

**IGOR MEDEIROS**



# Java Card no Mercado Brasileiro

Igor Medeiros  
[www.igormedeiros.com.br](http://www.igormedeiros.com.br)



# Agenda

- Mercado Brasileiro
- Smart cards
  - Arquitetura de software e hardware
  - Arquitetura de conectividade
- Nova Smart Card I/ O API (Java 6)
- Java Card



# Palestrante



**Igor Medeiros**

- Evangelista Java Card
- Bacharel em ciência da computação
- Trabalha cerca de 4 anos com Java
- Pioneiro na produção de literatura Java Card no Brasil
- Escreve para a revista MundoJava
- Colunista do PortalJava.com
- Mantenedor do Portal Java Card



# O que é um Smart Card?

- Cartão com chip embutido
- Transmite, armazena e processa dados
- Características definidas na ISO 7816



# Aplicações com Cartões inteligentes



**e- CPF**  
Certificação digital



# Aplicações com Cartões inteligentes



**Bilhete único**  
Transporte público de São Paulo

# Aplicações com Cartões inteligentes



Controle de estimativa de tráfego do trânsito de veículos em São Paulo

# Aplicações com Cartões inteligentes



GSM SIM Cards



# Aplicações com Cartões inteligentes



Uso de RFID contra falsificação de remédios



# Aplicações com Cartões inteligentes



Controle de acesso



# Aplicações com Cartões inteligentes



Cartão de sócio do clube

# Aplicações com Cartões inteligentes



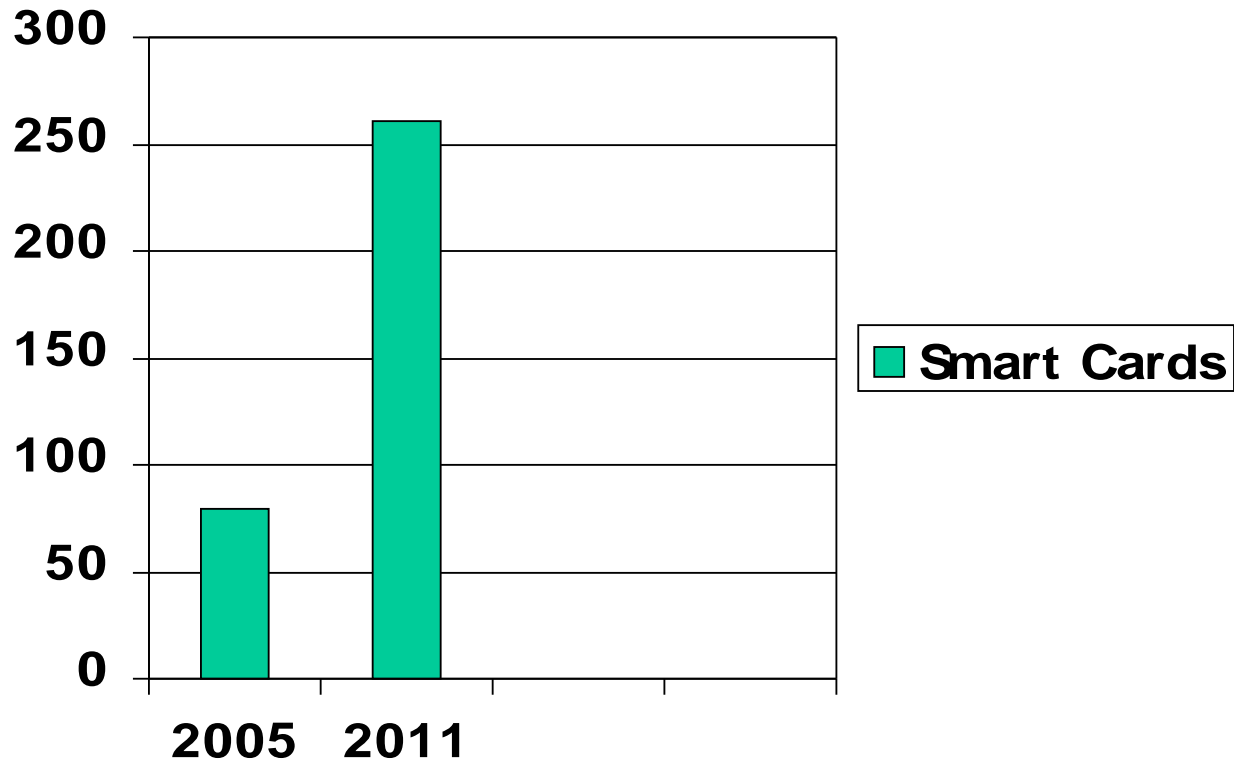
Ingressos para a copa 2006  
Com chips RFID



# Motivação



Quantidade de Smart Cards no Brasil (milhões)



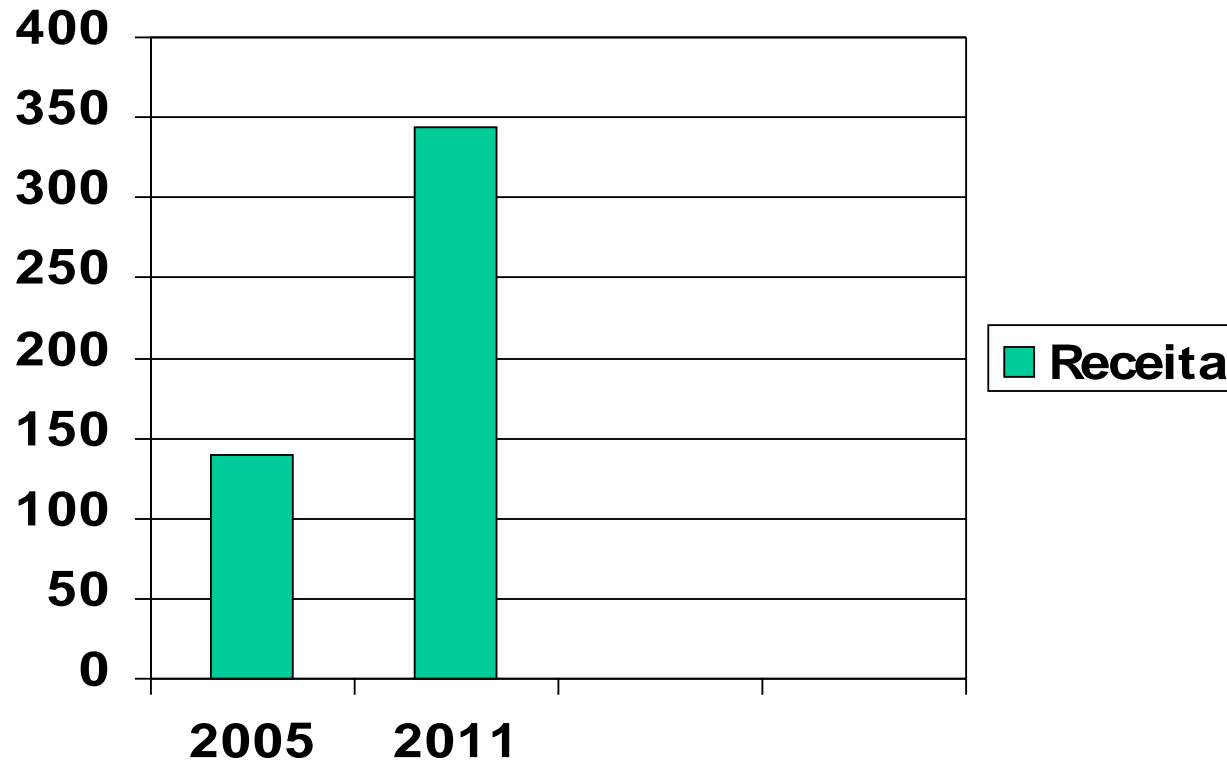
Fonte: Frost & Sullivan



# Motivação



Crescimento da Receita (milhões de US\$)



Fonte: Frost & Sullivan



# Motivação

## Desenvolvedores para **Smart Card** no **Brasil?**



# Motivação

- 99% dos POS Visanet aceitam smart cards
- 25% ATMs aceitam smart cards
- Internautas Brasileiros **perderam US\$ 120** milhões pela internet (só 2005!)

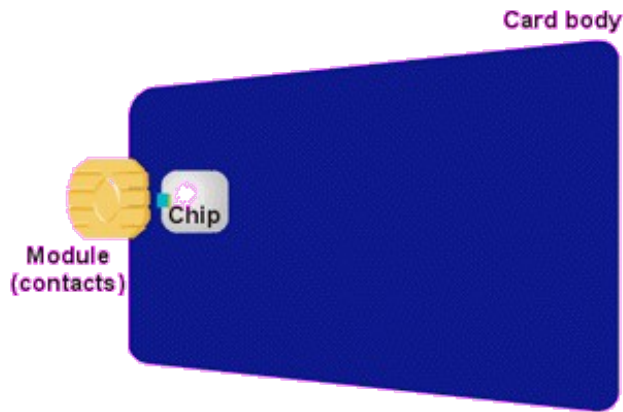


# Arquitetura de Hardware & Software



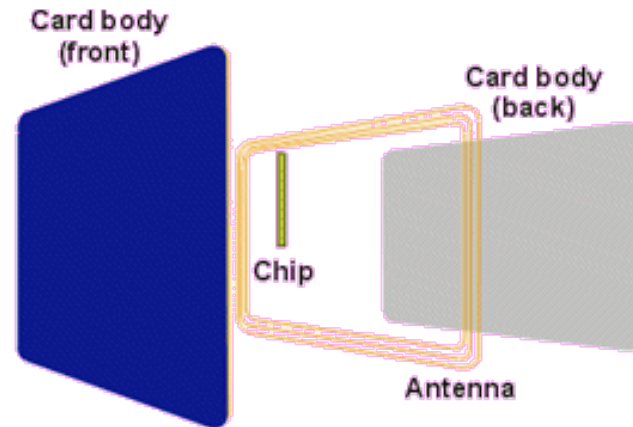
# Tipos de cartões inteligentes

- Diferenças entre funcionalidades e preço



Source: Gemplus - All About Smart Cards

Contato físico

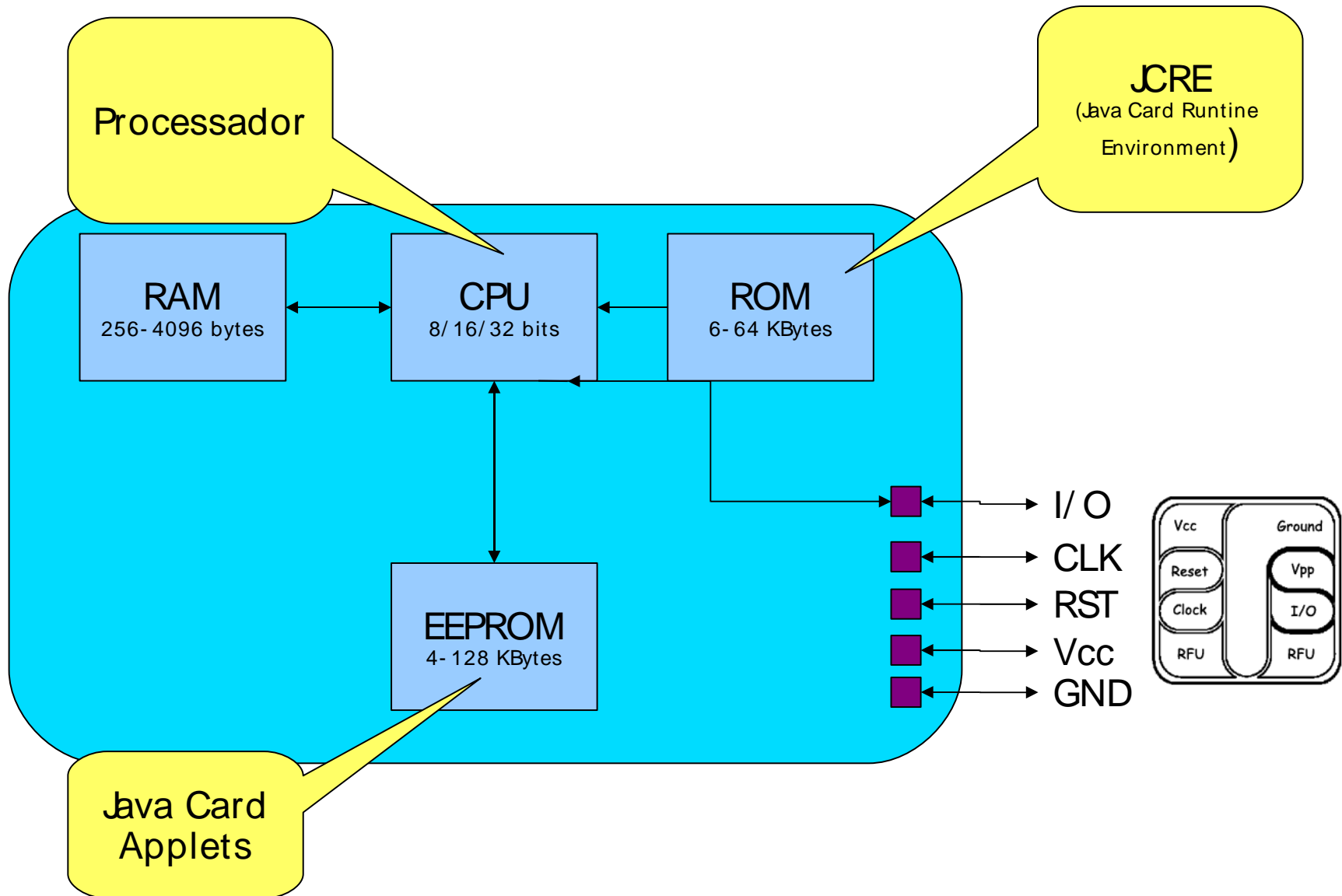


Source: Gemplus - All About Smart Cards

Rádio frequência  
Sem contato (*contactless*)

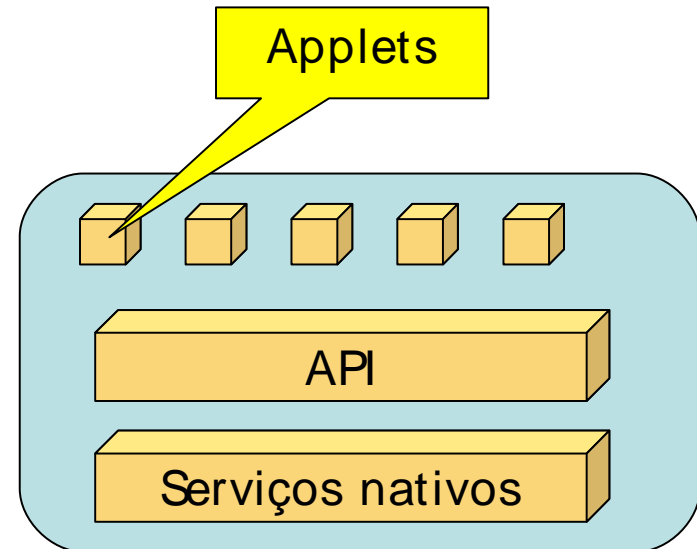


# Arquitetura (smart card de contato)



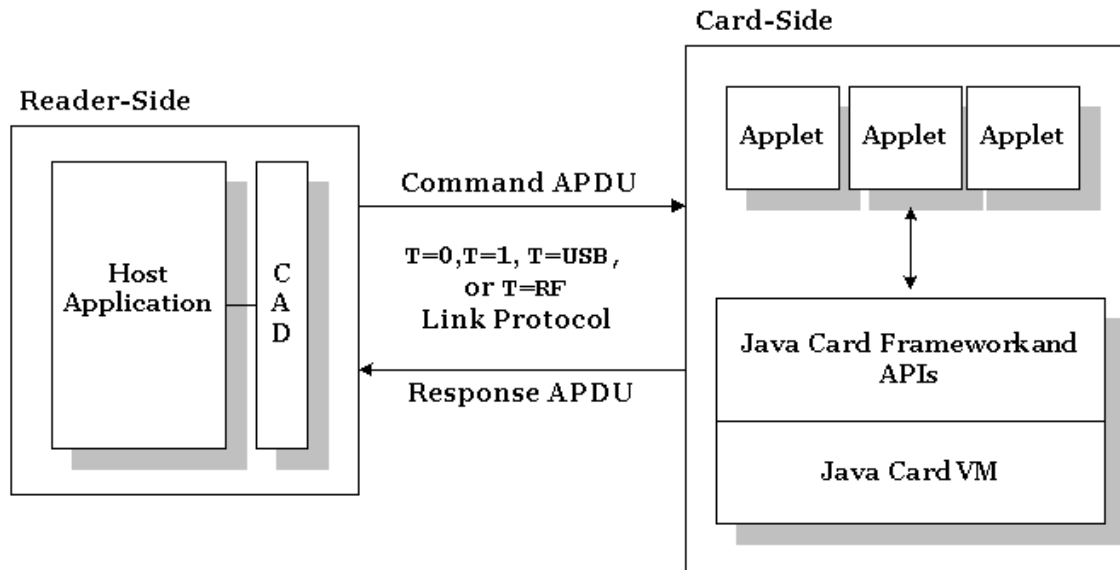
# Cartões multi- aplicação

- Suporte a mais de um aplicativo (applet)
  - Java Card
  - MULTOS
  - Smart Card for Windows



# Controle de comunicação - APDU

- Application Protocol Data Units
- Comunicação *half-duplex*
- T= 0 (envio byte- a- byte)
- T= 1 (envio de blocos de bytes)



# Estrutura APDU

## Comando-APDU

Cabeçalho obrigatório				opcional		
CLA	INS	P1	P2	Lc	Data Field	Le

## Resposta-APDU

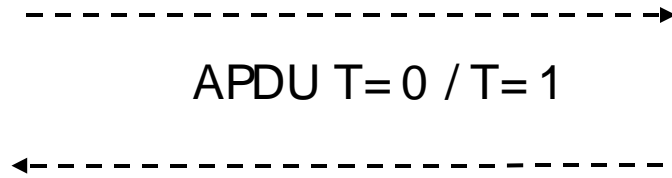
Corpo opcional	obrigatório	
Data Field	SW1	SW2



# Casos de APDU

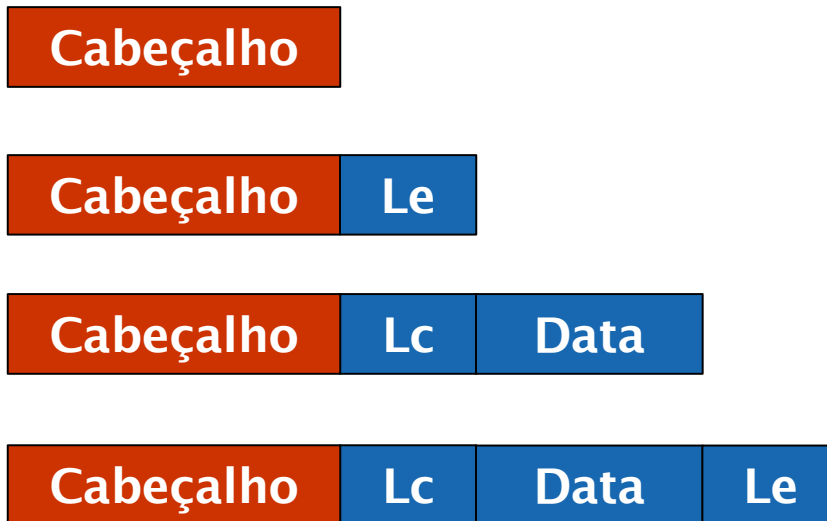


host

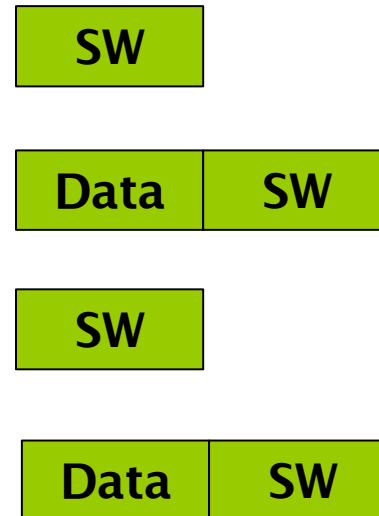


Device

## Comando



## Resposta



# Exemplos APDU

- **Get Challenge**

>> 00 84 00 00 08

<< 05 7F 16 4E E5 7E 37 F9 90 00

- **Delete Master File**

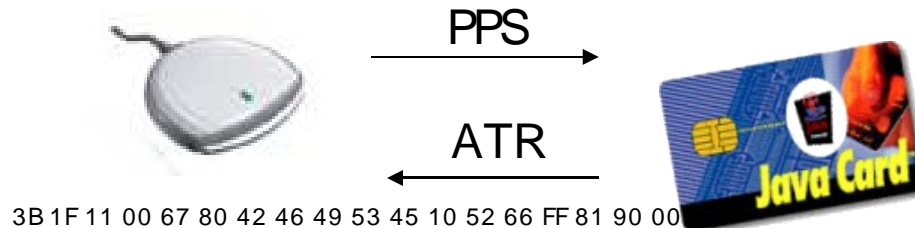
>> 80 E4 00 00 02 3F 00 (FORMAT C

<< 90 00



# ATR (Answer To Reset)

- Cadeia de caracteres com informações do próprio smart card
- Obtido quando o smart card é conectado
- Exemplos
  - 3B 23 00 35 13 FF (Schlumberger MicroPayflex)
  - 3B 1F 11 00 67 80 42 46 49 53 45 10 52 66 FF 81 90 00 (Nokia branded SC)
  - 3B 1F 94 00 6A 01 38 46 49 53 45 10 8C 02 FF 07 90 00 (GSM- SIM Saunalahti)



# Análise de um ATR

**3B A7 00 40 18 80 65 A2 08 01 01 52**

ATR\_analysis '3B A7 00 40 18 80 65 A2 08 01 01 52'

ATR: 3B A7 00 40 18 80 65 A2 08 01 01 52

+ TS = 3B -- > **Direct Convention**

+ T0 = A7, Y(1): 1010, K: 7 (historical bytes)

TB(1) = 00 -- > **Programming Param P: 0, I: 0**

TD(1) = 40 -- > **Y(i+1) = 0100, Protocol T = 0**

TC(2) = 18 -- > **Work waiting time: 960 x 24 x (Fi/ F)**

+ **Historical bytes: 80 65 A2 08 01 01 52**

Possibly identified card:

**3B A7 00 40 18 80 65 A2 08 01 01 52**

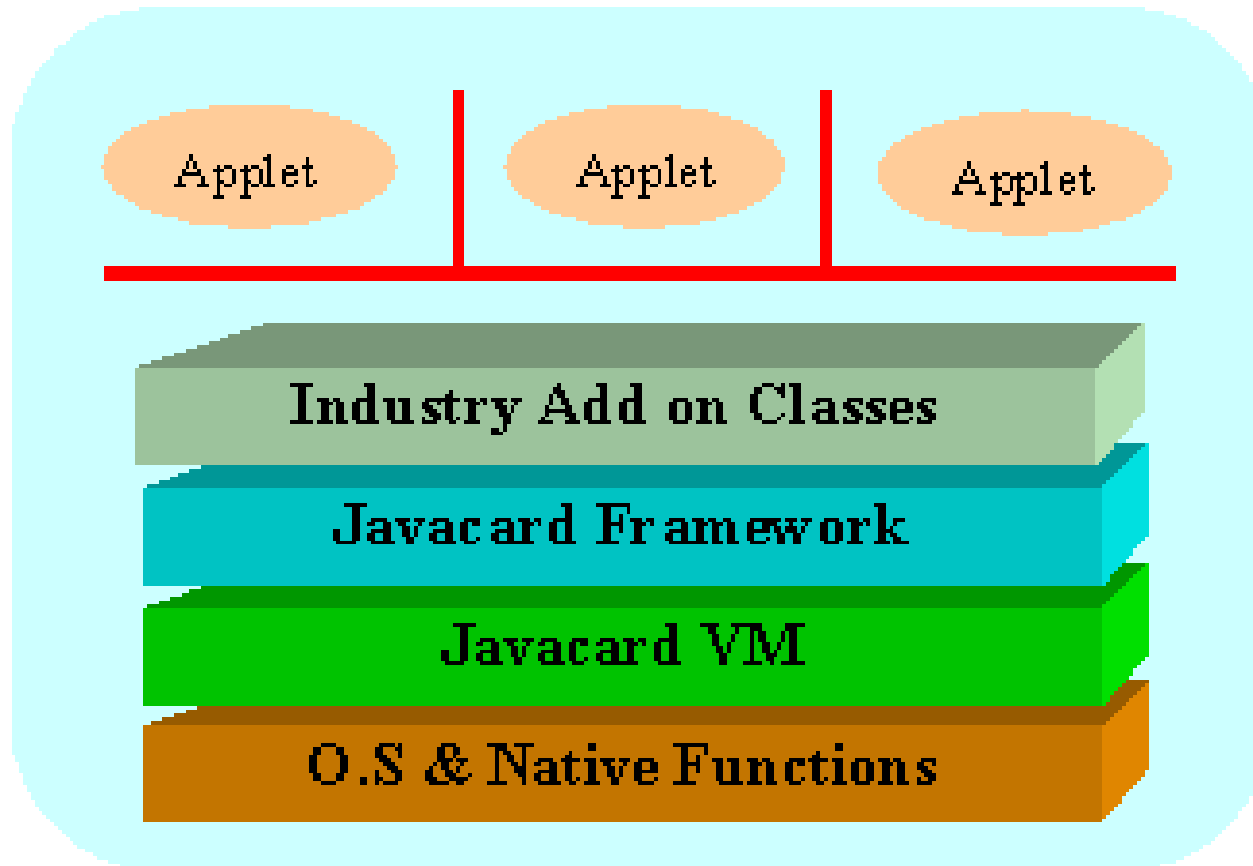
**Gemplus GPK8000**



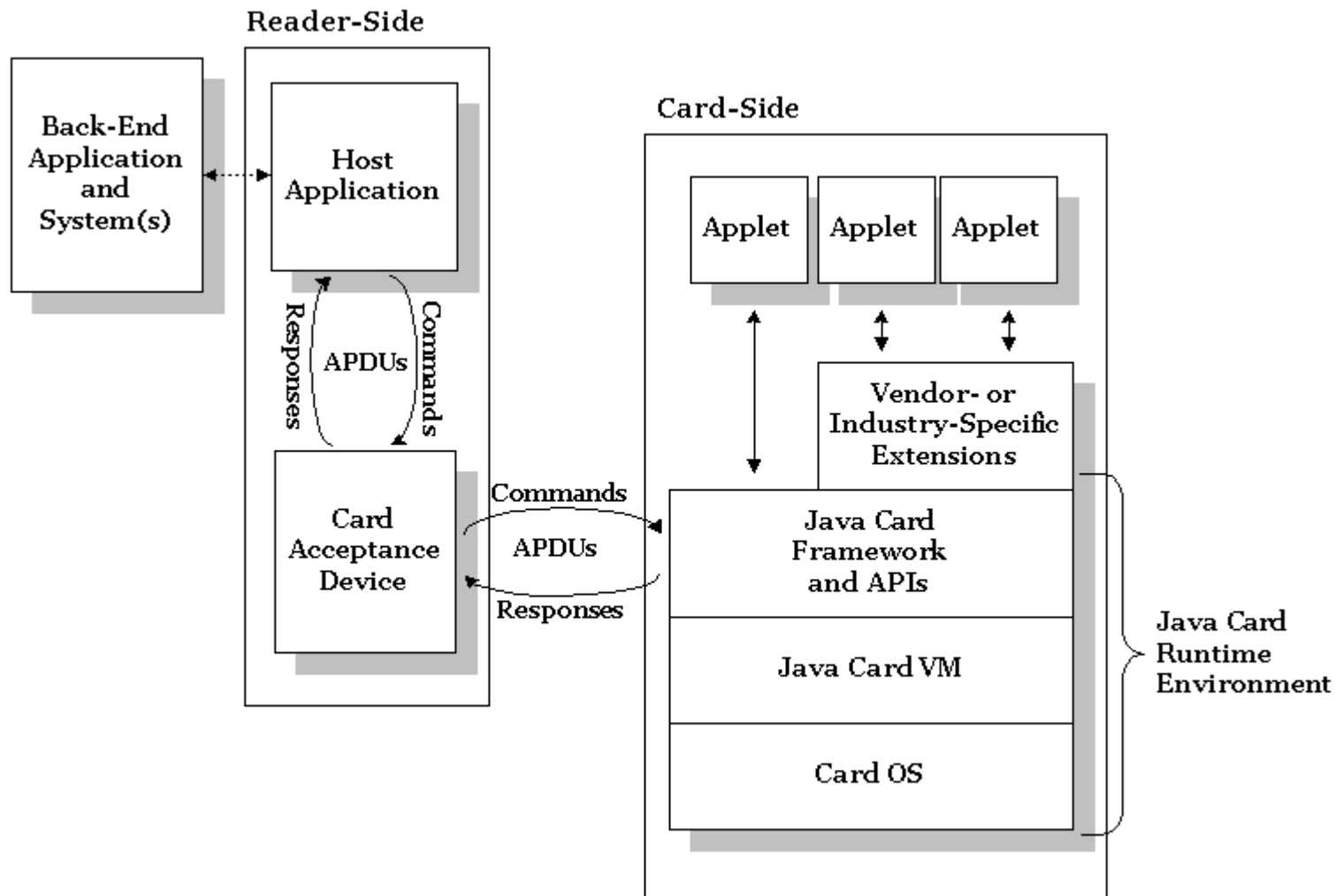
# Java Card



# Plataforma Java Card



# Aplicação com smart card



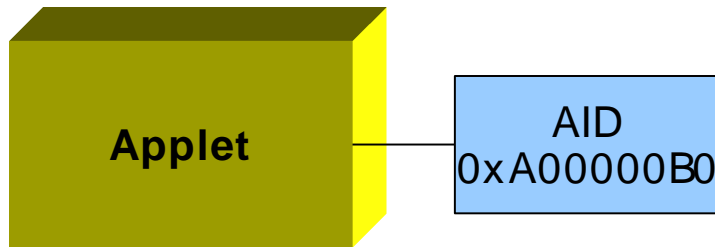
# Subset

- Tipos primitivos “grandes”: long, double, float;
- Caracteres e strings
- Arrays multidimensionais
- Carga dinâmica de classes
- Security manager
- Garbage collector
- Serialização e clone de objetos

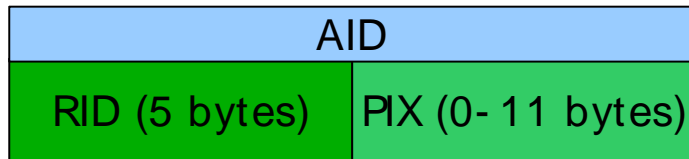


# Java Card Applets

- Pequenos aplicativos que executam em Java Cards (“Cardlets”)

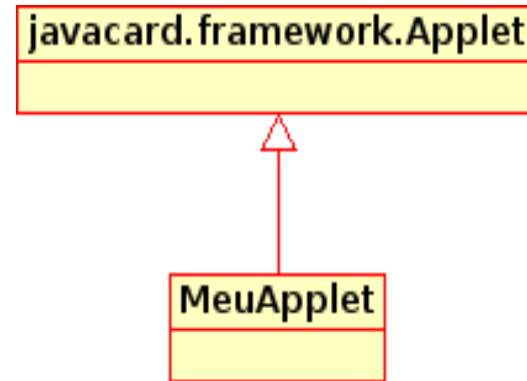


Todo applet deve conter um AID (Application Identifier)



Resource Identifier

Proprietary Identifier Extension



Todo applet deve *estender* javacard.framework.Applet



# Framework

- javacard.framework
  - Abstrações de Applet, PIN, APDU, System e Util
- javacardx.framework
  - Implementação de rotinas ISO 7816-4 (file system)
- javacardx.crypto
  - Suporte a criptografia



# Ferramentas de desenvolvimento

- Sun Java Card Development Kit (gratuito)  
[http://java.sun.com/products/javacard/dev\\_kit.html](http://java.sun.com/products/javacard/dev_kit.html)
- Axalto VIEWS (evaluation)  
<http://www.simagine.org> (requer login)
- JCOP (Plug eclipse)  
<http://www.zurich.ibm.com/jcop/news/news.html>
- JCardExpress (Projeto opensource brasileiro)  
<http://jcardexpress.dev.java.net>
- Outras  
<http://www.igormedeiros.com.br/sdk>



# Desenvolvendo um applet HelloWorld

## Definição do CLA e das INStruções

```
// CLA Byte
final static byte HELLO_CLA = (byte) 0xB0;

// Verify PIN
final static byte INS_HELLO = (byte) 0x20;
```



# Desenvolvendo um applet HelloWorld

Implementação do método  
*install*

```
public static void install(byte[] bArray, short
bOffset, byte bLength) {

    (new HelloWorldJC()).register(

        bArray,

        (short) (bOffset + 1),

        bArray[bOffset]);

}
```



# Desenvolvendo um applet HelloWorld

Implementação do método

*process*

```
// processa o comando APDU
public void process(APDU apdu) {

    byte[] buffer = apdu.getBuffer();

    // Select the appropriate instruction (Byte INS)
    switch (buffer[ISO7816.OFFSET_INS]) {
        case INS_HELLO :
            getHello(apdu);
            return;
        default :
            ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
    }
}
```



# Desenvolvendo um applet HelloWorld

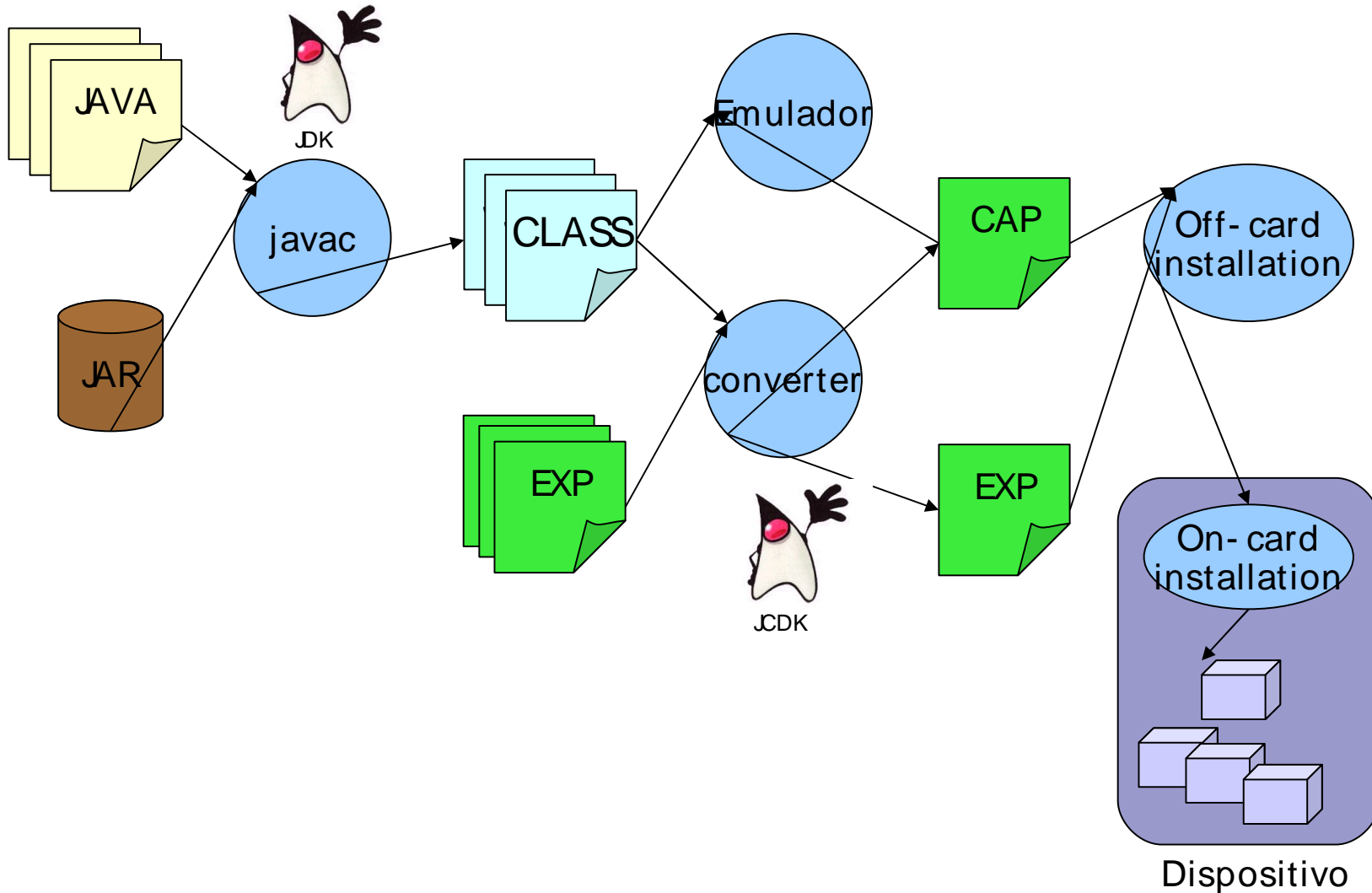
## Implementação do método

### *process*

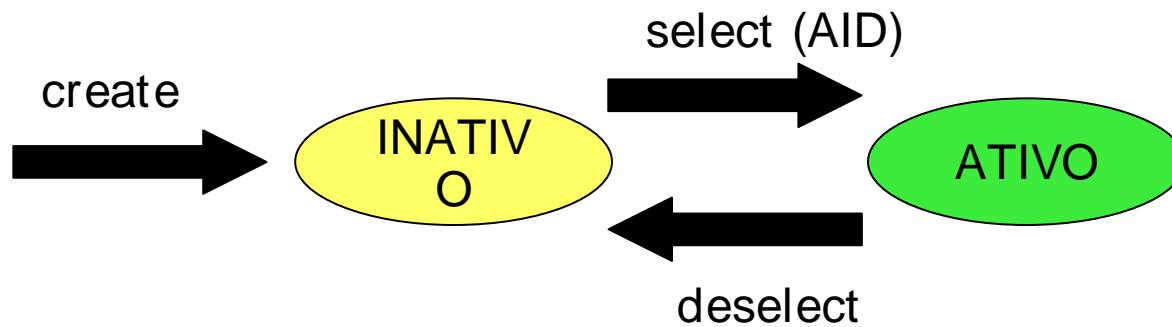
```
private void getHello(APDU apdu) {  
  
    // cadeia de bytes com a mensagem: "hello world"  
  
    byte[] hello = {'h','e','l','l','o','  
, 'w','o','r','l','d'};  
  
    // informa ao JCRE que será enviado uma resposta  
    apdu.setOutgoing();  
  
    short totalBytes = (short) hello.length;  
  
    // informa ao JCRE o tamanho da mensagem em bytes  
    apdu.setOutgoingLength(totalBytes);  
  
    // envia a mensgem para o host  
    apdu.sendBytesLong(hello, (short) 0, (short) hello.length);  
  
}
```



# Processo de desenvolvimento



# Ciclo de vida



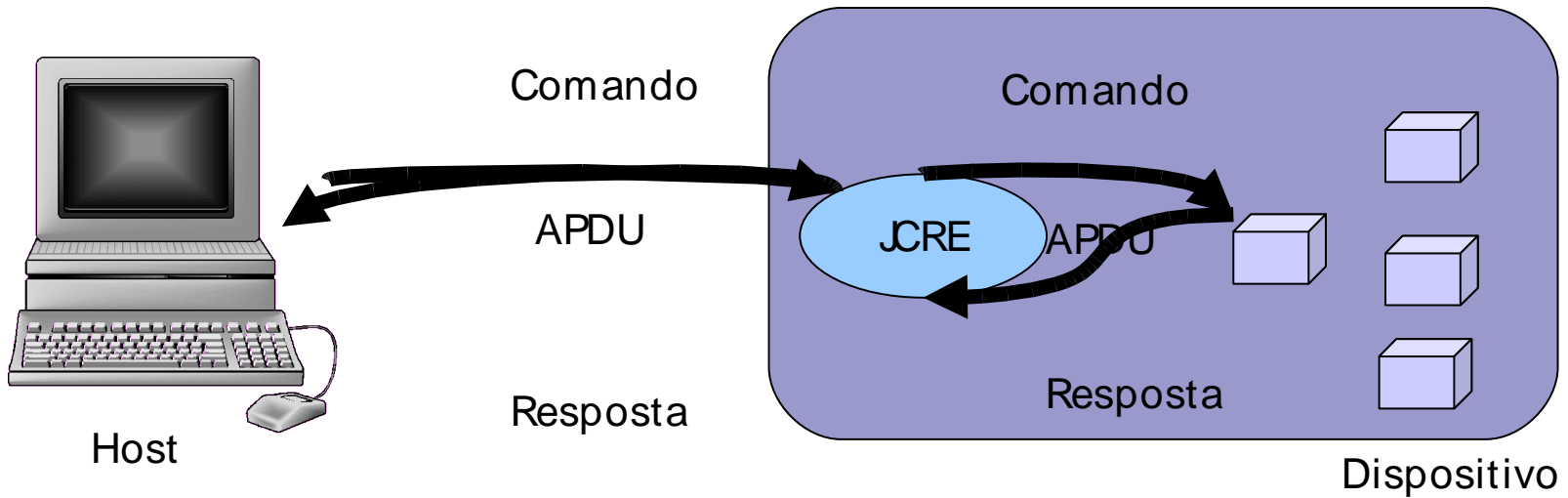
processa  
o comando

**javacard.framework.Applet**

```
+ install()  
+ select()  
+ deselect()  
+ process()
```



# Envio de APDU para Applets

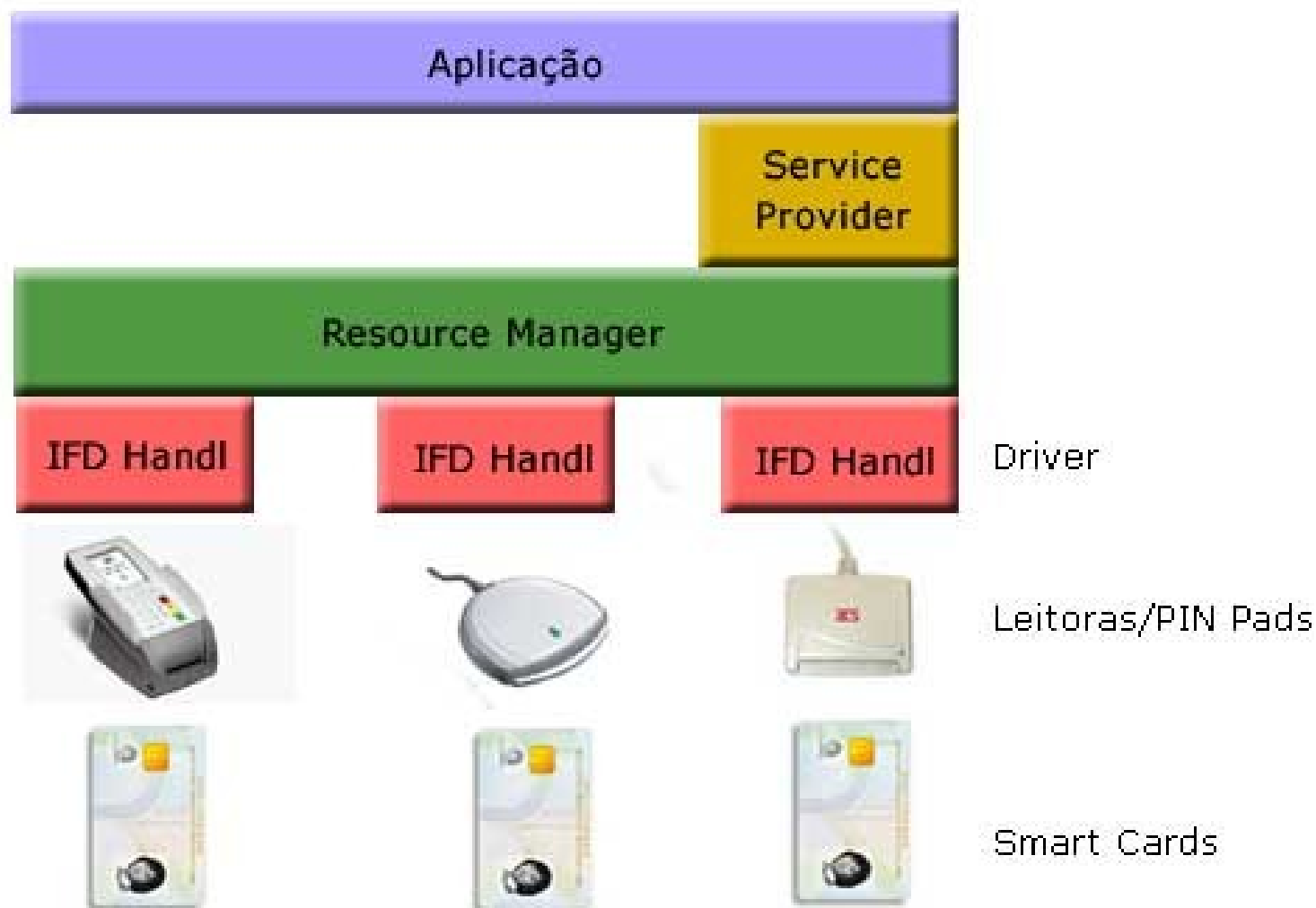


# PC/ SC

- Arquitetura para conectar um PC (desktop, laptop, etc) a um Smart Card
- Inclui um *Resource Manager* – controle da leitora e smart card para o nível de aplicação
- Trata eventos como “cartão inserido”, “cartão removido”
- Obtém status da leitora e do cartão
- Transmissão e recebimento de APDU



# PC/ SC



# Mustang: Novo Smart Card I/ O

- “Mustang”: Java 6!
- JSR 268: Smart Card I/ O
- JCE Provider (Wrapper JNI)
- API que provê funcionalidades PC/ SC e APDU em aplicativos Java: [javax.smartcardio](http://javax.smartcardio)



# Mustang: Novo Smart Card I/ O

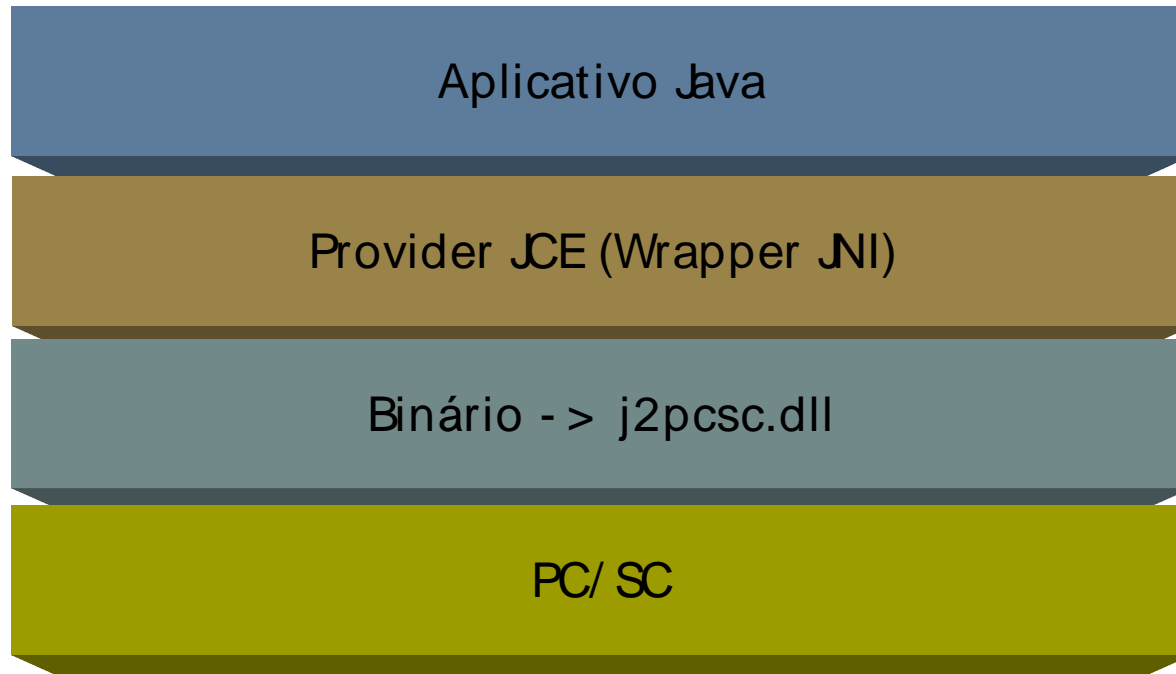
## Funcionalidades

- Estabelecer conexão com o smart card
- Listar leitoras instaladas no S.O.
- Obter status da leitora (ex.: presença do cartão)
- Transmitir APDUs e receber resultado



# Mustang: Novo Smart Card I/ O

## Camadas



# Mustang: Novo Smart Card I/O

## lib\security\java.security

```
#  
# List of providers and their preference orders (see above):  
#  
security.provider.1=sun.security.provider.Sun  
security.provider.2=sun.security.rsa.SunRsaSign  
security.provider.3=com.sun.net.ssl.internal.ssl.Provider  
security.provider.4=com.sun.crypto.provider.SunJCE  
security.provider.5=sun.security.jgss.SunProvider  
security.provider.6=com.sun.security.sasl.Provider  
security.provider.7=org.jcp.xml.dsig.internal.dom.XMLDSigRI  
security.provider.8=sun.security.smartcardio.SunPCSC  
security.provider.9=sun.security.mscaapi.SunMSCAPI
```



# Exemplo

```
public class TesteSmarCardIO {  
  
    public static void main(String[] args) {  
  
        // show the list of available terminals  
        TerminalFactory factory = TerminalFactory.getDefault();  
  
        List<CardTerminal> terminals = factory.terminals();  
        System.out.println("Terminals: " + terminals);  
  
        // get the first terminal  
        CardTerminal terminal = terminals.get(0);  
    }  
}
```



# Exemplo

```
try {
    // establish a connection with the card
    Card card = terminal.connect("T=0");

    System.out.println("card: " + card);

    CardChannel channel = card.getBasicChannel();

    byte[] comando = {(byte) 0x00, (byte) 0x84, (byte) 0x00,
        (byte) 0x00, (byte) 0x08};
    ResponseAPDU r = channel.transmit(new CommandAPDU(comando));

    System.out.println("ATR: " + card.getATR());
    System.out.println("Protocolo: " + card.getProtocol());
    System.out.println("response: " + r.toString());
    // disconnect
    card.disconnect(false);
} catch (CardException e) {
    e.printStackTrace();
}
}
```



# Aplicativo Java 6 PC/ SC

**ReaderAnalyzer Tester** [Minimizar] [Maximizar] [Fechar]

PC/SC Opções Ajuda

**PC/SC**

Leitora: SCM Microsystems Inc. SCR33x USB Sma... [Desconectar]

Estado da leitora: Alterado,unknown

Presença do cartão inteligente: present

Protocolo de Transporte: T1

Cadeia ATR: 3B B7 18 00 C0 3E 31 FE 65 53 50 4B 32 34 90 00 25

Tipo do cartão inteligente: Assíncrono 7816

**Comando APDU**

CLA	INS	P1	P2	LC	Data	LE	
00	a4	04	00	02	3f00	00	<input type="checkbox"/> Modo ASCII

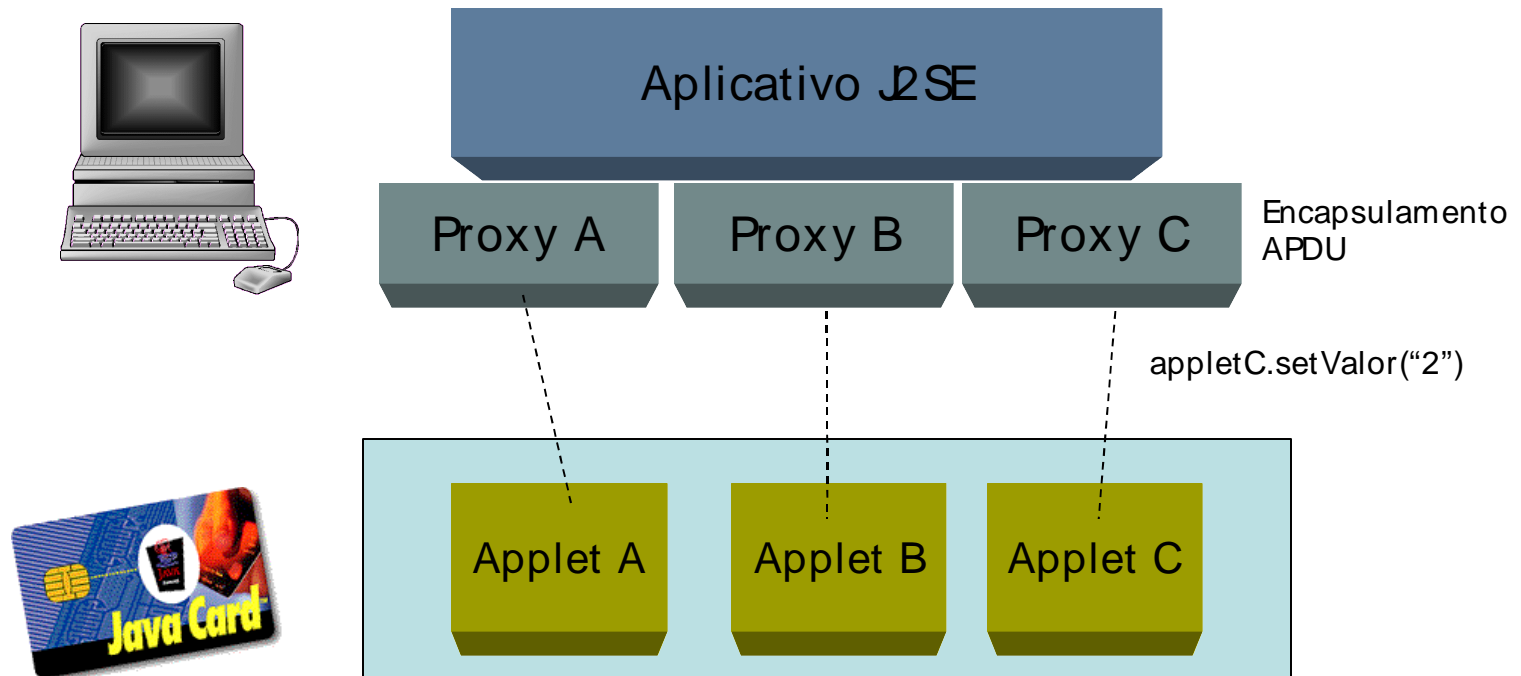
Pré-definidos [Desce] [Envia]

Cadeia ATR: 3B B7 18 00 C0 3E 31 FE 65 53 50 4B 32 34 90 00 25  
Tipo do cartão inteligente (baseado na cadeia ATR): Assíncrono 7816  
>> 8: 00A40000023F0000  
<< 2: 62 84  
>> 8: 00A40001023F0000



# Frameworks off card complexos

- OCF (OpenCard Framework)
- Java Card RMI



# Futuro próximo

- Java Card 3.0
- Smart cards com limitações menores de hardware



# Perguntas

An exception 06 has occurred at 0028:C11B3ADC in \xD DiskTSD(03) + 00001660. This was called from 0028:C11B40C8 in \xD voltrack(04) + 00000000. It may be possible to continue normally.

- \* Press any key to attempt to continue.
- \* Press CTRL+ALT+RESET to restart your computer. You will lose any unsaved information in all applications.

Press any key to continue

# Conclusões

- Smart card é realidade no Brasil, mão-de-obra não!
- Java Card promove uma série de benefícios que atraem indústria e desenvolvedores
- PC/ SC é a arquitetura mais sólida e utilizada para conectar PCs à Smart Cards
- Um framework para host deve incluir conceitos de inteoperabilidade, segurança, serviços específicos para cada cartão, etc
- A nova API Smart Card I/ O: Simplicidade (API leve) e Versatilidade (Provider de serviços JCE)

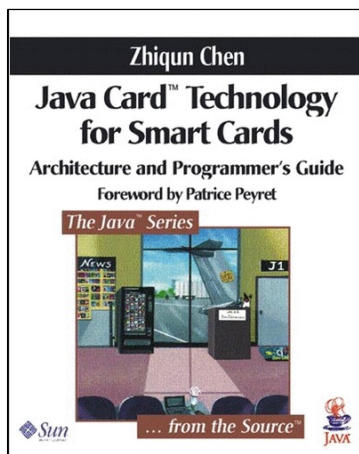


# Referências

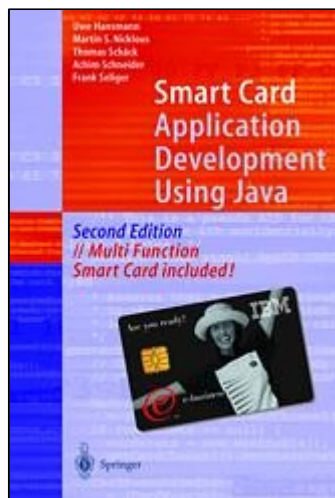
- Java Card (site oficial)  
<http://java.sun.com/products/javacard>
- JSR 268  
<http://jcp.org/aboutJava/communityprocess/edr/jsr268/index.html>
- Portal Java Card Brasil  
<http://www.javacard.com.br>
- PC/ SC WorkGroup  
<http://www.pcscworkgroup.com>
- JPC/ SC | PC/ SC Lite  
<http://www.musclecard.com/middle.html>
- ITI Brasil  
<http://www.iti.gov.br>
- OpenCard Framework  
<http://www.opencard.org>



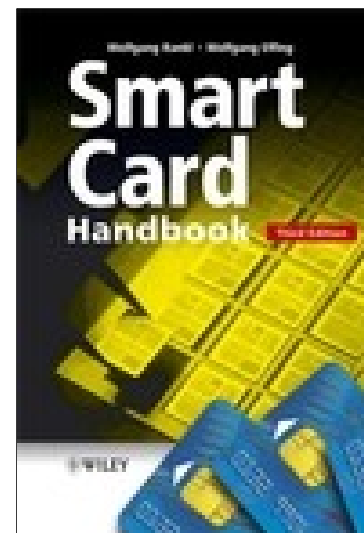
# Bibliografia



ISBN: 0201703297



ISBN:  
3540658297



ISBN: 0-470-  
85668-8





# OBRIGADO!

## Igor Medeiros

[igor@igormedeiros.com.br](mailto:igor@igormedeiros.com.br)

[www.igormedeiros.com.br](http://www.igormedeiros.com.br)

MSN: javacardman

