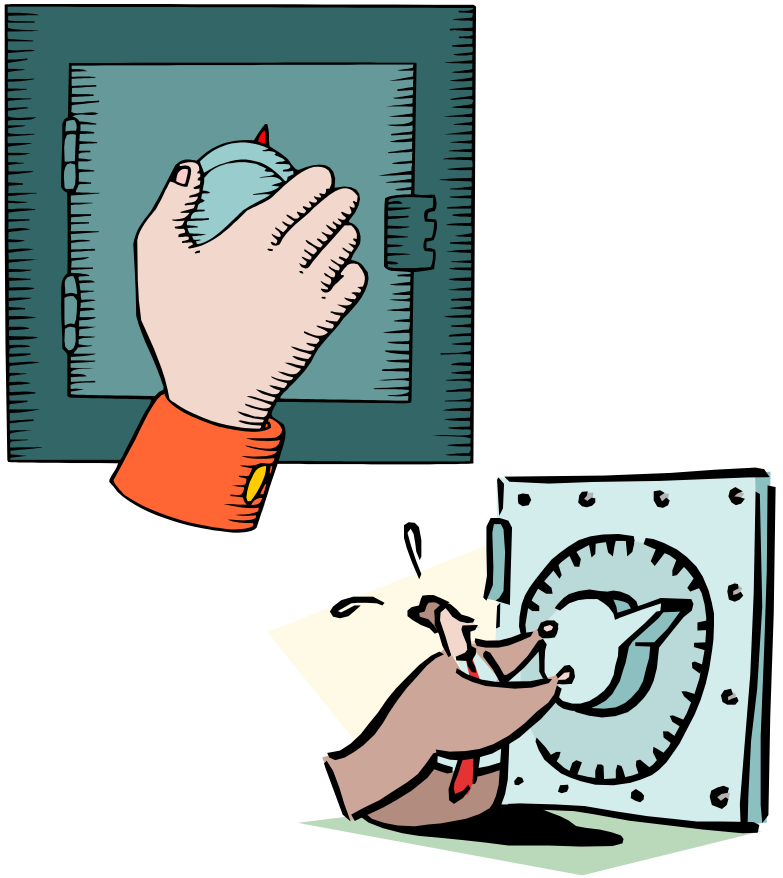
The shields



- The micro-chip uses an internal shield to protect itself against an X-Ray scanning
- It guarantees the same delay for both branches of an alternative statement
- It guarantees the same power consumption in all cases

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

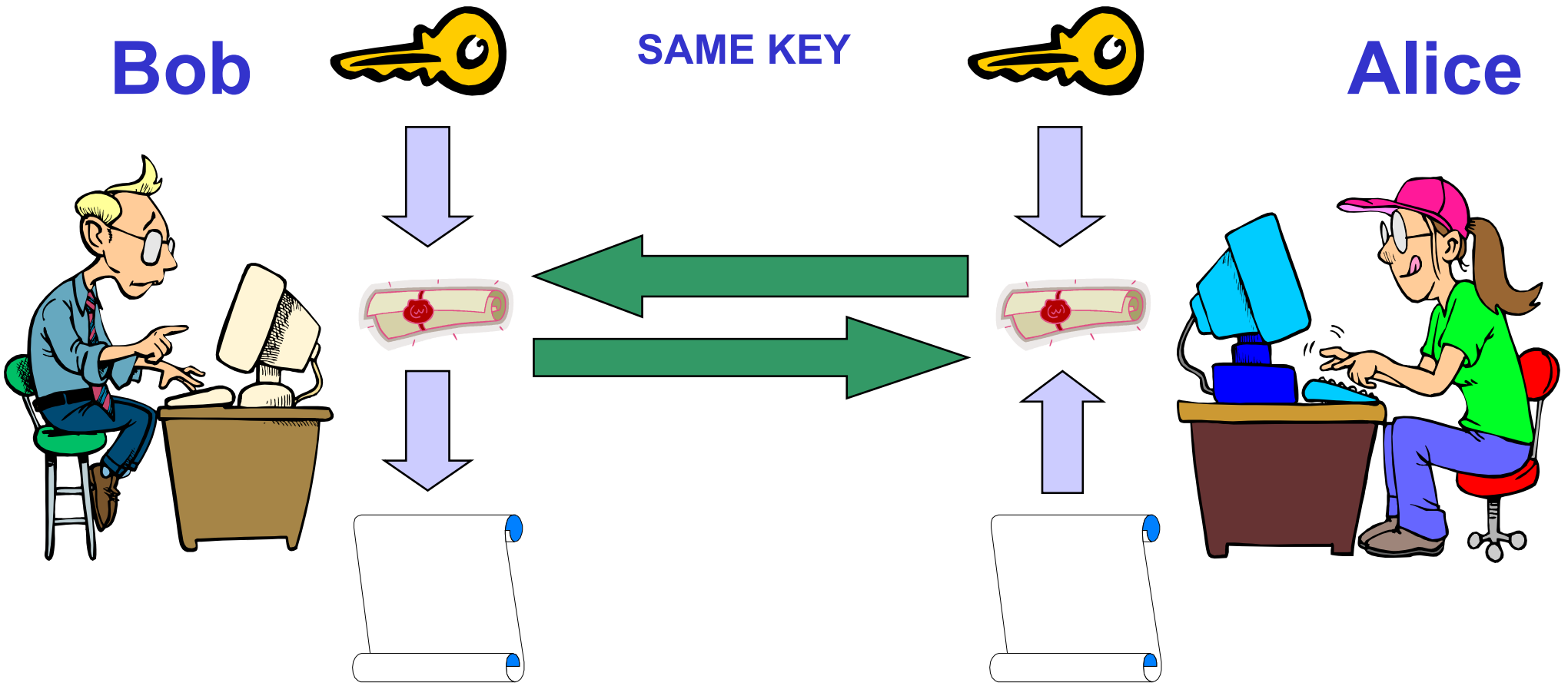
Software attacks and shields



- Data are protected using cryptography
  - Various techniques
    - DES, DES3, AES
    - RSA
    - SHA
- Cryptography is based on
  - A public algorithm
  - A key
    - Private (DES, DES3, AES)
    - Public (RSA)

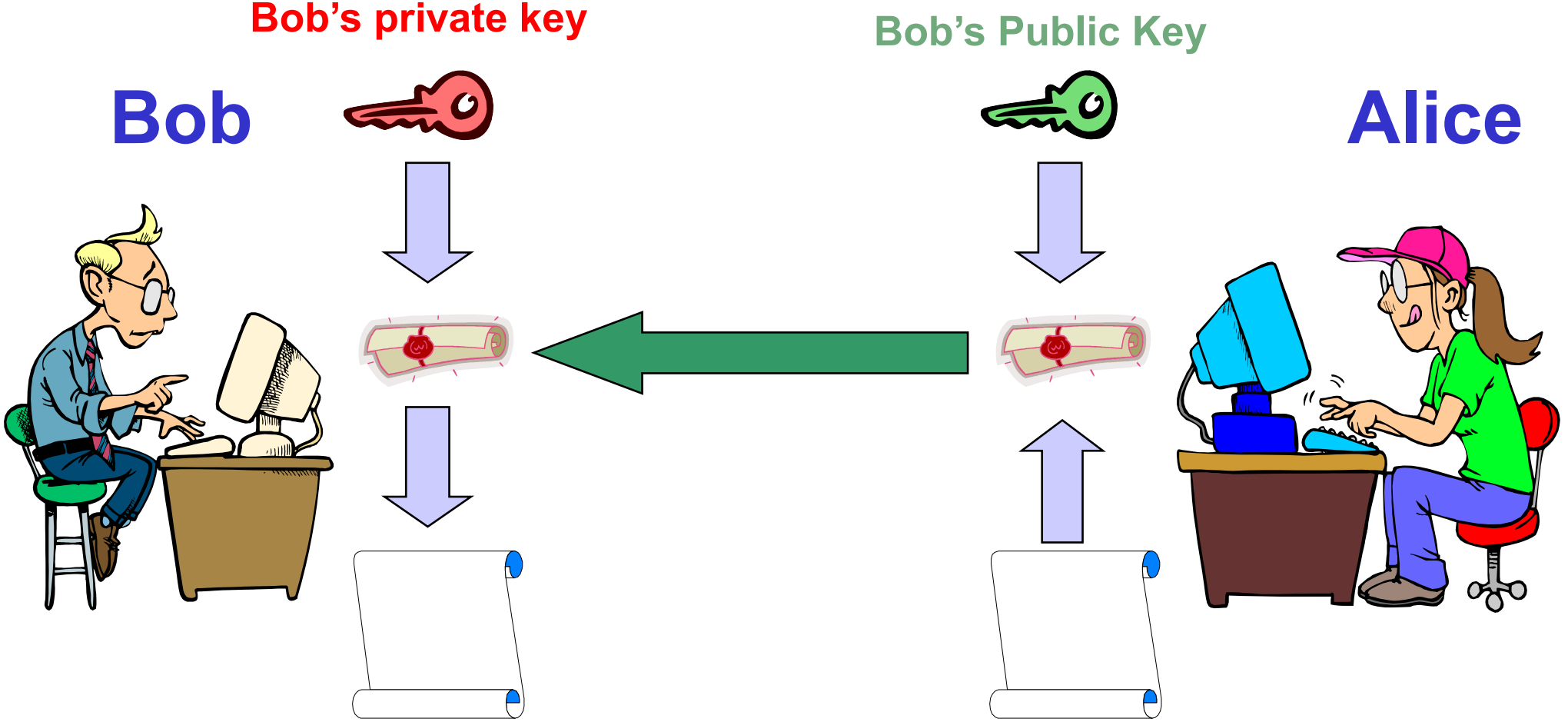
Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Symmetric Enciphering



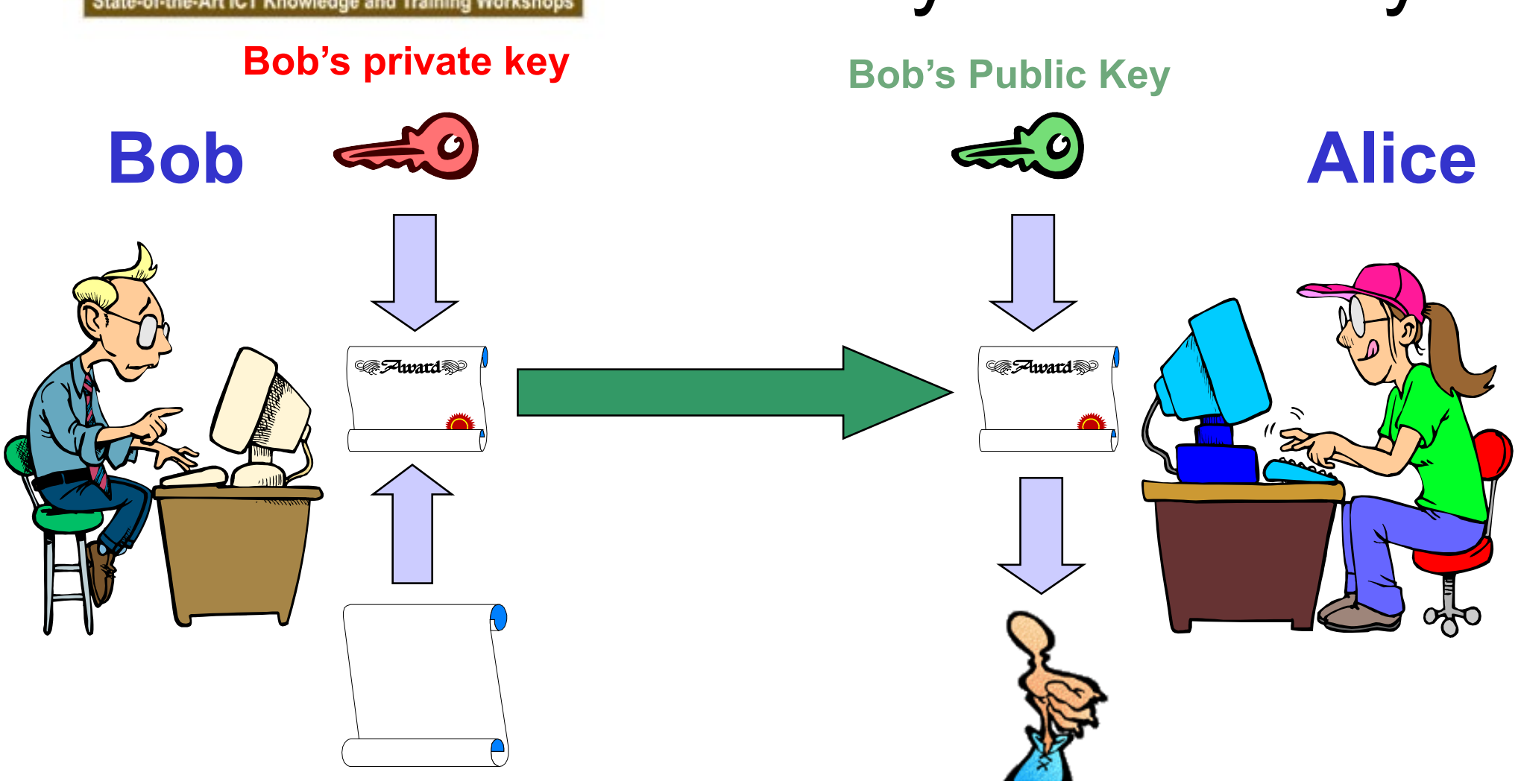
Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Asymmetric enciphering



Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Signing using asymmetric keys

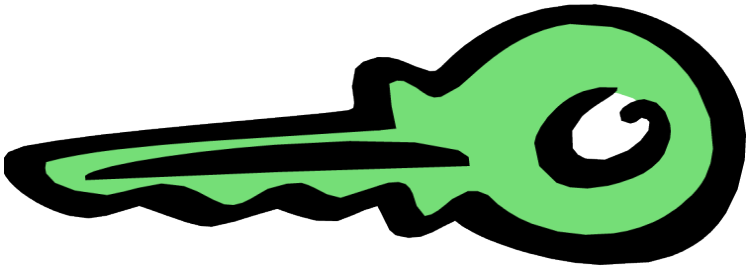


Certify public key

X509 Certificate

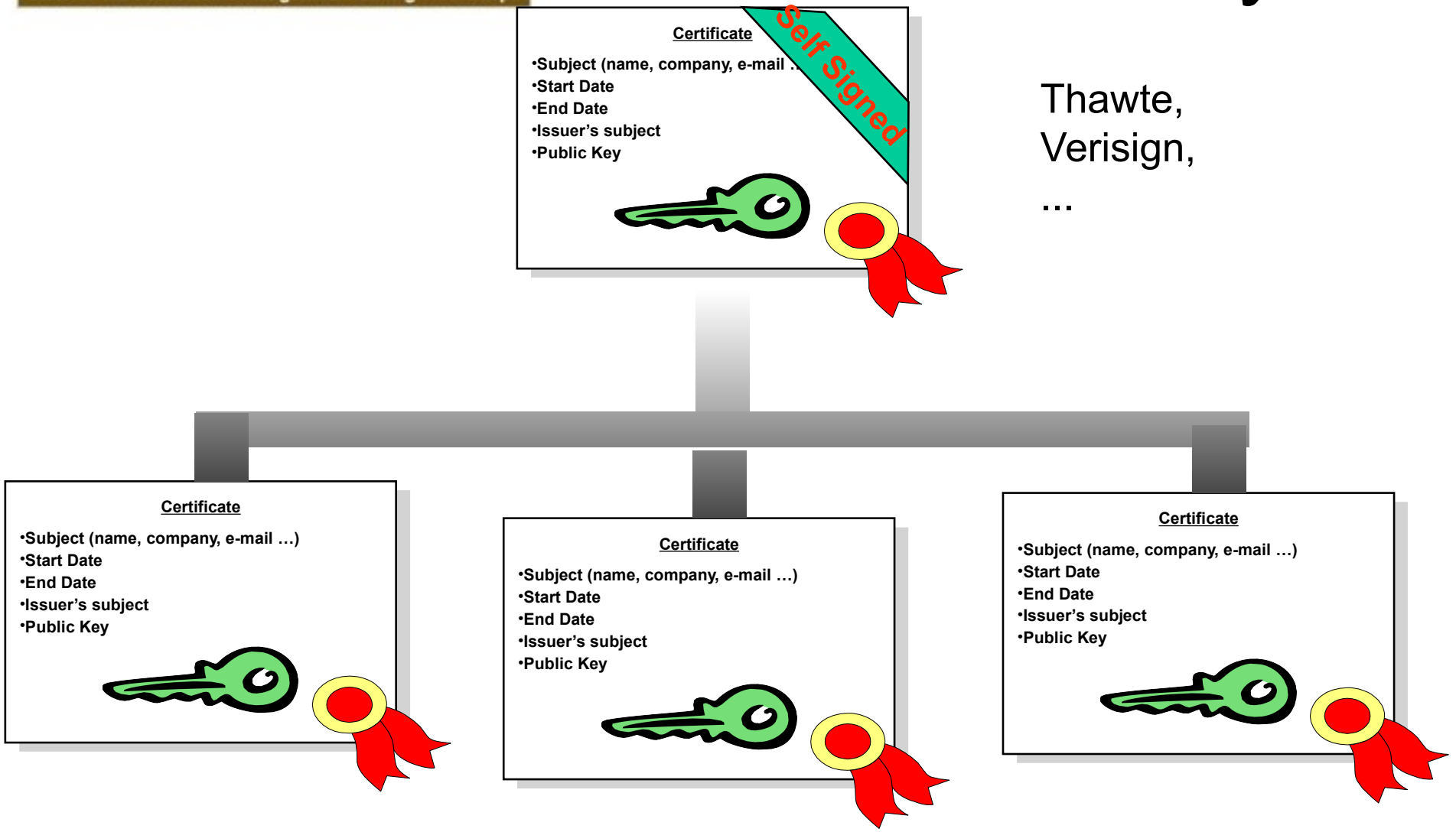
- Subject (name, company, e-mail …)

- Issuer's subject
- Public Key



Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Certification Authority



Thawte,
Verisign,
...

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Authentication

Authorization

Privacy

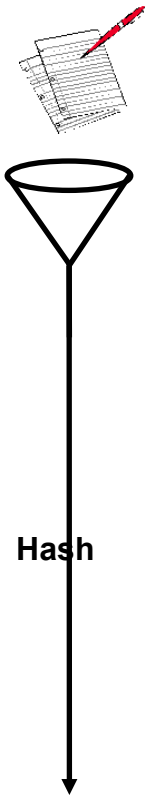
Integrity

Non-repudiation

Protect private key With Smart cards

- The Private key is born, lives and dies inside the card
  - Key pair generation
  - Secure access
  - Cryptographic algorithm process inside the card
- Physically secure
  - No hard drive storage of the private key
- Portable
  - No multi-key
  - Multiple Device
- Enciphering is done inside the card
  - Computer Independent

Hashing (a.k.a FingerPrint)

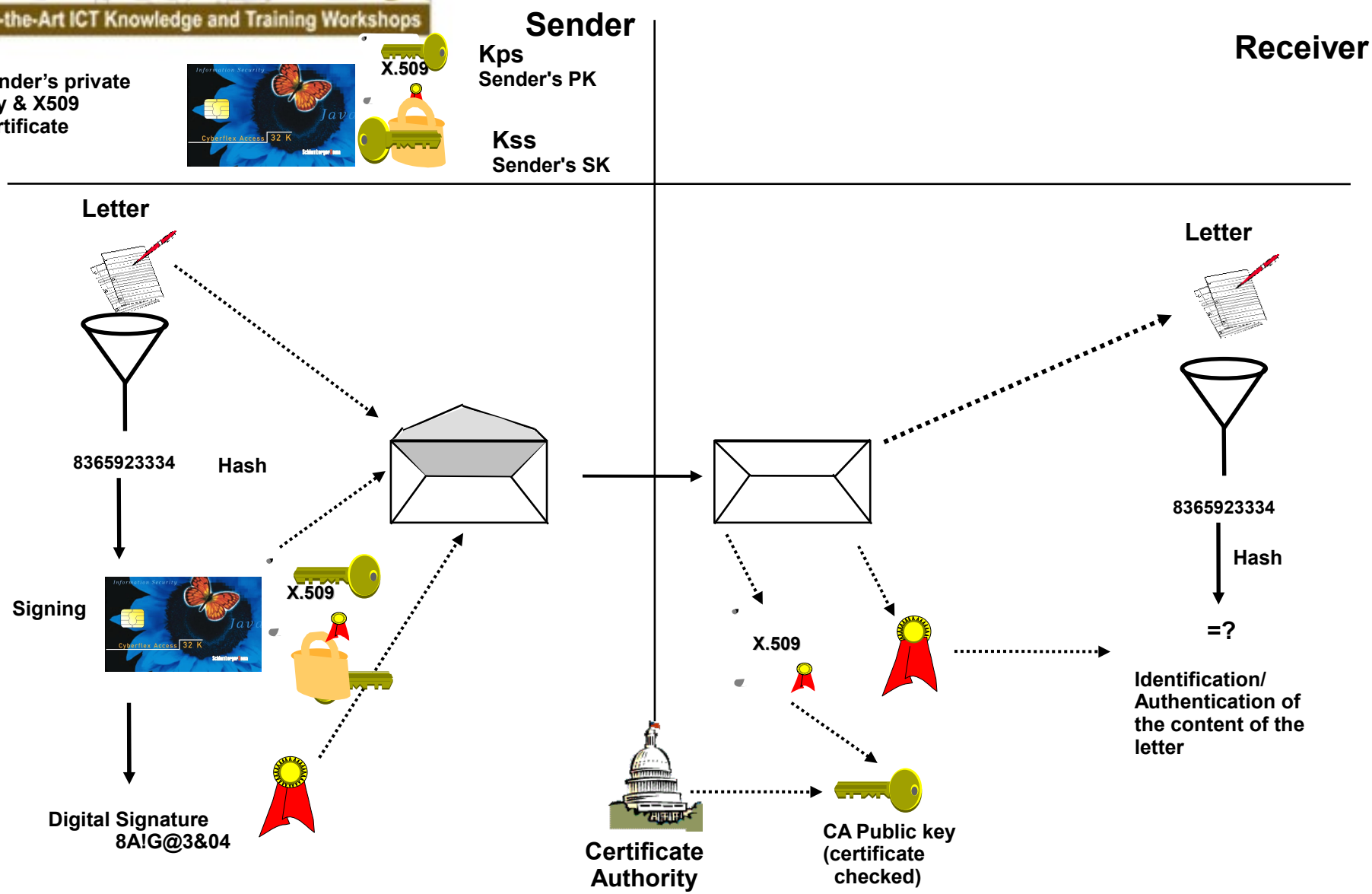


- Modifying one bit completely changes the Hash

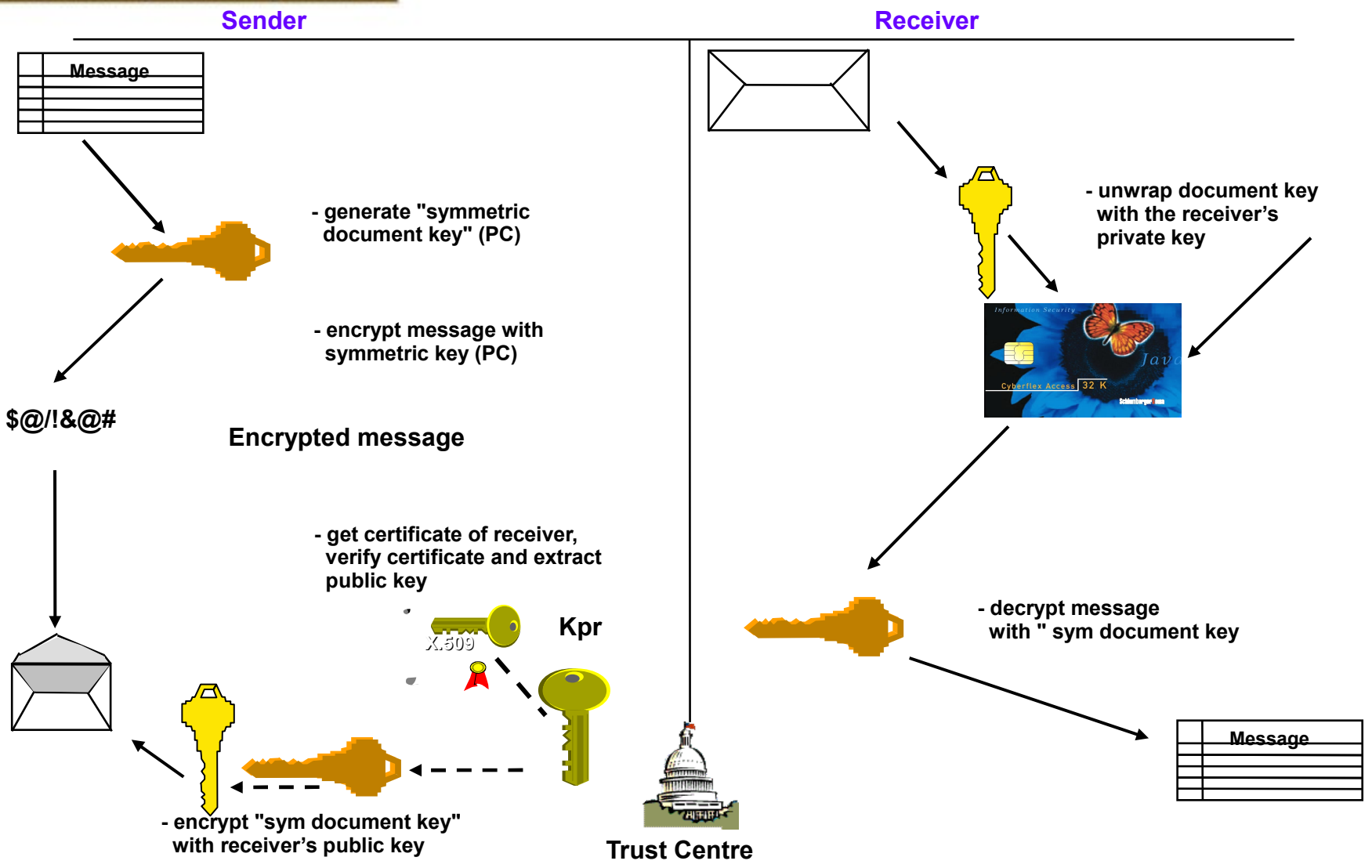
- Hash result is completely unpredictable

- Usual algorithms are MD5 (used for linux Password storage) or SHA-1

Digital Signature (Email)



S/MIME Encryption



Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example

KeyManager.java - Editor

KeyManager.java

```
1  package crypto;
2
3  import javacard.framework.*;
4  import javacard.security.*;
5  import javacardx.crypto.*;
6
7  /**
8   * @author Michel Koenig
9   */
10 public class KeyManager {
11
12     private byte [] bufferA = null, bufferB = null;
13     private short size = (short)0;
14     private RSAPublicKey publicKey=null;
15     private RSAPrivateKey privateKey=null;
16     private Cipher cip=null;
17
```

16 | 5 INS

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example

KeyManager.java - Editor

KeyManager.java

```
18      public KeyManager(short size) {
19            this.size = size;
20            bufferA = new byte[size];
21            bufferB = new byte[size];
22
23            KeyPair keyPair = null;
24            keyPair = new KeyPair(KeyPair.ALG_RSA,(short)(size * (short)8));
25            keyPair.genKeyPair();
26            publicKey = (RSAPublicKey)keyPair.getPublic();
27            privateKey = (RSAPrivateKey)keyPair.getPrivate();
28
29            cip = Cipher.getInstance(Cipher.ALG_RSA_PKCS1,false);
30       }
31
```

16 | 5 INS

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example

KeyManager.java - Editor

KeyManager.java

```
32      public short getPublicKey(byte buffer[], short offset) throws javacard.framework.ISOException {
33          short length = publicKey.getExponent(buffer,offset);
34          offset += length;
35          length += publicKey.getModulus(buffer,offset);
36          return length;
37      }
38
39      public short getModulusPrivateKey(byte buffer[], short offset) throws javacard.framework.ISOException {
40          short length = privateKey.getModulus(buffer,offset);
41          return length;
42      }
43
44      public short getExponentPrivateKey(byte buffer[], short offset) throws javacard.framework.ISOException {
45          short length = privateKey.getExponent(buffer,offset);
46          return length;
47      }
48
```

16 | 5 INS

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example

KeyManager.java - Editor

KeyManager.java

```
49    public short encryptPublic(byte [] buffer, short offset, short length){
50        cip.init(publicKey,Cipher.MODE_ENCRYPT);
51        Util.arrayCopy(buffer, offset, bufferA, (short)0,length);
52        length = cip.doFinal(bufferA, (short)0, length, bufferB, (short)0);
53        Util.arrayCopy(bufferB, (short)0, buffer, offset, length);
54        return length;
55    }
56
57    public short decryptPublic(byte [] buffer, short offset, short length){
58        cip.init(publicKey,Cipher.MODE_DECRYPT);
59        Util.arrayCopy(buffer, offset, bufferA, (short)0,length);
60        length = cip.doFinal(bufferA, (short)0, length, bufferB, (short)0);
61        Util.arrayCopy(bufferB, (short)0, buffer, offset, length);
62        return length;
63    }
```

80 | 1 INS

Example

KeyManager.java - Editor

KeyManager.java

```
65    public short encryptPrivate(byte [] buffer, short offset, short length){
66        cip.init(privateKey,Cipher.MODE_ENCRYPT);
67        Util.arrayCopy(buffer, offset, bufferA, (short)0,length);
68        length = cip.doFinal(bufferA, (short)0, length, bufferB, (short)0);
69        Util.arrayCopy(bufferB, (short)0, buffer, offset, length);
70        return length;
71    }
72
73    public short decryptPrivate(byte [] buffer, short offset, short length){
74        cip.init(privateKey,Cipher.MODE_DECRYPT);
75        Util.arrayCopy(buffer, offset, bufferA, (short)0,length);
76        length = cip.doFinal(bufferA, (short)0, length, bufferB, (short)0);
77        Util.arrayCopy(bufferB, (short)0, buffer, offset, length);
78        return length;
79    }
```

57 | 44 INS

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example

Crypto.java - Editor

Crypto.java

```
1   package crypto;
2
3   import javacard.framework.*;
4
5   /**
6    * @author Michel Koenig
7    */
8   public class Crypto extends Applet {
9           private static final byte KMT_CLA = (byte)0x80;
10
11          private static final byte KMT_GET_PUBLIC  = (byte)0xB0;
12          private static final byte KMT_ENC_PUBLIC  = (byte)0xB2;
13          private static final byte KMT_DEC_PUBLIC  = (byte)0xB4;
14          private static final byte KMT_ENC_PRIVATE = (byte)0xB6;
15          private static final byte KMT_DEC_PRIVATE = (byte)0xB8;
16
17          private KeyManager keyManager = null;
18
19      public static void install(byte[] bArray, short bOffset, byte bLength) {
20          new Crypto();
21      }
```

23 | 17 INS

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example

Crypto.java - Editor

Crypto.java

```
23      protected Crypto() {
24          keyManager = new KeyManager((short)64);
25          register();
26      }
27
        public void process(APDU apdu) {
29          byte []buffer = apdu.getBuffer();
30          short length = (short)0;
31
32          if(selectingApplet())
33              return;
34
35          if(buffer[ISO7816.OFFSET_CLA] != KMT_CLA)
36                  ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
37
```

34 | 1 INS

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example

Crypto.java - Editor

Crypto.java

```
38          try{
39              switch(buffer[ISO7816.OFFSET_INS]){
40                  case KMT_GET_PUBLIC:
41                          length = keyManager.getPublicKey(buffer, (short) 0);
42                          apdu.setOutgoing();
43                          apdu.setOutgoingLength(length);
44                          apdu.sendBytesLong(buffer,(short)0,length);
45                          break;
46
47                  case KMT_ENC_PUBLIC:
48                          length = apdu.setIncomingAndReceive();
49                          length = keyManager.encryptPublic(buffer, ISO7816.OFFSET_CDATA, length);
50                          apdu.setOutgoing();
51                          apdu.setOutgoingLength(length);
52                          apdu.sendBytesLong(buffer,ISO7816.OFFSET_CDATA,length);
53                          break;
54
```

34 | 1 INS

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example

Crypto.java - Editor

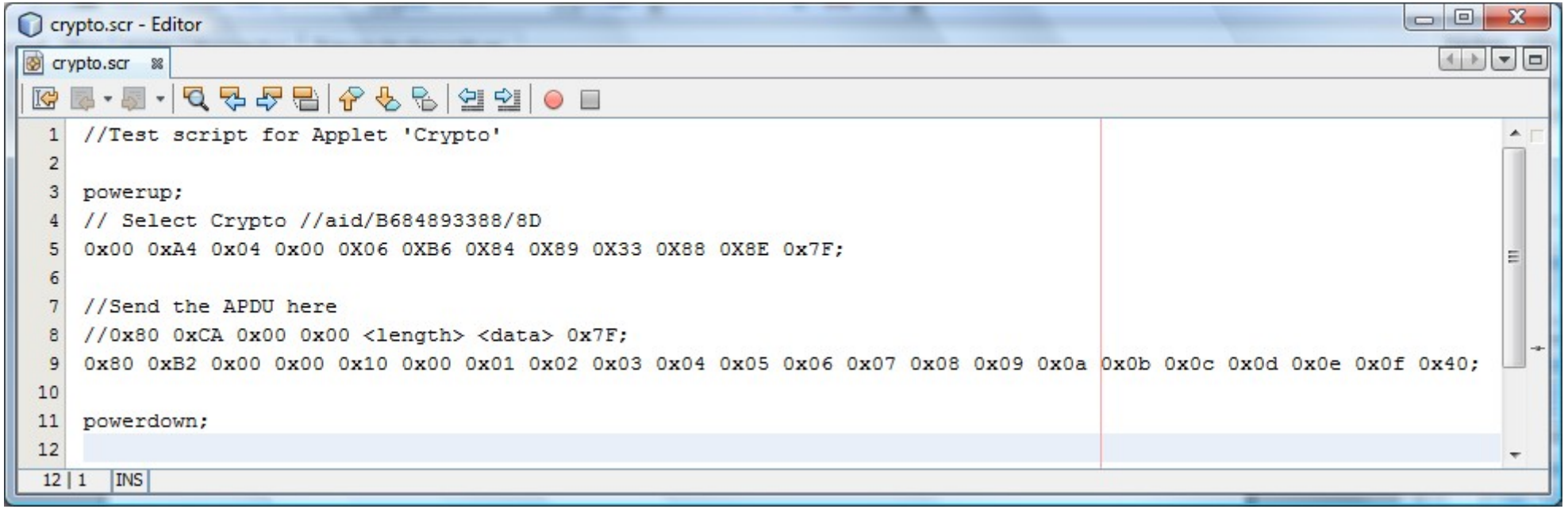
Crypto.java

```
61                      case KMT_DEC_PRIVATE:
62                              length = apdu.setIncomingAndReceive();
63                              length = keyManager.decryptPrivate(buffer, ISO7816.OFFSET_CDATA, length);
64                              apdu.setOutgoing();
65                              apdu.setOutgoingLength(length);
66                              apdu.sendBytesLong(buffer,ISO7816.OFFSET_CDATA,length);
67                              break;
68
69                      default:
70                              ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
71                  }
72              }catch(ISOException e){
73                      ISOException.throwIt(ISO7816.SW_DATA_INVALID);
74              }
75          }
76      }
77
```

34 | 1 INS

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Encrypting w/public



Result

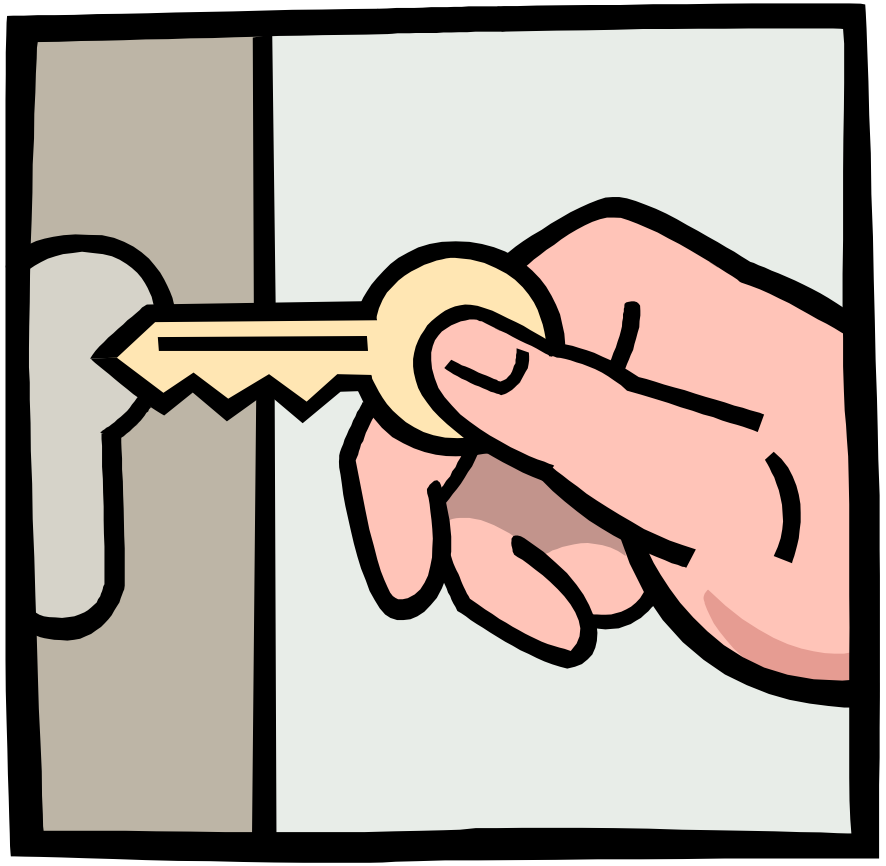
```
Received ATR = 0x3b 0xf0 0x11 0x00 0xff 0x00

CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 06, b6, 84, 89, 33, 88, 8e, Le: 00,
SW1: 90, SW2: 00

CLA: 80, INS: b2, P1: 00, P2: 00, Lc: 10,
00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0a, 0b, 0c, 0d, 0e, 0f,
Le: 40,
66, ff, e8, 04, 8a, 41, 9e, c2, dd, e7, 44, 08, a5, 41, c2, e5,
79, 3d, 65, 31, a5, c6, c8, 54, bd, 49, 52, eb, d3, 65, 0e, b6,
da, 99, f0, e4, 89, b6, 08, a4, f6, 64, f9, 3d, ba, bb, 93, 61,
f8, a4, 95, 3a, 13, 2d, 17, 73, 7b, 4c, 49, 27, 9f, 1e, 8c, 9c,
SW1: 90, SW2: 00
```

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Conclusion



- In this chapter, we have seen
  - An introduction about the security aspects of the smart cards
    - From a hardware point of view
    - From a software point of view

SIM Cards

Proactive SIM cards

Introduction

- In this chapter, we'll see
  - The standards driving the smart cards for mobile telephony
  - What is the SIM Toolkit
  - How Java Card handles the SIM toolkit
  - A full example of a Java Card applet built using the SIM Toolkit library

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

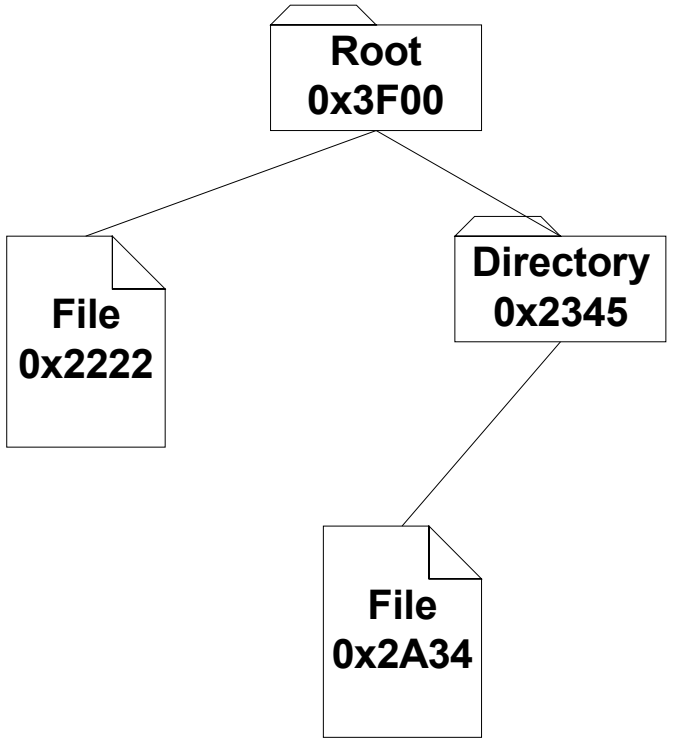
SIM cards

- Standardized by ETSI for GSM
- GSM 11.11 V6.1.0
  - SIM specs
    - **S**ubscriber **I**dentification **M**odule
- GSM 11.14 V7.1.0
  - SIM Toolkit specs
- GSM 03.19 V1.0.0
  - Javacard SIM API

Proactives SIM

- Using the SIM Toolkit, possibility to
  - Program the SIM
  - Make the SIM card application driving the phone
    - Access to keyboard, display, …

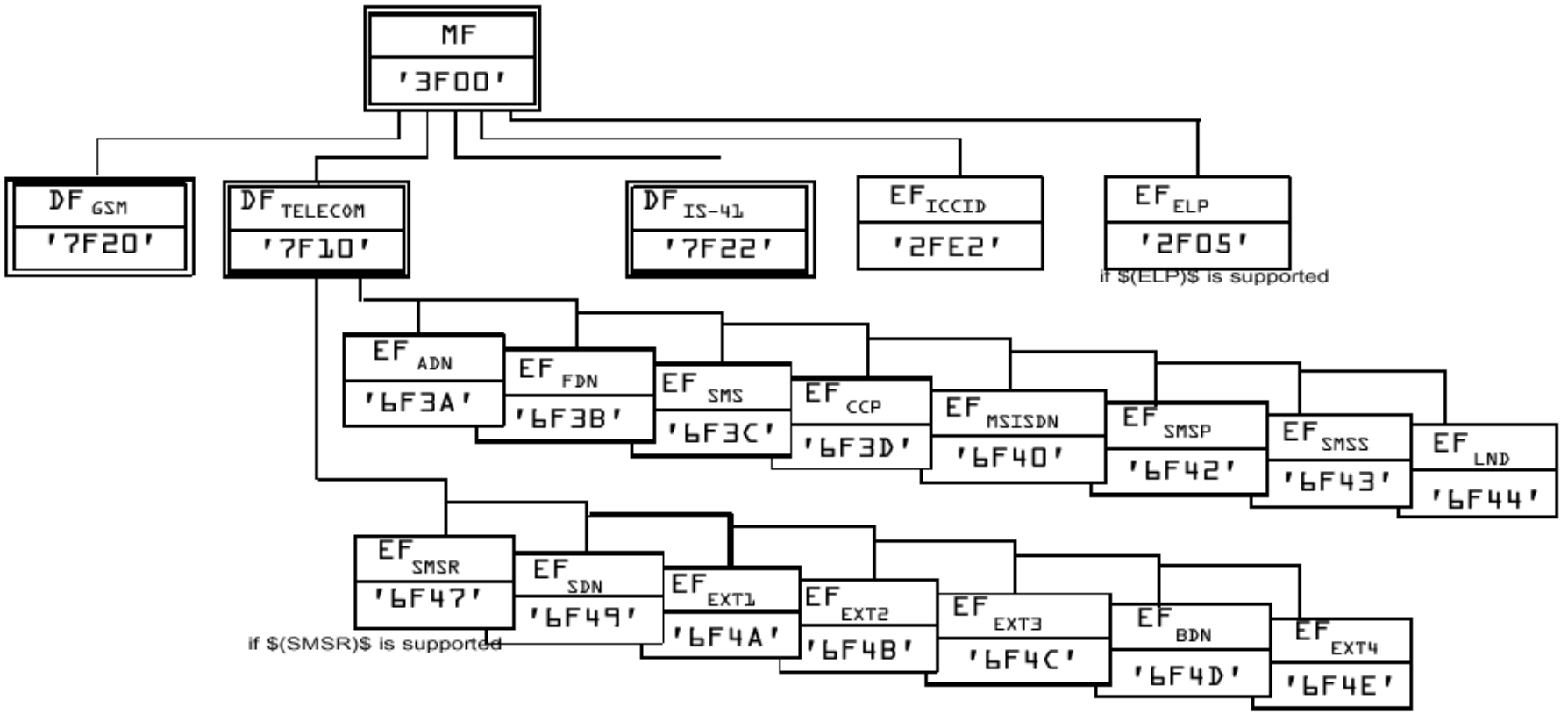
Internal organization



- The SIM contains a certain number of "files" grouped into "directories"
- Terminology:
  - **E**lement **F**ile: file
  - **D**edicated **F**ile: directory

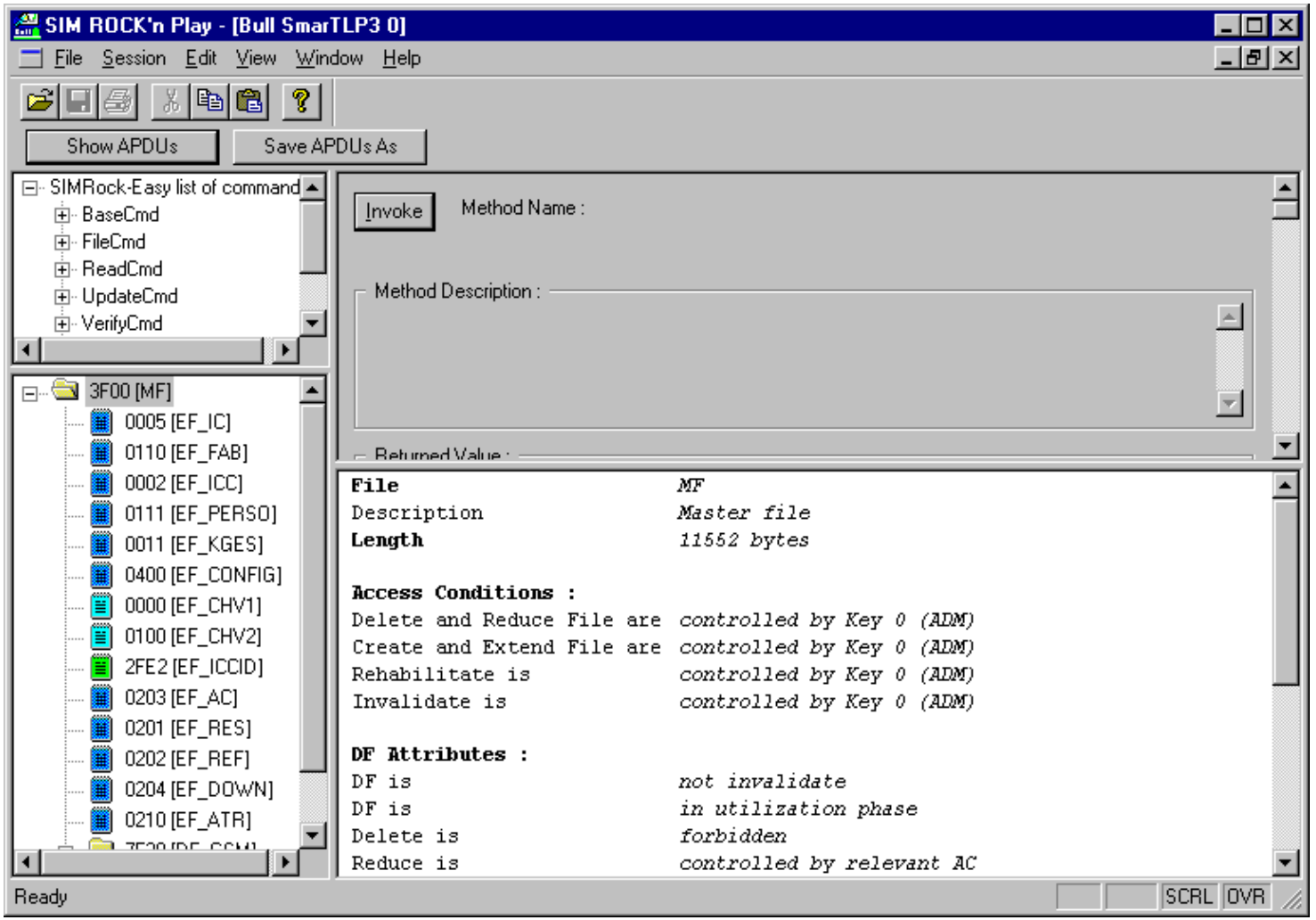
Smart University
State-of-the-Art ICT Knowledge and Training Workshops

File hierarchy



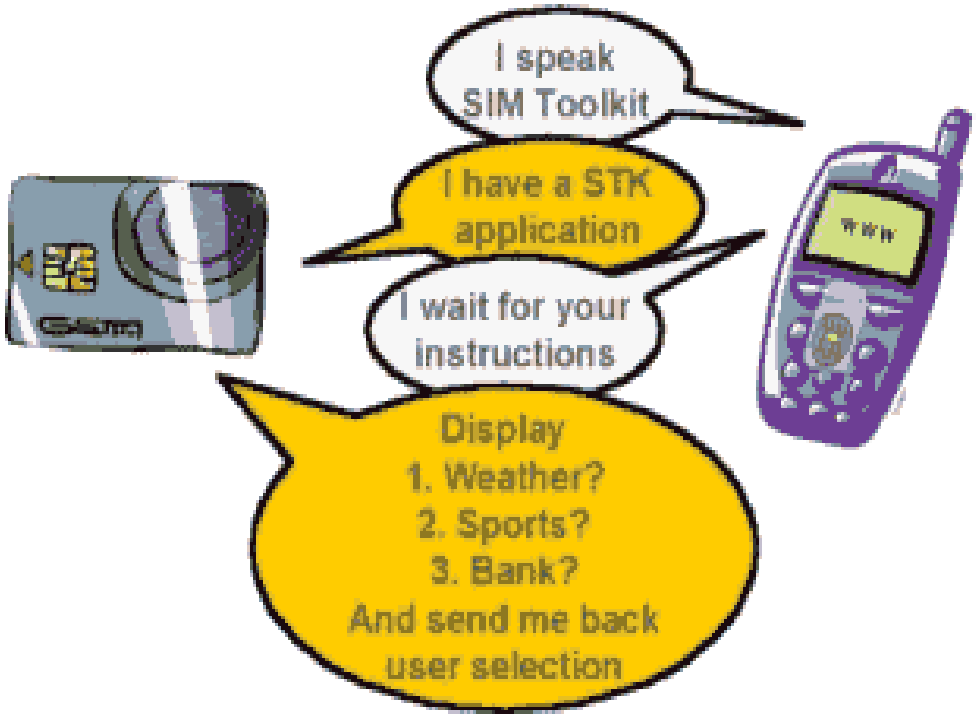
Smart University
State-of-the-Art ICT Knowledge and Training Workshops

File hierarchy



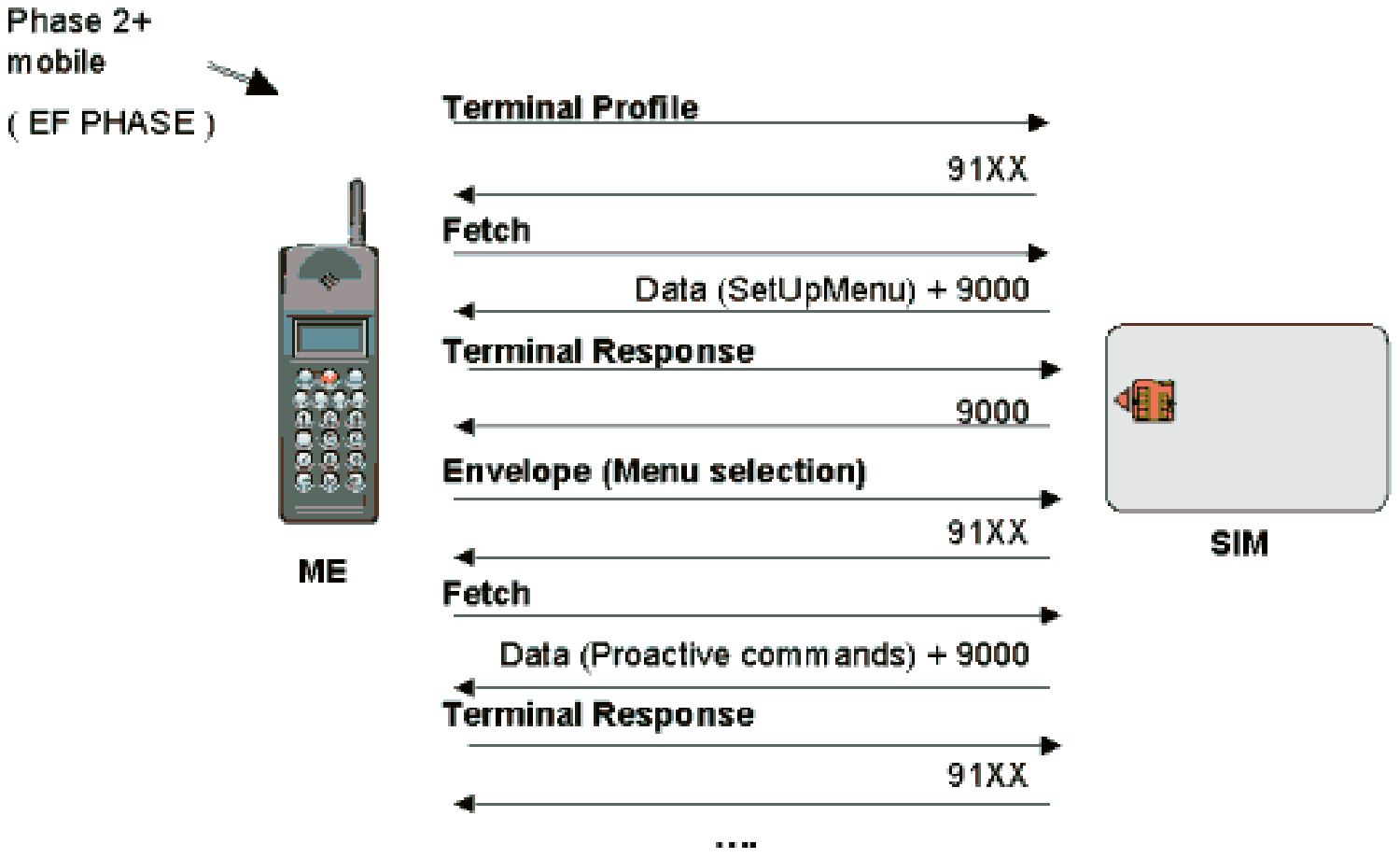
Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Proactive SIM



- The ISO7816 standard does not permit that the card starts talking first
  - A card is waiting for an APDU and responds when it receives the APDU
- Proactive SIM cards use a specific status word to indicate to the **Mobile Equipment** that they want to talk to it

Proactive protocol



Allowed commands for the SIM

```
Proactive SIM: DISPLAY TEXT
Proactive SIM: GET INKEY
Proactive SIM: GET INPUT
Proactive SIM: MORE TIME
Proactive SIM: PLAY TONE
Proactive SIM: POLL INTERVAL
Proactive SIM: POLLING OFF
Proactive SIM: REFRESH
```

```
Proactive SIM: SELECT ITEM
Proactive SIM: SEND SHORT MESSAGE
Proactive SIM: SEND SS
Proactive SIM: SEND USSD
Proactive SIM: SET UP CALL
Proactive SIM: SET UP MENU
Proactive SIM: PROVIDE LOCAL INFORMATION
               (MCC,MNC,LAC,Cell ID&IMEI)
Proactive SIM: PROVIDE LOCAL INFORMATION
               (NMR)
```

- The SIM card can
  - Display text on the phone display
  - Input data from the keyboard
  - Play tone
  - Send a SMS
  - Process an incoming SMS
  - …

SIM Toolkit applet

- A SIM Toolkit applet must
  - Import **`sim.access`** and **`sim.toolkit`** packages
  - Extend the **`javacard.framework.Applet`**
  - Implement the interfaces
    - **`ToolkitInterface`**
    - **`ToolkitConstants`**

SIM Toolkit applet

- Example:

```
import sim.toolkit.*;
import sim.access.*;
import javacard.framework.*;

public class MyApplet1 extends javacard.framework.Applet implements
ToolkitInterface, ToolkitConstants {
// Mandatory variables
private SIMView gsmFile;
private ToolkitRegistry reg;
```

SIMView

- The **SIMView** interface is the interface between the applet and the GSM filesystem
- It proposes
  - Constants to identify in a simple way the regular GSM files
  - Methods to access these files

SIMView

- Example:

```
/** DF under MF */
/** File identifier : DF TELECOM = 0x7F10 */
public static final short FID_DF_TELECOM        = (short)0x7F10;
/** File identifier : DF GSM = 0x7F20 */
public static final short FID_DF_GSM            = (short)0x7F20;
/** File identifier : DF DCS-1800 = 0x7F21 */
public static final short FID_DF_DCS_1800       = (short)0x7F21;
/** File identifier : DF IS-41 = 0x7F22 */
public static final short FID_DF_IS_41          = (short)0x7F22;
/** File identifier : DF FP-CTS = 0x7F23 */
```

SIMView

- Example:

```
    public short select(short fid,
                        byte  fci[],
                        short fciOffset,
                        short fciLength) throws
NullPointerException,
ArrayIndexOutOfBoundsException,
SIMViewException;
```

SIMSystem

- The **SIMSystem** class provides one method which is
  - **SIMView getTheSIMView()**

ToolkitRegistry

- The SIM Applet communicates with the mobile equipment through the ToolkitRegistry
- The SIM applet get an entry from the **ToolkitRegistry** in order
  - To receive and process the events sent by the mobile equipment
  - To send command to the mobile equipment

SIM Toolkit applet

- Example

```
// Main Menu
private byte idMenu1;
private byte[] Menu1;

public MyApplet1() {
// Get the GSM application reference
gsmFile = SIMSystem.getTheSIMView();
// Get the reference of the applet ToolkitRegistry object
reg = ToolkitRegistry.getEntry();
/**@todo: Customize your menu titles here*/
Menu1 = new byte[] { (byte) '1', (byte) ' ', (byte) 'M', (byte) 'e',
(byte) 'n', (byte) 'u', (byte) '1' };
// Define the applet Menu Entry
idMenu1 = reg.initMenuEntry(Menu1, (short) 0, (short) Menu1.length,
PRO_CMD_SELECT_ITEM, false, (byte) 0, (short) 0);
}
```

initMenuEntry

```
public byte initMenuEntry(
      byte[] menuEntry,    /* the menu entry string  */
      short offset,    /* its offset              */
      short length,    /* its length              */
      byte nextAction,     /* action associated      */
      boolean helpSupported,   /* true if help available */
      byte iconQualifier,
      short iconIdentifier /* 0 if no icon           */
)
throws   java.lang.NullPointerException,
         java.lang.ArrayIndexOutOfBoundsException,
         ToolkitException,
         TransactionException
```

SIM Toolkit applet

```
/**
 * Method called by the JCRE at the installation of the applet
 * @param bArray the byte array containing the AID bytes
 * @param bOffset the start of AID bytes in bArray
 * @param bLength the length of the AID bytes in bArray
 */
public static void install(byte[] bArray, short bOffset, byte bLength) {
     // Create the Java SIM toolkit applet
     MyApplet1 StkCommandsExampleApplet = new MyApplet1();
     // Register this applet
     StkCommandsExampleApplet.register(bArray,
     (short) (bOffset + 1), (byte) bArray[bOffset]);
}
```

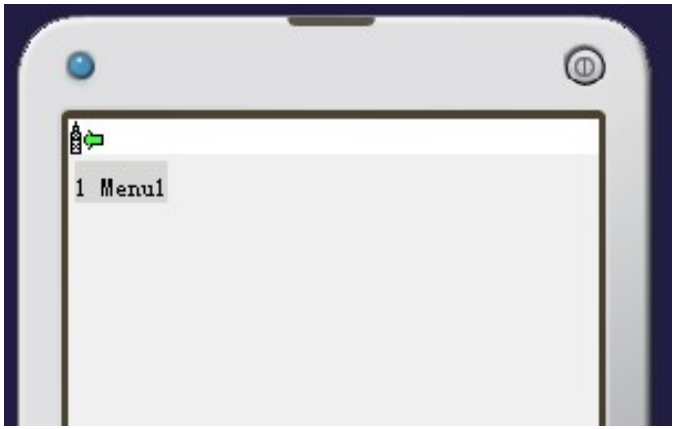
SIM Toolkit applet

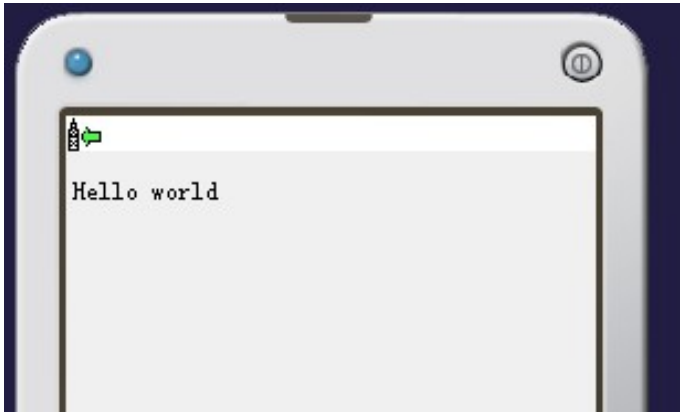
```
/**
 * Method called by the SIM Toolkit Framework
 * @param event the byte representation of the event triggered
 */
public void processToolkit(byte event) {
     // Manage the request following the MENU SELECTION event type
     if (event == EVENT_MENU_SELECTION) {
     // Get the selected item
     EnvelopeHandler envHdlr = EnvelopeHandler.getTheHandler();
     byte selectedItemId = envHdlr.getItemIdentifier();
     // Perform the required service following the Menu1 selected
     // item
     if (selectedItemId == idMenu1) {
          menu1Action();
     }
```

SIM Toolkit applet

```
private byte [] helloWorld;
private void menu1Action() {
// Get the received envelope
ProactiveHandler proHdlr = ProactiveHandler.getTheHandler();
   helloWorld = new byte[]{(byte)'H',(byte)'e',(byte)'l',(byte)'l',
   (byte)'o',(byte)' ',(byte)'w',(byte)'o',(byte)'r',(byte)'l',(byte)'d'};
   // Initialize the display text command
   proHdlr.initDisplayText((byte) 0x00, DCS_8_BIT_DATA, helloWorld,
   (short) 0, (short) (helloWorld.length));
   proHdlr.send();
   return;
}
```

Running





Documentation

- More documentation in
  - 3gpp 43019-560

Conclusion

- In this chapter, we have seen
  - The standards driving the smart cards for mobile telephony
  - What is the SIM Toolkit
  - How Java Card handles the SIM toolkit
  - A full example of a Java Card applet built using the SIM Toolkit library

Smart Card Web Server

An other way for the SIM card to control the handset

Introduction

- In this chapter, we'll see:
  - A new approach to interface the applications in the SIM card, using the handset
  - The architecture of the SCWS
  - A full application for a SIM card supporting SCWS

Introduction

- SIM Toolkit was introduced at the time when handset had few capabilities for interfacing
  - Text oriented display
  - No graphics
  - Hierachical menus
- Modern handsets support
  - Full color graphic interface
  - Point and pin menus

Introduction

- Axalto developers proposed at Cartes 2000 a simplified web server inside the SIM card
  - SESAME 2000
- With
  - the introduction of the USB port
  - the powerfulness of modern SIM card
  - the size of SIM applications

  this solution was rapidly adopted and standardized

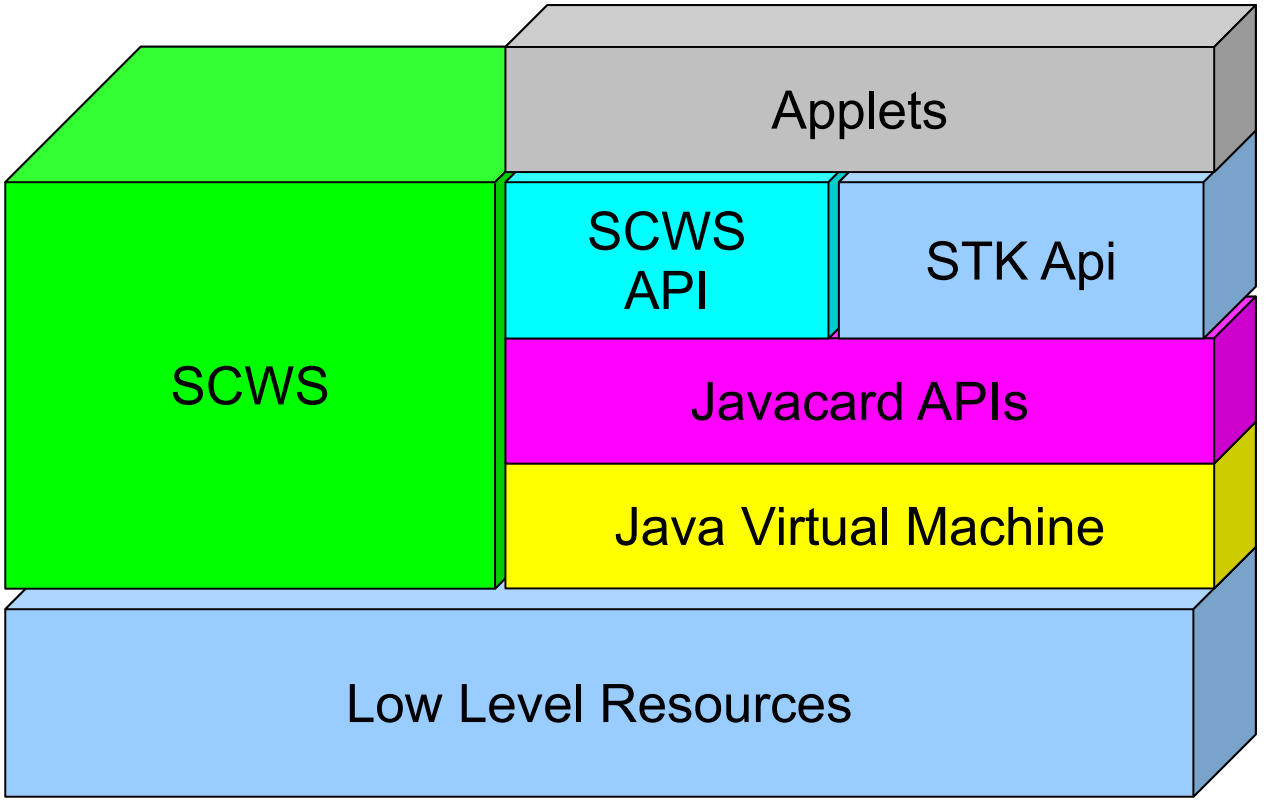
SCWS

- The standard adopted is called:

  Smart **C**ard **W**eb **S**erver

- This standard supposes
  - A TCP/IP link
    - On USB
  - A TCP/IP stack on board

SCWS


Applets
SCWS
API
STK Api
SCWS
Javacard APIs
Java Virtual Machine
Low Level Resources

Packages and classes

```
/*
 * Imported packages
 */
import javacard.framework.*;
import uicc.scws.HttpRequest;
import uicc.scws.HttpResponse;
import uicc.scws.ScwsConstants;
import uicc.scws.ScwsException;
import uicc.scws.ScwsExtension;
import uicc.scws.ScwsExtensionRegistry;
```

Universal Integrated Circuit Card

ScwsConstants

- MIME types
  - **`CONTENT_TYPE_IMAGE_GIF`**
  - **`CONTENT_TYPE_TEXT_HTML`**
- Status code
  - **`SC_OK`** (200)
  - **`SC_NOT_FOUND`** (404)
- Parsing tags
  - **`URI_QUERY_TAG`**

ScwsExtension

- The applet (servlet!) working in SCWS mode must implement **`ScwsExtension`**
- That means overiding the methods
  - **`doGet()`**
  - **`doPost()`**
  - **`doHead()`**
  - **`...`**

HttpRequest

- Not really the J2EE HttpRequest but enough to extract data from a HTTP request
- Provides methods like
  - **`findAndCopyKeywordValue`**
  - **`getContentLength`**
  - **`getContentType`**

`HttpResponse`

- As for **`HttpRequest`**, helps the user to provide an HTTP response to the request
- Provides methods like
  - **`setContentType()`**
  - **`appendContent()`**
  - **`writeStatusCode()`**
  - **`flush()`**

Example

- In the next servlet, the strings are encoded as arrays of bytes
  - **`String`**s are not supported by Java Card 2
- In the next two pages, the pseudo code written in comment show how the servlet would be written if **`String`** was supported by this release of Java Card

Example

```
/*
public class HelloWorld extends javacard.framework.Applet  implements
   AppletEvent, ScwsExtension {
public final static String url = "/HelloWorld";
public final static String appId = "HelloWorld;

public byte[] temporaryBuffer;
public final static short TEMPORARY_BUFFER_LENGTH = (short) 100;
public final static String HTML_BEGIN = "<html>"+"<head>"+
          "<title>"+"Hello"+"</title>"+"</head>"+
          "<body BGCOLOR=\"#FFFFFF\">"+"<center>";
public final static String HELLO = "Hello ";
public final static String HTML_END = "</center>"+"</body>"+
          "</html>"
```

Example

- Unfortunately String are not yet supported by Java Card
  - Strings are supported by Java Card 3
- The arrays of bytes are not so easy to read, but the result is the same

Example

```
public class HelloWorld extends javacard.framework.Applet implements
   AppletEvent, ScwsExtension {

/** the servlet url */

public final static byte[] url = { (byte)'/', (byte)'H', (byte)'e',
   (byte)'l', (byte)'l', (byte)'o', (byte)'W', (byte)'o', (byte)'r',
   (byte)'l', (byte)'d' };

public final static byte[] appId = { (byte)'H', (byte)'e', (byte)'l',
   (byte)'l', (byte)'o', (byte)'W', (byte)'o', (byte)'r', (byte)'l',
   (byte)'d' };
```

Example

```
// Temporary operation buffer
public byte[] temporaryBuffer;
public final static short TEMPORARY_BUFFER_LENGTH = (short) 100;
public final static byte[] HTML_BEGIN = {
   (byte)'<',(byte)'h',(byte)'t',(byte)'m',(byte)'l',(byte)'>',
   (byte)'<',(byte)'h',(byte)'e',(byte)'a',(byte)'d',(byte)'>',
   (byte)'<',(byte)'t',(byte)'i',(byte)'t',(byte)'l',(byte)'e',(byte)'>',
   (byte)'H',(byte)'e',(byte)'l',(byte)'l',(byte)'o',
   (byte)'<',(byte)'/',(byte)'t',(byte)'i',(byte)'t',(byte)'l',(byte)'e',
    (byte)'>',
   (byte)'<',(byte)'/',(byte)'h',(byte)'e',(byte)'a',(byte)'d',(byte)'>',
```

Example

```
   (byte)'<',(byte)'b',(byte)'o',(byte)'d',(byte)'y',(byte)' ',

   (byte)'B',(byte)'G',(byte)'C',(byte)'O',(byte)'L',(byte)'O',(byte)'R',
    (byte)'=',(byte)'"',(byte)'#',(byte)'F',(byte)'F',(byte)'F',(byte)'F',
    (byte)'F',(byte)'F',(byte)'"',(byte)'>',

   (byte)'<',(byte)'c',(byte)'e',(byte)'n',(byte)'t',(byte)'e',(byte)'r',
    (byte)'>'};
public final static byte[] HELLO ={(byte)'H',(byte)'e',(byte)'l',
    (byte)'l',(byte)'o',(byte)' '};


public final static byte[] HTML_END = {

   (byte)'<',(byte)'/',(byte)'c',(byte)'e',(byte)'n',(byte)'t',(byte)'e',
    (byte)'r',(byte)'>',

   (byte)'<',(byte)'/',(byte)'b',(byte)'o',(byte)'d',(byte)'y',

   (byte)'<',(byte)'/',(byte)'h',(byte)'t',(byte)'m',(byte)'l',(byte)'>'};
```

Example

```
public HelloWorld(byte[] buffer, short offset, byte length) {
    // First LV is instance AID
    short aid = offset;
    offset += buffer[offset] + (byte) 1;
    // Second LV is Privilege
    offset += buffer[offset] + (byte) 1;
    // Third LV is specific install parameter (extract from TAG C9)
    offset++; // skip C9 Length
    // Register the new applet instance to the JCRE
    register(buffer, (short) (aid + (short) 1), buffer[aid]);
    //Register application id,there is corresponding appId in the
    // Run/Debug configuration for URL Mapping
    ScwsExtensionRegistry.register(this, appId, (short) 0,
                                          (short) appId.length);
```

Example

```
try {
      // Create a temporary buffer for read/write
      temporaryBuffer = JCSystem.makeTransientByteArray(
      TEMPORARY_BUFFER_LENGTH, JCSystem.CLEAR_ON_RESET);
} catch (SystemException se) {
      // create buffer in persistent memory as not enough transient
      // is available
      temporaryBuffer = new byte[TEMPORARY_BUFFER_LENGTH];
}
```

Example

```
public void doGet(HttpRequest req, HttpResponse resp) throws ScwsException {
  try {
    resp.writeStatusCode(ScwsConstants.SC_OK);
    resp.setContentType(ScwsConstants.CONTENT_TYPE_TEXT_HTML);
    resp.enableChunkMode();
    short queryLength = req.findAndCopyKeywordValue(
          ScwsConstants.URI_QUERY_TAG, temporaryBuffer,(short)0,
          (short)temporaryBuffer.length);
    resp.appendContent(HTML_BEGIN,(short)0,(short)HTML_BEGIN.length);
    resp.appendContent(HELLO, (short)0, (short)HELLO.length);
    resp.appendContent(temporaryBuffer, (short)0, queryLength);
    resp.appendContent(HTML_END,(short)0, (short)HTML_END.length);
  }catch(Exception e) {resp.writeStatusCode(ScwsConstants.SC_BAD_REQUEST);}
  resp.flush();
```

Static HTML

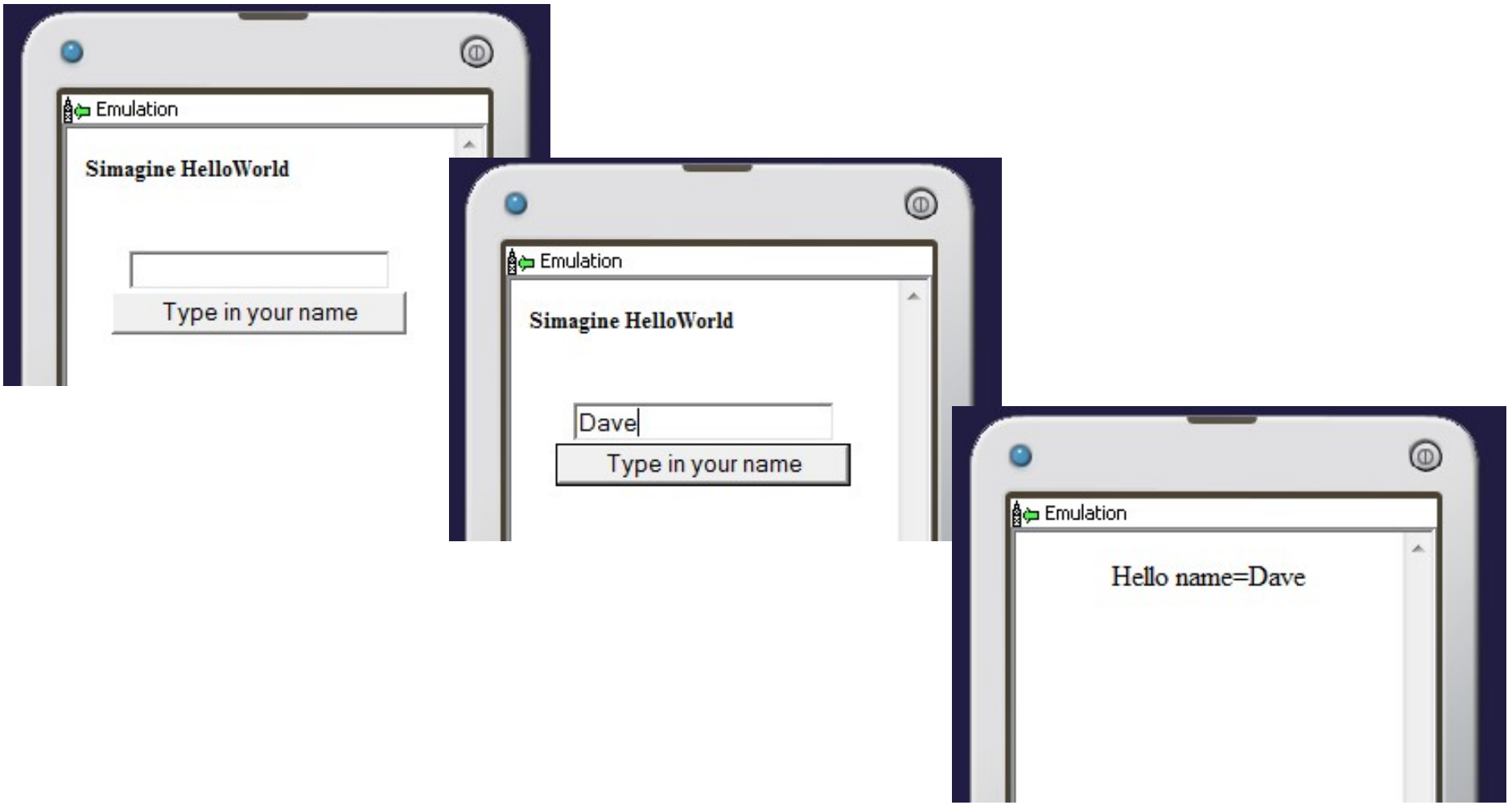
- Static HTML file : helloworld.html

```
<html>
<body>
<p>Simagine HelloWorld</p>
<br>
<form action="/HelloWorld" method="get">
    <input name="name" type="text"> <br>
<input value="Type in your name" type="submit">
</form>
</body>
</html>
```

URI of the SCWS servlet

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Running



Conclusion

- In this chapter, we have seen:
  - A new approach to interface the applications in the SIM card, using the handset
  - The architecture of the SCWS
  - A full application for a SIM card supporting SCWS

Java Card 3.0 Connected Edition

A new and rich flavour of Java Card

Introduction

- In this chapter, we'll see
  - The main enhancements introduced by Java Card 3
  - The restrictions of Java Card 3 compared to Java SE
  - A full example of a servlet

Features

- Java Card 3.0 has two editions:
  - The Classic Edition
    - Compatible with Java Card 2
    - Applications are built with applets
  - The Connected Edition
    - With a WEB server embedded
    - HTTP, TCP/IP over USB

Features

- Java Card 3.0 classic edition remains applet oriented
- Java Card 3.0 connected edition is servlet oriented
  - Specifications of the supported servlets are extracted from the Servlet API Specifications 2.4
    - Everything which deals with floating point numbers, J2EE, etc. are not taken in account.

Features

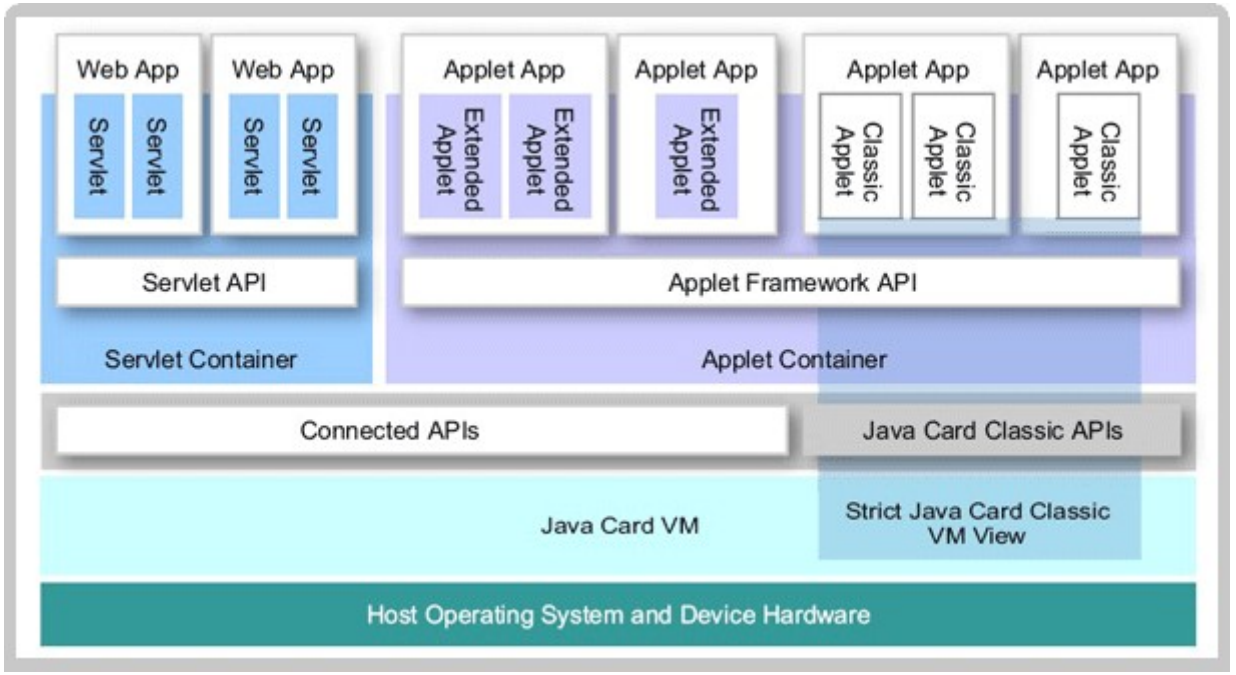
- But, like traditionnal servlets, the Java Card 3 servlets support the methods:
  - doGet
  - doPost
  - doHead
  - doPut
  - doDelete
  - doOptions
  - doTrace

Features

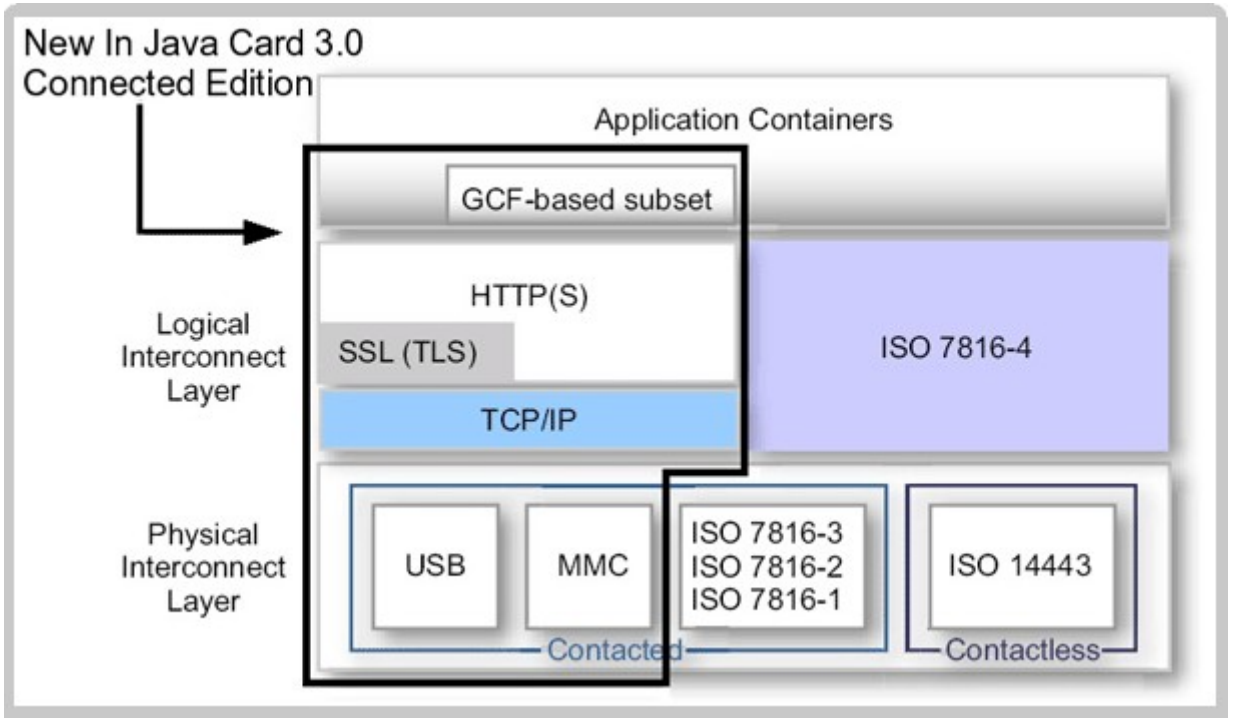
- Better support of the Java language
  - All data types except float and double
  - Multiple threads
  - Extensive API support (java.lang, java.util, GCF, and so on)
  - Direct handling of class files, with all loading and linking on card
  - All new Java language syntax constructs, like enums, generics, enhanced for loops, auto boxing/unboxing, and so on
  - Automatic garbage collection

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Architecture



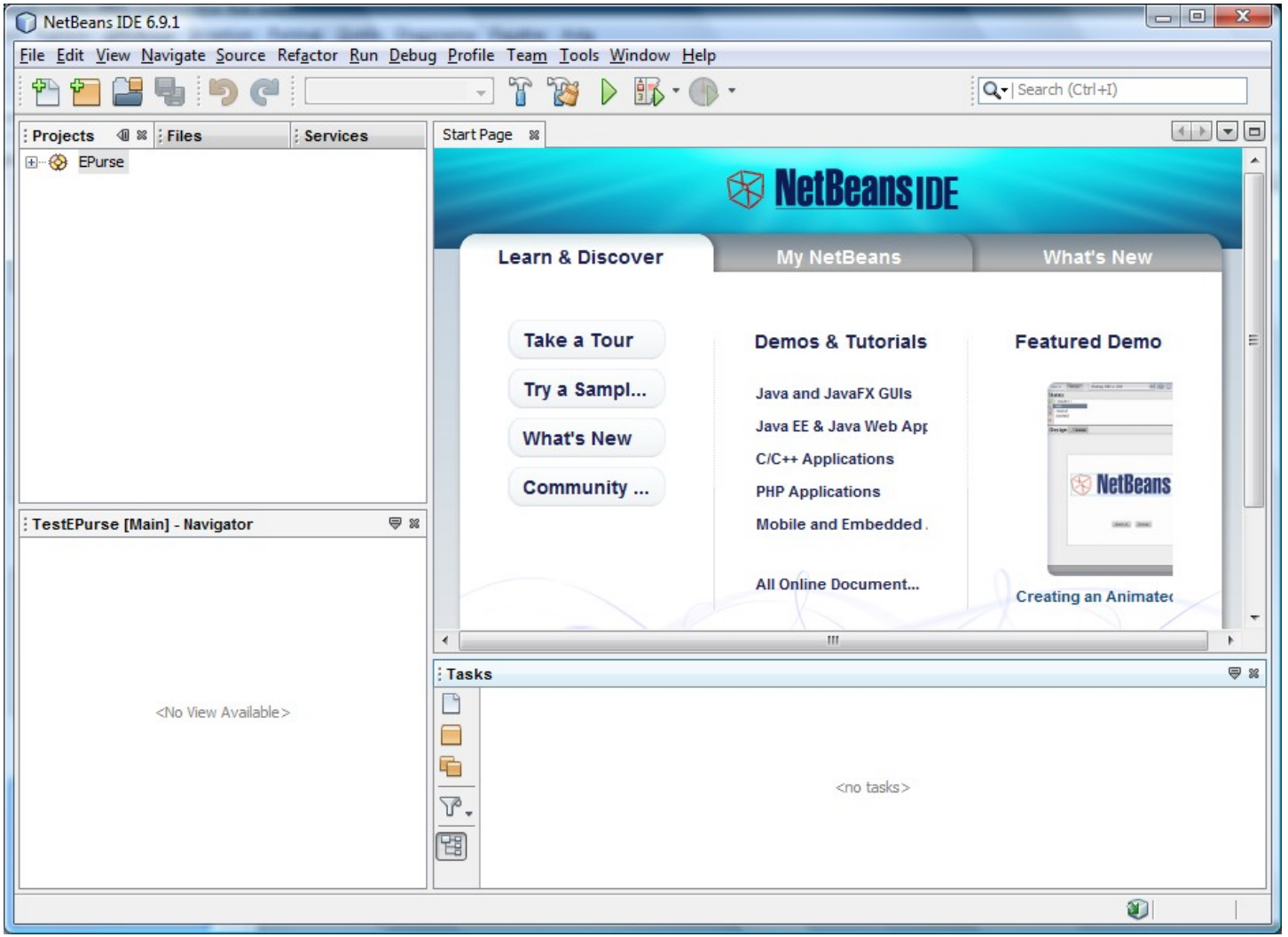
Architecture



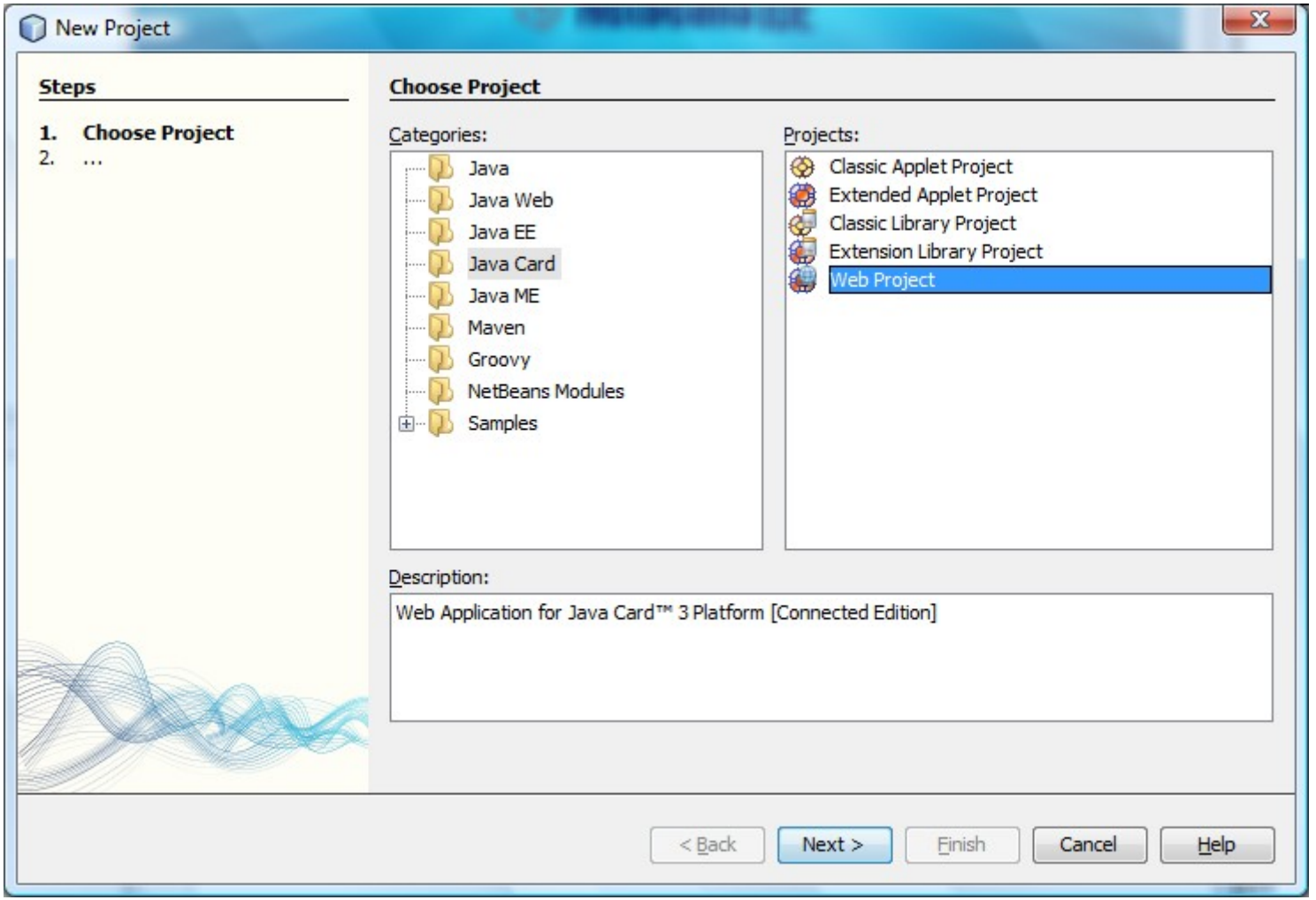
Example

- The following example is created with NetBeans 6.9 with the Java Card wizard
- It is the web instance of the very well known « Hello world » program
  - Most code is automatically generated by the Java Card wizard

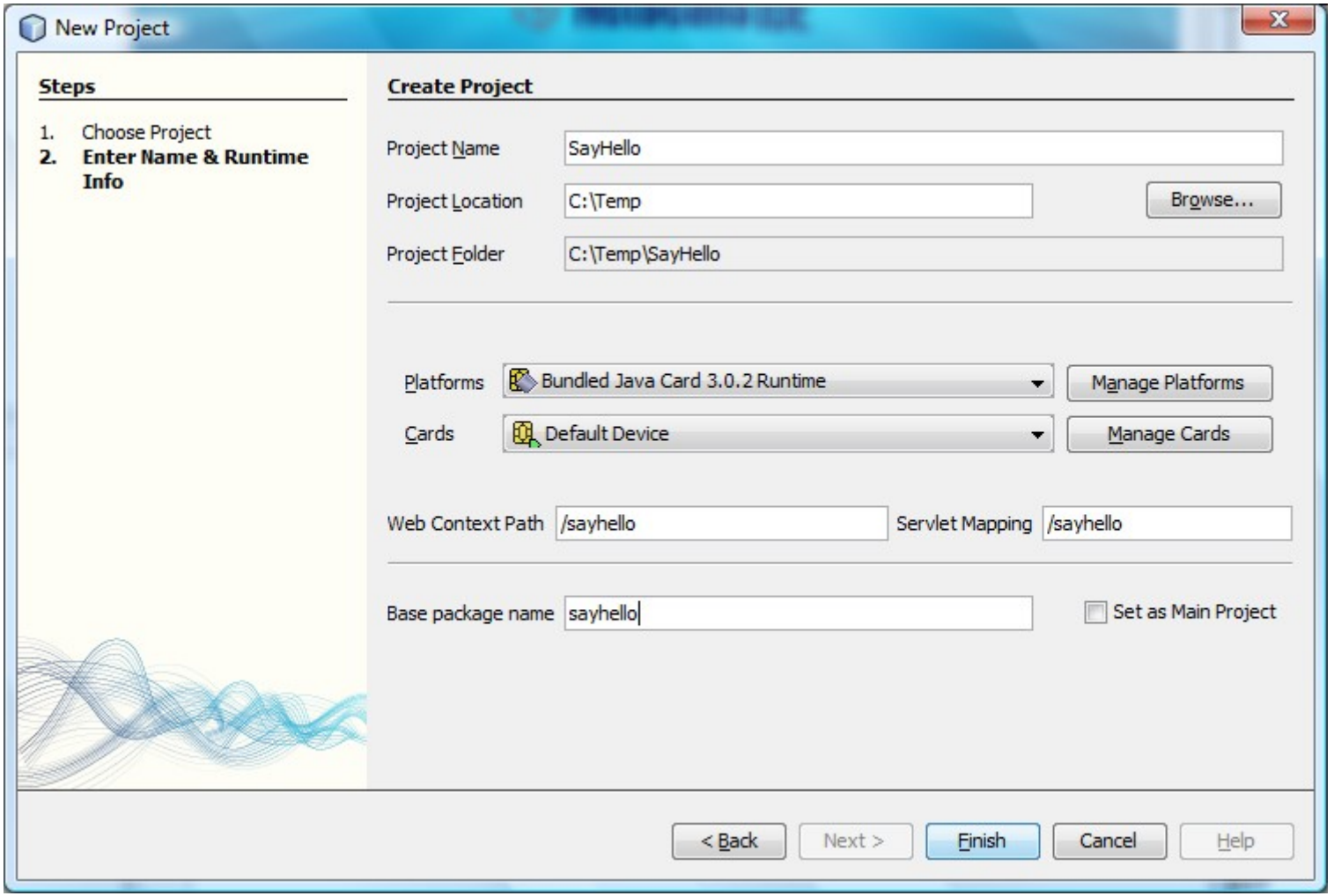
Example


NetBeans IDE 6.9.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects
Files
Services
EPurse
Start Page
NetBeans IDE
Learn & Discover
My NetBeans
What's New
Take a Tour
Try a Sampl...
What's New
Community ...
Demos & Tutorials
Java and JavaFX GUIs
Java EE & Java Web App
C/C++ Applications
PHP Applications
Mobile and Embedded
All Online Document...
Featured Demo
Creating an Animatec
TestEPurse [Main] - Navigator
<No View Available>
Tasks
<no tasks>

Example

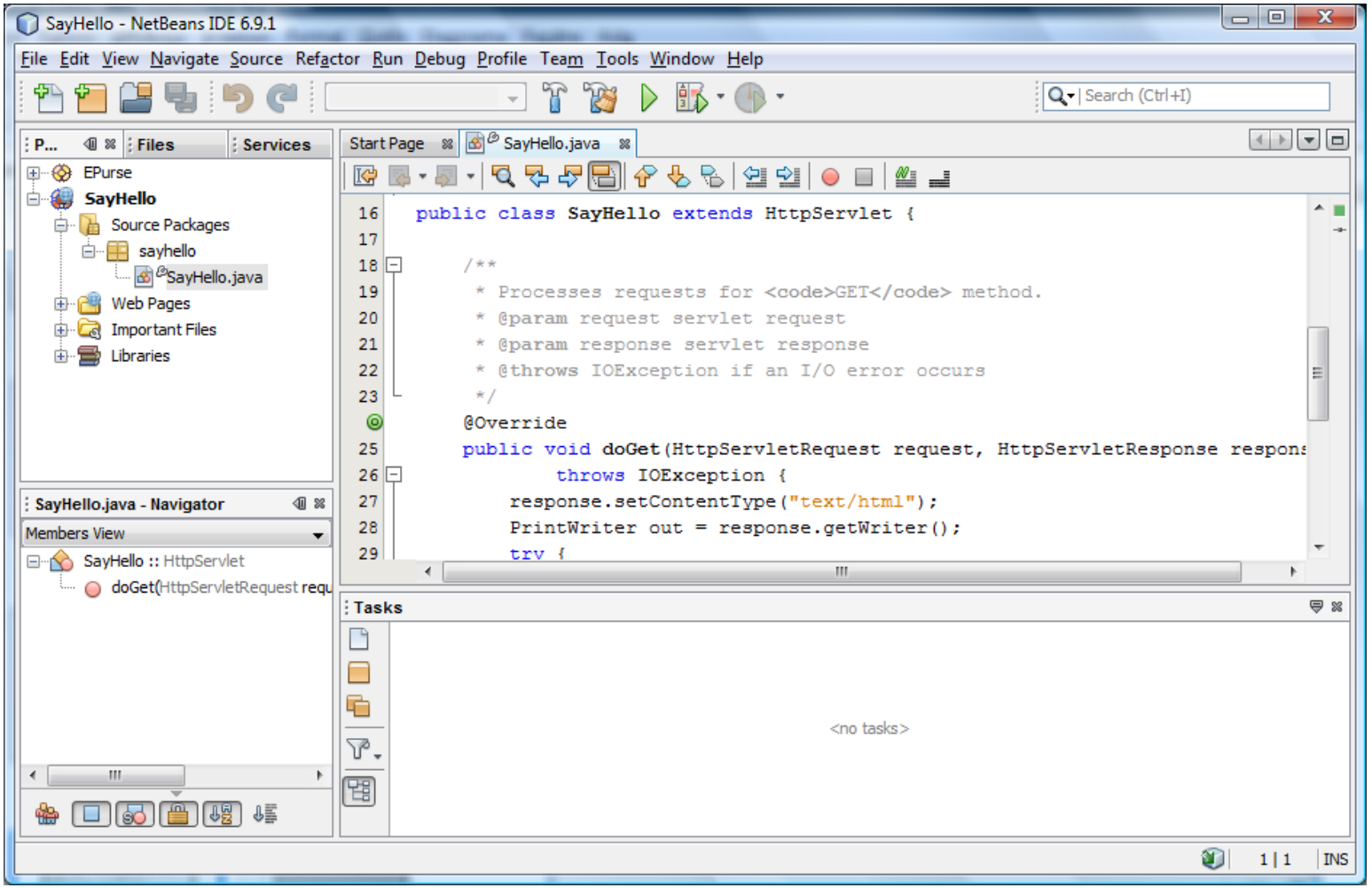

New Project
Steps
1. Choose Project
2. ...
Choose Project
Categories:
Java
Java Web
Java EE
Java Card
Java ME
Maven
Groovy
NetBeans Modules
Samples
Projects:
Classic Applet Project
Extended Applet Project
Classic Library Project
Extension Library Project
Web Project
Description:
Web Application for Java Card™ 3 Platform [Connected Edition]
< Back
Next >
Finish
Cancel
Help

Example


New Project
Steps
1. Choose Project
2. Enter Name & Runtime Info
Create Project
Project Name
SayHello
Project Location
C:\Temp
Browse...
Project Folder
C:\Temp\SayHello
Platforms
Bundled Java Card 3.0.2 Runtime
Manage Platforms
Cards
Default Device
Manage Cards
Web Context Path
/sayhello
Servlet Mapping
/sayhello
Base package name
sayhello
Set as Main Project
< Back
Next >
Finish
Cancel
Help

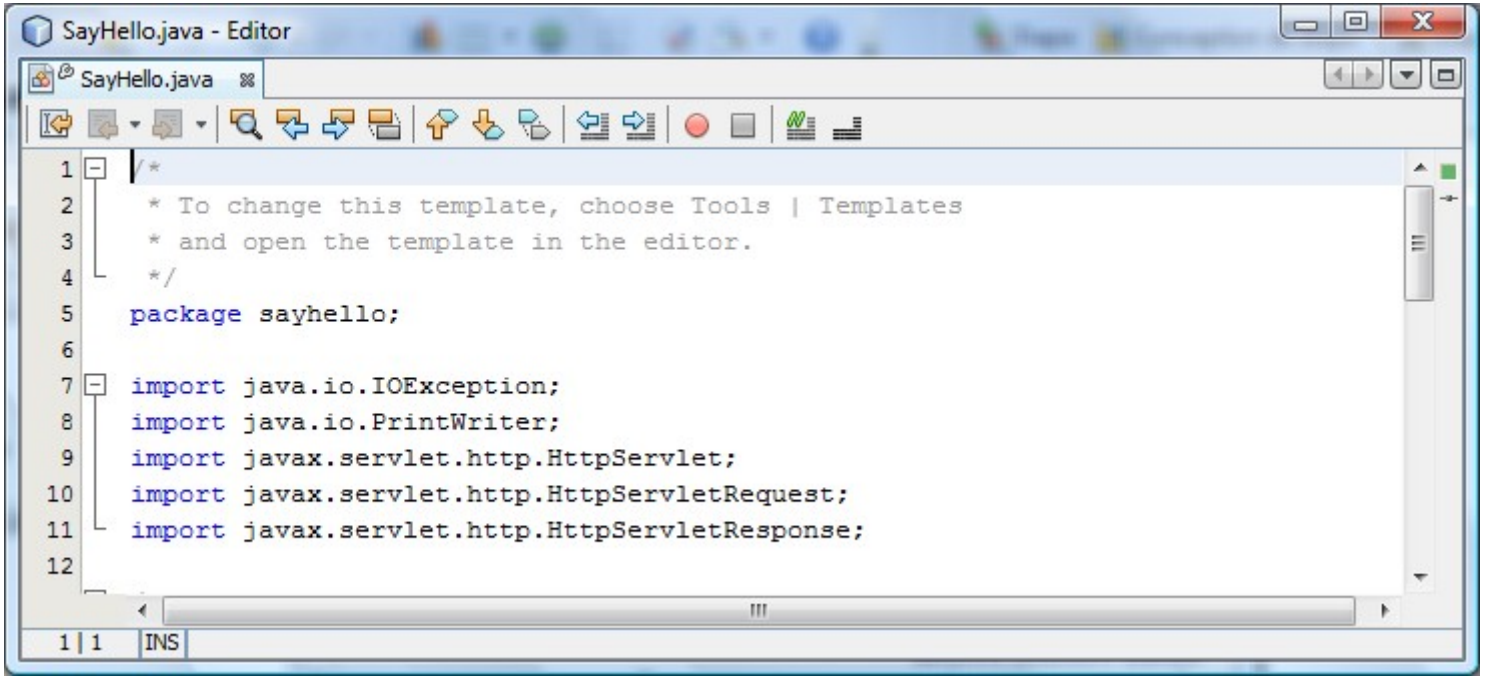
Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example



Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example



Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example

SayHello.java - Editor

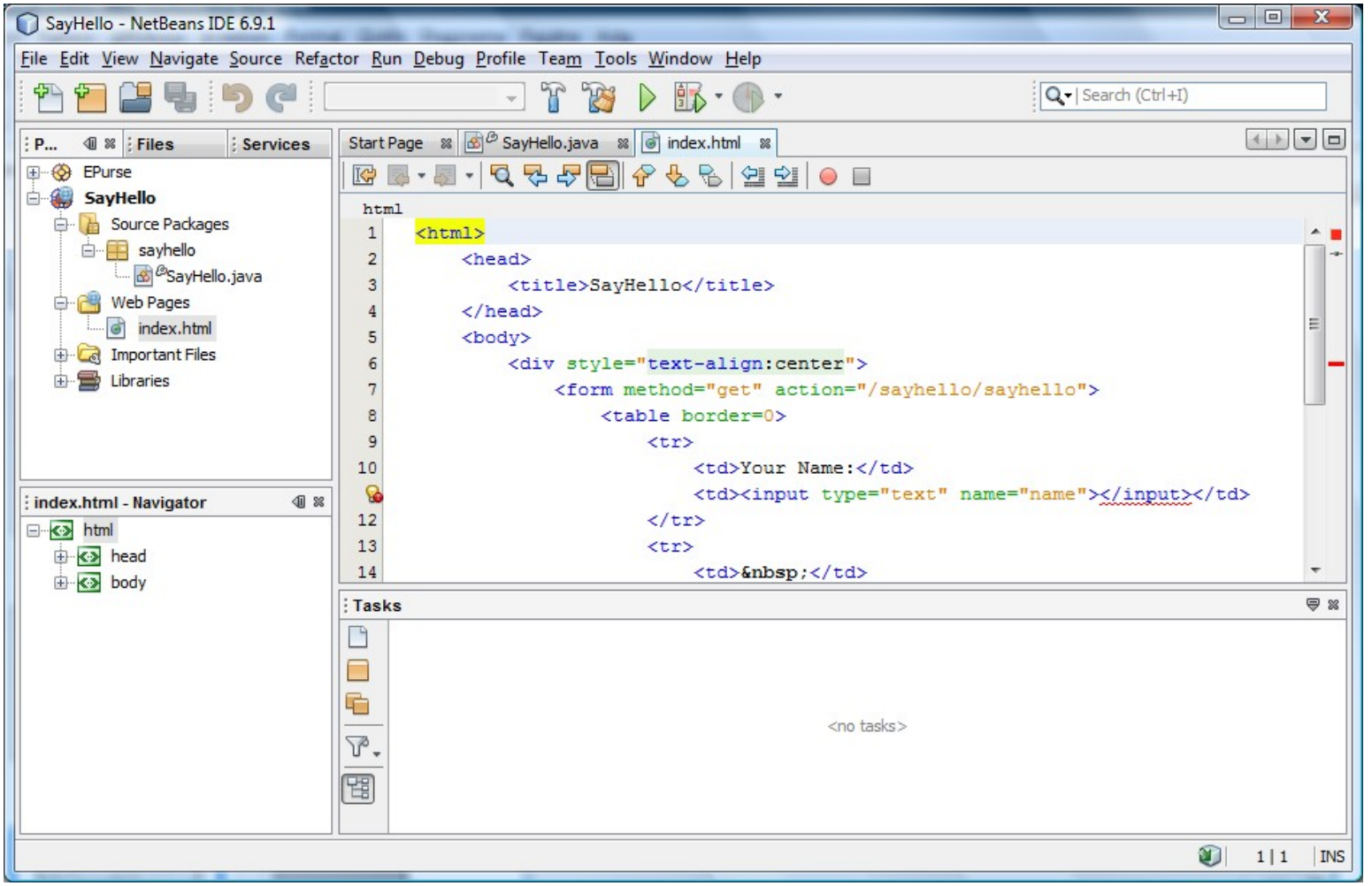
SayHello.java

```
16  public class SayHello extends HttpServlet {
17
19      @Override
20      public void doGet(HttpServletRequest request, HttpServletResponse response)
21              throws IOException {
22          response.setContentType("text/html");
23          PrintWriter out = response.getWriter();
24          try {
25              out.println("<html><head><title>SayHello</title></head>");
26              out.println("<body><h1>SayHello</h1>");
27              out.println("Hello from sayhello.SayHello to");
28              out.println(request.getParameter("name"));
29              out.println("</body></html>");
30          } finally {
31              out.close();
32          }
33      }
34  }
```

1 | 1 INS

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example



Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example

index.html - Editor

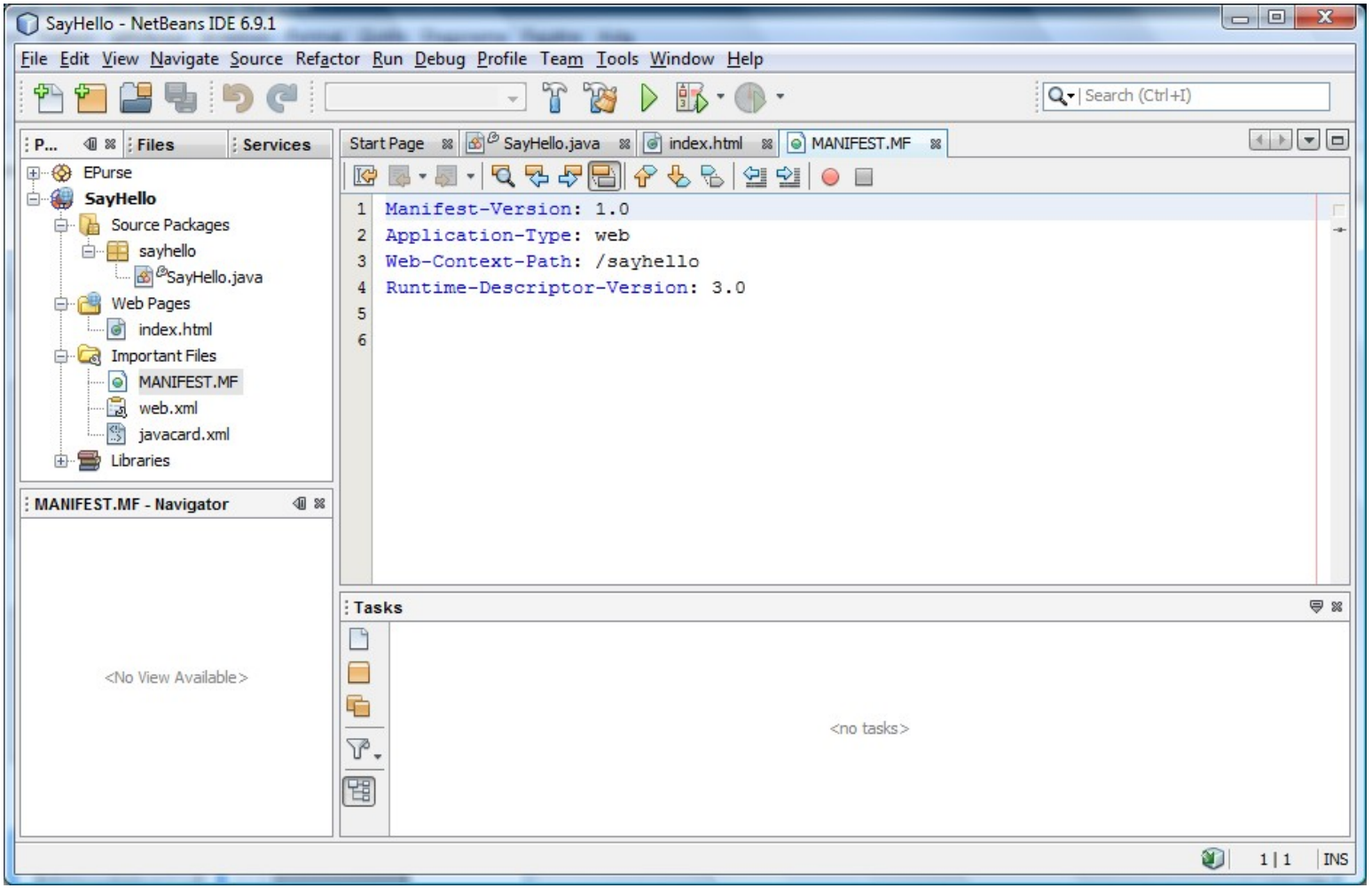
index.html

```
1   <html>
2       <head>
3           <title>SayHello</title>
4       </head>
5       <body>
6           <div style="text-align:center">
7               <form method="get" action="/sayhello/sayhello">
8                   <table border=0>
9                       <tr>
10                          <td>Your Name:</td>
                            <td><input type="text" name="name"></input></td>
12                      </tr>
13                      <tr>
14                          <td> </td>
15                          <td><input type="submit" value="Say Hello"></td>
16                      </tr>
17                  </table>
18              </form>
19          </div>
20      </body>
21  </html>
```

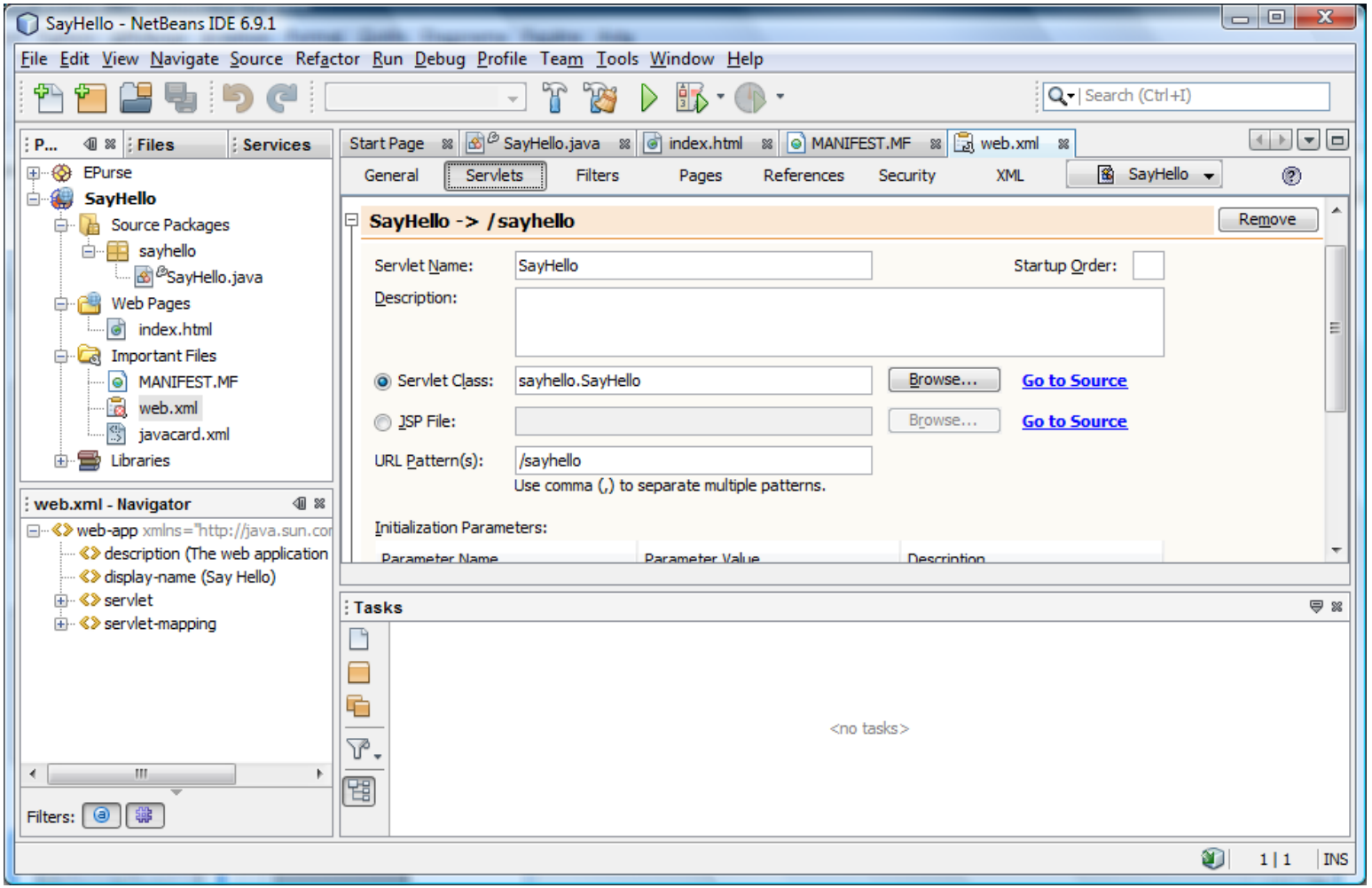
21 | 8 INS

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example

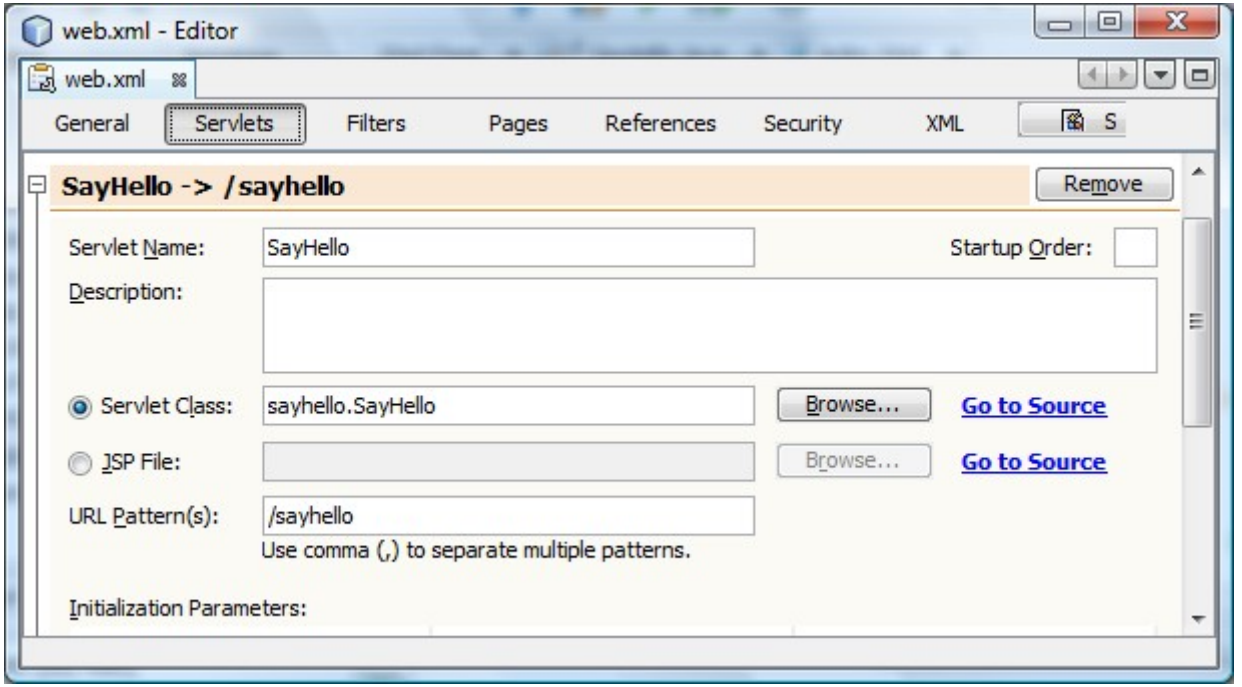


Example


SayHello - NetBeans IDE 6.9.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Files
Services
EPurse
SayHello
Source Packages
sayhello
SayHello.java
Web Pages
index.html
Important Files
MANIFEST.MF
web.xml
javacard.xml
Libraries
web.xml - Navigator
web-app xmlns="http://java.sun.cor
description (The web application
display-name (Say Hello)
servlet
servlet-mapping
Filters:
Start Page
SayHello.java
index.html
MANIFEST.MF
web.xml
General
Servlets
Filters
Pages
References
Security
XML
SayHello
SayHello -> /sayhello
Remove
Servlet Name:
SayHello
Startup Order:
Description:
Servlet Class:
sayhello.SayHello
Browse...
Go to Source
JSP File:
Browse...
Go to Source
URL Pattern(s):
/sayhello
Use comma (,) to separate multiple patterns.
Initialization Parameters:
Parameter Name
Parameter Value
Description
Tasks
<no tasks>
1 | 1
INS

Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example



Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example

web.xml - Editor

web.xml

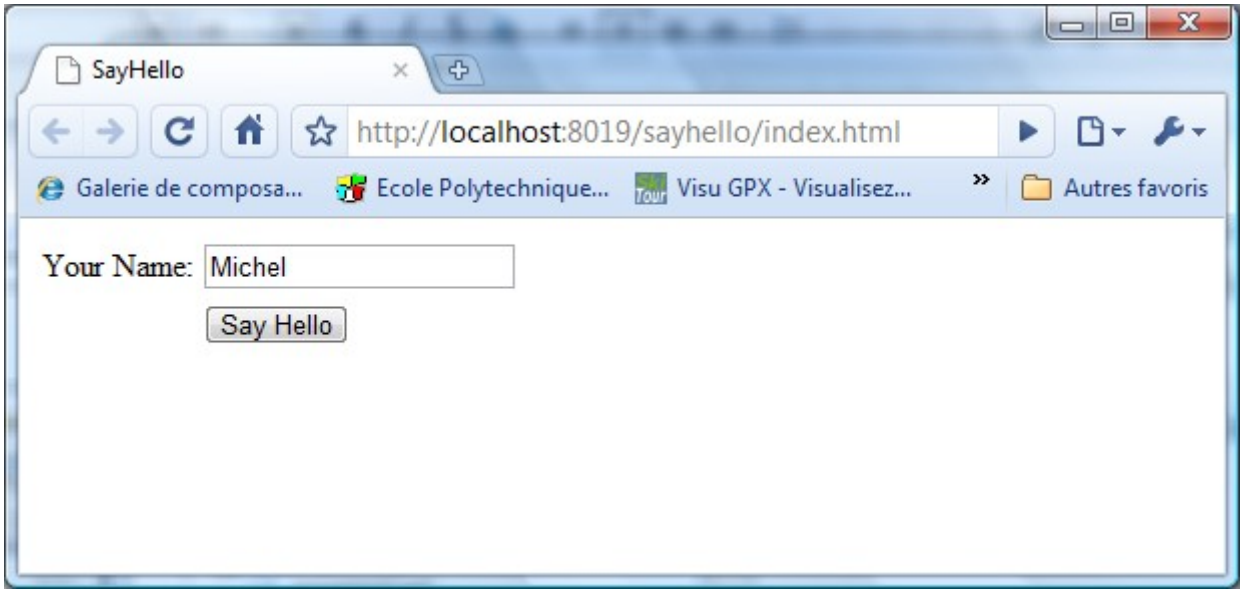
General | Servlets | Filters | Pages | References | Security | XML

```
1  <web-app xmlns="http://java.sun.com/xml/ns/j2ee"
2           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3           version="2.4"
4           xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/javacard/jcweb-app_3_0.xsd">
5      <display-name>Say Hello</display-name>
6      <servlet>
7          <servlet-name>SayHello</servlet-name>
8          <servlet-class>sayhello.SayHello</servlet-class>
9      </servlet>
10     <servlet-mapping>
11         <servlet-name>SayHello</servlet-name>
12         <url-pattern>/sayhello</url-pattern>
13     </servlet-mapping>
14 </web-app>
```

13 | 23 INS

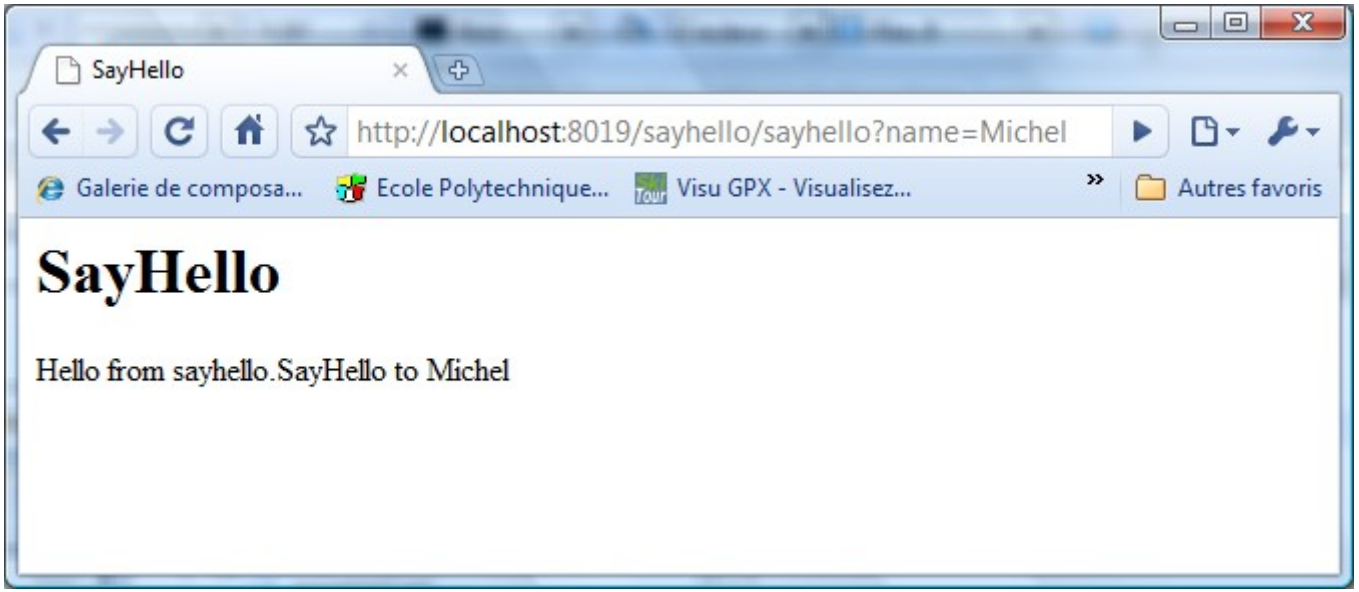
Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example



Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Example



Conclusion

- In this chapter, we have seen
  - The main enhancements introduced by Java Card 3
  - The restrictions of Java Card 3 compared to Java SE
  - A full example of a servlet


Smart University
State-of-the-Art ICT Knowledge and Training Workshops

Conclusion

Conclusion

- In 1996, the Java Card system changed dramatically the way to program secure applications for smart cards
- Despite many concurents on the field, this system remains today the first language for smart cards in the world
- Combined with Java for Mobile Equipment it represents the solution to develop secure applications for the future powerful smartphones