01. Úvod do programování čipových karet Java Card

1. Ve Windows spustit NetBeans 8.2. (Novější Netbeans neobsahují potřebné komponenty. Ani pro Linux podpora není.)
2. V učebně není potřeba, ale na vlastním počítači je potřeba mít Java SDK 1.8 a Netbeans 8.2 pro Javu (s novějšími to nechodí). Potom je potřeba v Netbeans provést (poprvé pro uživatele):
    - Tools - Plugins,
    - Available Plugins, Search: java card,
    - zaškrtnout všechny 3 pluginy, Install, potvrdit licence
    - Po instalaci pluginů povolit restart IDE
3. File - New Project
    - Kategorie Java Card, Classic Applet Project,
    - můžeme nechat výchozí hodnoty, Next,
    - Finish Teď máme vytvořen základ pro vlastní Java Card applet.

Základní struktura Java Card appletu

```
package pack1;
import javacard.framework.*;
public class Applet1 extends Applet {
  // declare your variables here
  byte [] array;
  public static void
  install(byte[] bArray, short bOffset, byte bLength) { // called on installation
    new Applet1();
  }
  protected Applet1() { // applet constructor
    // create your variables here
    array = new byte[128];
    register();
  }
  public void process(APDU apdu) {
    // command processing here
  }
}
```

- install() – je zavolán frameworkem při instalaci appletu, to je první, co se vykoná. Zde je vhodné zavolat vlastní konstruktor.

- konstruktor appletu, např. Applet1() – zde vytvoříme instance všech objektů (polí apod.), které budeme potřebovat pro běh appletu. Na závěr zavoláme register(), což zaregistruje naši instanci v rámci frameworku na kartě.
- process() – je zavolán, když nám přijde jakýkoli příkaz formou APDU (první bude SELECT našeho appletu). Zde naprogramujeme požadované funkce (příp. voláním dalších našich metod, s použitím našich jiných objektů apod.)

Požadavky na Váš applet

Úkoly:

- Vytvořte Java Card Applet, který bude podporovat následující instrukce:
  - INS=0x01 přijme data, maximálně 20 bajtů
  - INS=0x02 odešle data zapsaná pomocí 0x01, maximálně tolik bajtů, kolik je požadováno (Le) a kolik bylo předtím přijato
  - INS=0x00 vrátí Vaše jméno zakódované v ASCII
- Svůj applet otestujte v simulátoru Java karty (v NetBeans ve Windows)
  - Otestujte předpokládanou správnou funkci
  - Ošetřete a otestujte nesprávnou hodnotu CLA – má vracet ISO7816.SW_CLA_NOT_SUPPORTED (0x6E00). Viz struktura Class byte (Odkaz na cardwerk.com) (http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-4_5_basic_organizations.aspx#chap5_4_1)
  - Ošetřete a otestujte nesprávnou hodnotu INS – má vracet ISO7816.SW_INS_NOT_SUPPORTED (0x6D00)
  - Ošetřete a otestujte instrukci INS=0x01 s příliš dlouhým tělem – má vracet ISO7816.SW_WRONG_LENGTH (0x6700)
  - Ošetřete a otestujte instrukci INS=0x02 s nesprávnou požadovanou délkou – má vracet SW 0x6Cxx, kde xx je správný počet bajtů, které jsou uloženy na kartě. Můžete použít ISO7816.SW_CORRECT_LENGTH_00 (0x6C00) sečtenou se správnou hodnotou délky.

Komunikace karty s terminálem

Tabulka 1. Command APDU

Hlavička				**Tělo**		
(povinná)				(nepovinné)		
CLA	INS	P1	P2	Lc	data	Le

Používejte CLA 0x80 pro vlastní příkazy (0x00 pro standardní, tedy SELECT, viz ISO-7816-4) Byte INS si můžete zvolit, v appletu pak testovat Parametry P1 a P2 můžete používat dle libosti Přenášíte-li data do karty, bude následovat byte Lc – délka dat Případně data Očekáváte-li odpověď, bude následovat byte Le – délka očekáváné odpovědi. Nečekáte-li nic, bajt se neposílá – ale simulátor jej v tomto případě přesto potřebuje – pošlete 0x7f.

Tabulka 2. Response APDU

Tělo	Zápatí	
(nepovinné)	(povinné)	
data	SW1	SW2

Uvedené kombinace nepovinných částí APDU dávají dohromady 4 případy výměny příkaz – odpověď:

Tabulka 3. Výměna příkaz-odpověď

Případ	Command APDU							———	Response APDU		
1	CLA	INS	P1	P2						SW1	SW2
2	CLA	INS	P1	P2	Le				data	SW1	SW2
3	CLA	INS	P1	P2	Lc	data				SW1	SW2
4	CLA	INS	P1	P2	Lc	data	Le		data	SW1	SW2

Programování komunikace v kartě

Komunikaci vždy zahajuje terminál, nám přijde APDU jako parametr metody process. Hlavičku a poté data lze zpřístupnit v APDU bufferu

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

zpřístupní nejprve hlavičku APDU, data ještě nemůžeme číst ani odesílat

K elementům hlavičky CLA, INS, P1 a P2 máme přístup v APDU bufferu na offsetech 0, 1, 2, 3. Offsety mají také symbolická jména, např. ISO7816.OFFSET_CLA = 0. Na offsetu 4 se nachází bajt, jehož význam je buď Lc, nebo Le, podle použitého případu APDU.

```
byte ins = buf[ISO7816.OFFSET_INS];
switch(ins) {
    case 0x01: ...
```

Příjem dat

Přepneme objekt APDU do stavu příjmu dat, a získáme přijatá data (pro jednoduchost předpokládáme, že se všechna najednou vejdou do APDU bufferu, jinak bychom to museli zpracovávat složitěji)

```
short len = 0;
...
```

```
len = apdu.setIncomingAndReceive(); // actually received data length stored to 'len'
```

Nyní můžeme data jakkoli zpracovávat, jsou v APDU bufferu od pozice ISO7816.OFFSET_CDATA. Například je můžeme zkopírovat do pole 'array' od začátku (pozice 0):

```
Util.arrayCopy(buf, ISO7816.OFFSET_CDATA, array, (short)0, len);
```

nebo

```
Util.arrayCopyNonAtomic(buf, ISO7816.OFFSET_CDATA, array, (short)0, len);
```

Odesílání dat

Pro odeslání dat odpovědi musíme použít sekvenci

- `setOutgoing()` – vrací short počet očekávaných bajtů odpovědi (Le)
- `setOutgoingLength(short length)` – nastavíme počet bajtů, které hodláme odeslat (počet se může lišit od Le, je na nás, kolik pošleme)
- `sendBytes()` nebo `sendBytesLong()` podle toho, jestli máme odpověď připravenou už v APDU bufferu, nebo jestli chceme odesílat data z jiného pole např.

```
len = apdu.setOutgoing();
if (len > array.length) len = (short)array.length;
apdu.setOutgoingLength(len);
apdu.sendBytesLong(array, (short)0, len);
```

Odeslání stavového slova

Jako povinný prvek odpovědi karta odesílá stavové slovo (status word, bajty SW1, SW2). K tomu slouží metoda `ISOException.throwIt(short status_word)`, jejímž zavoláním je odpověď karty ukončena.

Jako příklad uvedeme reakci na úplně první APDU, která nám přijde, což je SELECT [00 A4 04 00 Lc …applet AID…], na kterou nemusíme odpovídat daty, ale musíme potvrdit vybrání našeho appletu.

```
if (selectingApplet()) ISOException.throwIt(ISO7816.SW_NO_ERROR);
```

Standardizované hodnoty stavového slova najdete např. v tabulce status word (https://cardwerk.com/smart-card-standard-iso7816-4-section-5-basic-organizations/#chap5_4_5)

Spouštění a ladění v simulátoru

Applet lze spustit (Run) i ladit (Debug). Ve výchozím nastavení se automaticky pouští APDU skript (viz scripts/(aplet name).scr)

```
powerup;
// Select Applet1 //aid/18DD731C82/79
0x00 0xA4 0x04 0x00 0X06 0X18 0XDD 0X73 0X1C 0X82 0X7A 0x7F;
//Send the APDU here
//0x80 0xCA 0x00 0x00 <length> <data> 0x7F;
0x80 0x01 0x00 0x00 0x03 0xa1 0xa2 0xa3 0x7F;
0x80 0x00 0x00 0x00 0x00 0x03;
powerdown;
```

Formát APDU příkazů pro simulátor je dán požadavkem nástroje "apdutool", a mírně se liší od standardního zápisu:

- Pole Lc i Le jsou povinná
- Chcete-li Le vynechat, napište místo něj 0x7f,
- Chcete-li Lc vynechat, napište místo něj 0x00.
- Pokud Vám simulátor hlásí Connection refused, pravděpodobně je kolize s jeho předchozí spuštěnou instancí, stačí dát znovu Debug.
- V předchozí (staré) instalaci v učebně simulátor občas/často padal, a někdy pomohlo (jako nouzové řešení):
- "C:\Program Files\NetBeans 8.2\javacard\JCDK3.0.2_ConnectedEdition\bin\cjcre" -debug yes

Další zdroje

Oracle: Writing a Java Card Applet (https://www.oracle.com/technetwork/java/embedded/javacard/documentation/intro-139322.html)