ScanCmds.java

```
//-----------------------------------------------------------------------------
// ScanCmds
// Demontration of scanning the supported commands of a smart card
// using the OCF 1.2 with PC/SC to transmit the data?
// file: ScanCmds.java
// uses: ICC.java
//
// This program uses the PassThruCardService from OCF opencard.opt.util package.
// Additional informations and reference implementation: www.opencard.org.
// Testet with Towitoko Chipdrive micro on USB port and the appropriate PC/SC dri
// from Towitoko on Windows 98. Testet with Omnikey CardMan Dongle 6020 on USB po
// and the appropriate PC/SC drivers from Omnikey on Windows 2000. Testet with
// JBuilder 6 and JDK 1.3.1.
//
// Uses OCFPCSC1.DLL, OCFPCSC1.DLL, pcsc-wrapper-src.jar, base-core.jar, base-opt
// from OCF 1.2
// The content of the file "opencard.properties" must be changed to:
// OpenCard.terminals = com.ibm.opencard.terminal.pcsc10.Pcsc10CardTerminalFactor
// For understanding the mechanisms of OCF it is useful to look first into the
// SimpleOCF example.
//
// This program was tested with a GSM 11.11 Smart Card. The scan needs about
// 30 sec for scanning the available classes and instructions of a SIM.
//
// REMARKS: The PC/SC return code 0000045d will be thrown if a case 3 or case 4
//          command is scanned, because in this case the ICC expect some data
//          after the APDU header.
//          A scan of P1, P2 and P3 makes no sense because there is no unambiguou
//          returncode if such a parameter is wrong or not supported.
//
// WARNING: The usage of this programm is at your own risk. The execution of
//          this program could damage your smart card because of calling all
//          available commands and there migth be commands actived which kill
//          the card. Use only test cards to work with this programm.
//
// This source code is under GNU general public license (see www.opensource.org f
// Please send corrections and ideas for extensions to Wolfgang Rankl (www.wrankl
// Copyright 2003-2004 by Wolfgang Rankl, Munich
//
//  2. Sept. 2004 - V 5, rw: migrate to a Swing based GUI, add min/max value for
//                           ans instruction byte, add not scanned classes list,
//                           add Commands.java for decoding of smart card command
//                           add GPL statement,remark to 0000045d problems,
//                           division between user interface and smart card funct
//                           4th published version
// 15. Dez.  2003 - V 4, rw: minor clarifications, add GPL statement
//                           remark to 0000045d problems, 3rd published version
// 23. Jan.  2003 - V 3, rw: simplifications byte to int conversion, test with ne
//                           card reader, review of the source code, 2nd publishe
// 14. Oct.  2002 - V 2, rw: improved documentation, 1st published version
// 10. Oct.  2002 - V 1, rw: initial runnable version
//-----------------------------------------------------------------------------

package scpack;

import java.awt.*;
import java.awt.event.*;
import java.text.*;
import java.util.*;
import javax.swing.*;
```

```
public class ScanCmds extends JFrame implements ActionListener {
  static final boolean DEBUG = false;     // true: show the APDUs during scanning
  static final int FIRST_BYTE_ATR = 0;   // index to the first byte of the ATR

  static final int CLA_MIN = 0xA0;     // minimum class byte value for scanning
  static final int CLA_MAX = 0xA9;     // maximum class byte value for scanning
  static final int INS_MIN = 0x00;     // minimum instruction byte value for scann
  static final int INS_MAX = 0x90;     // maximum instruction byte value for scann

  static final byte[] NOTSCAN_INS = {(byte)0x01, (byte)0x03};  // not scanned ins

  public static JTextArea ta = new JTextArea(15, 100);


  /** build the UI with all the components
  */
  ScanCmds() {
    //----- create text area with scroll pane
    ta.setTabSize(4);
    ta.setFont(new Font("Monospaced", 0, 12));      // Courier, 12 point size
    ta.setBackground(new Color(  0,   0,   0));     // black blue
    ta.setForeground(new Color(200, 255, 100));     // yellowgreen
    getContentPane().add(new JScrollPane(ta), BorderLayout.CENTER);
    //----- create panel with buttons
    JPanel buttonPanel = new JPanel(new GridLayout(20, 1));
    JButton button1 = new JButton("Exit");
    button1.addActionListener(this);
    buttonPanel.add(button1);
    getContentPane().add(buttonPanel, BorderLayout.EAST);
  } // ScanCmds

  /** dispatcher which the main parts of the programm
   *  @param event event for the pressed button
   */
  public void actionPerformed(ActionEvent event) {
    boolean loopbreak;
    byte[] atr;
    int result;
    long starttime, deltatime, noofwrites;

    showText("\n-----------------------------------------\n");
    String cmd = event.getActionCommand();

    if (cmd.equals("Exit")) {
      showText("exit button pressed\n");
      System.exit(0);
    } // if
  } // actionPerformed


  /** show text in the text window
   *  @param text text string to show on the display
   */
  public static void showText(String text) {
    ta.setText(ta.getText() + text);
  } // showText


  public static String IntTo2CharHexString(int x) {
    // converts a integer to the format "xx", x = hex digit
    String s;
    s = Integer.toHexString(x).toUpperCase();
    if (s.length() == 1) s = "0" + s;
    return s;
```

```
  } // IntTo2CharHexString


/** format a ATR
 *  @param atr ATR byte string with the raw ATR data in hex
 *  @return ATR string with the ATR in hex
 */
public static String formatATR(byte[] atr) {
  int n, x;
  String w = new String();
  String s = new String();

  w = "ATR: ";
  for (n = 0; n < atr.length; n++) {
    x = (int) (0x000000FF & atr[n]);
    w = Integer.toHexString(x).toUpperCase();
    if (w.length()== 1) w = "0" + w;
    s = s + w + " ";
  }  // for
  return s;
} // formatATR

  /** format a command APDU
 *  @param CmdAPDU  byte string with the raw command APDU
 *  @param LenCmdAPDU length of the raw command APDU bytw string
 *  @return  formated string with the command APDU
 */
public static String formatCmdAPDU(byte[] CmdAPDU, int LenCmdAPDU) {
  int n, x;
  String w = new String();
  String s = new String();

  s = "IFD->ICC: ";
  for (n = 0; n < LenCmdAPDU; n++) {
    x = (int) (0x000000FF & CmdAPDU[n]);
    w = Integer.toHexString(x).toUpperCase();
    if (w.length() == 1) w = "0" + w;
    s = s + w + " ";
  }  // for
  return s;
 } // formatCmdAPDU


/** format a response APDU
 *  @param RspAPDU byte string with the raw response APDU
 *  @param LenRspAPDU length of the raw response APDU bytw string
 *  @return formated string with the response APDU
 */
public static String formatRspAPDU(byte[] RspAPDU, int LenRspAPDU) {
  int n, x;
  String w = new String();
  String s = new String();

  s = "ICC->IFD: ";
  for (n = 0; n < LenRspAPDU; n++) {
    x = (int) (0x000000FF & RspAPDU[n]);
    w = Integer.toHexString(x).toUpperCase();
    if (w.length()== 1) w = "0" + w;
    s = s + w + " ";
  }  // for
  return s;
} // formatRspAPDU
```

```
  public static void main(String [] args) {
    boolean scan;
    byte[] atr;
    int x, cla, ins, result;

    //--- build the GUI and set the parameters
    ScanCmds scmds = new ScanCmds();
    scmds.setLocation(0, 0);
    scmds.pack();
    scmds.setTitle("Scan Smart Card Commands");
    scmds.setVisible(true);

    showText("Start Program: ScanCmds\n\n");
    ICC icc = new ICC();

    try {
      showText("activate smart card\n\n");
      icc.start();
      atr = icc.start();
      if (atr[FIRST_BYTE_ATR] != 0) {
          // no error occur, a smart card is in the terminal, ATR received
        showText ("ATR [hex]:  " + formatATR(atr) + "\n");

        // this is the main loop for scanning all the commands within the presele
        showText("\nstart CLA and INS scan\n");
        for (cla = CLA_MIN; cla <= CLA_MAX; cla++) {  // try all classes
          ins = INS_MIN;
          showText("test cla=" + IntTo2CharHexString(cla) + " ");
          showText("ins=" + IntTo2CharHexString(ins) + "\n");
          result = icc.sendCmd(cla, ins);   // set class and instruction
          if (result != ICC.CLA_NOT_FOUND) {
            // found a supported class try now the instructions range with this c
            showText("\t\t\t\t\tfound cla=" + IntTo2CharHexString(cla) + "\n");
            for (ins = INS_MIN; ins <= INS_MAX; ins++) {
              scan = true;
              for (x = 0; x < NOTSCAN_INS.length; x++) {
                if (NOTSCAN_INS[x] == ins) scan = false;
              } // for
              if (scan == true) {
                showText("test cla=" + IntTo2CharHexString(cla) + " ");
                showText("ins=" + IntTo2CharHexString(ins) + "\n");
                result = icc.sendCmd(cla, ins);   // set class and instruction
                if (result == ICC.INS_FOUND) {
                  // found a supported instruction
                  showText("\t\t\t\t\t\t\tfound ins=" + IntTo2CharHexString(ins)
                  showText(Commands.decodeCmd(cla, ins) + "\n");
                } // if INS found
              } // if scan
            } // for INS loop
          } // if CLA found
        } // for CLA loop

        showText("\ndeactivate smart card\n");
        icc.stop();
       } // if
     } // try
     catch (Exception except) {
         showText("\n\nCaught exception '" + except.getClass() + "' - " + except.
     } // catch
   } // main
} // class
//---------------------------------------------------------------------------
```

ScanCmds.java