Exercise 3a

Transactions

In this exercise, you will analyze a (partial) applet that has several problems related to the use of transactions. The applet contains functions to verify and update a PIN number, including refusing to attempt verification after a certain amount of incorrect attempts.
Carefully analyze the code (see back), identify incorrect use of transactions, explain why the section in question is problematic, and suggest ways to improve it. (*Hint*: There are at least 3 distinct transactional issues.)

Exercise 3b

Transactions, Memory Management and Multiple Channels

In this exercise, you will write a simple applet that utilizes multiple channels and different types of memory. The applet keeps track of a single short value. Upon instantiation, this value shall be set to 0 (zero). The applet provides some functions to modify this value on channel 0, and functions to read the value, as well as give some statistical information on any other channel. The applet shall also ensure that this data remains consistent in case of failure (such as a card tear event).

If the current channel is 0 (zero), the applet shall satisfy the following requirements:

1. The applet shall accept any CLA byte value.
2. If the INS byte is 0x00, the applet shall add the value of P1 to its internal value.
3. If the INS byte is 0x02, the applet shall subtract the value of P2 from its internal value.
4. If the INS byte is 0x04, the applet shall set its internal value to the product of P1 and P2.
5. The applet shall reject all other INS values by returning SW_INS_NOT_SUPPORTED.

If the current channel is not 0 (zero), the applet shall satisfy the following requirements:

1. The applet shall accept any CLA byte value.
2. If the INS byte is 0x00, the applet shall return its internal value as two byte short DATA in the RAPDU.
3. If the INS byte is 0x02, the applet shall return the amount of time the internal value has been modified since the last selection of the applet as two byte short DATA in the RAPDU.
4. If the INS byte is 0x04, the applet shall return the amount of time the internal value has been modified since the last card reset as two byte short DATA in the RAPDU.
5. If the INS byte is 0x06, the applet shall return the amount of time the internal value has been modified since the applet was installed as two byte short DATA in the RAPDU.
6. If the INS byte is 0x08, the applet shall return the internal value, as it was before the last three modifications, as two byte short DATA in the RAPDU. If the value has not been modified three times, the applet shall return SW_CONDITIONS_NOT_SATISFIED.
7. If the value of the internal value did not change as result of an operation on channel 0, the operation does not qualify as a modification.
8. The applet shall reject all other INS values by returning SW_INS_NOT_SUPPORTED.

```
public class Transactions extends Applet {
        private final byte MAX_PIN_TRIES = 3;
        private final byte MAX_PIN_SIZE = 8;
        private final byte[] PIN;
        private byte pin_size;
        private byte pin_tries;
        private final boolean[] PIN_VERIFIED;

        private Transactions(byte[] bArray, short bOffset, byte bLength) {
                // Create new byte array for PIN (persistent)
                PIN = new byte[MAX_PIN_SIZE];
                // Create new boolean array for PIN state, transient)
                PIN_VERIFIED =
                        JCSystem.makeTransientBooleanArray((short) 1,
                                JCSystem.CLEAR_ON_DESELECT);
                // skip instance AID
                bOffset += (short) (bArray[bOffset] + 1);
                // skip application privileges
                bOffset += (short) (bArray[bOffset] + 1);
                // Now we can look at the install parameters
                updatePin(bArray, (short) (bOffset + 1), bArray[bOffset]);
                pin_tries = 0;
        }

        public static void install(byte[] bArray, short bOffset, byte bLength) {
                new Transactions(bArray, bOffset, bLength).register(
                        bArray, (short) (bOffset + 1), bArray[bOffset]);
        }

        public void process(APDU apdu) {
                if (selectingApplet()) {
                        return;
                }
                byte[] buf = apdu.getBuffer();
                switch (buf[ISO7816.OFFSET_INS]) {
                case (byte) 0x00 : // verify pin
                        verifyPin(buf, ISO7816.OFFSET_CDATA,
                                (byte) apdu.setIncomingAndReceive());
                        break;
                case (byte) 0x02 : // update pin
                        checkPinStatus();
                        updatePin(buf, ISO7816.OFFSET_CDATA,
                                (byte) apdu.setIncomingAndReceive());
                        break;
                default :
                        ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
                }
        }

        private void updatePin(byte[] bArray, short bOffset, byte bLength) {
                if (bLength > MAX_PIN_SIZE) {
                        ISOException.throwIt(ISO7816.SW_WRONG_DATA);
                }
                JCSystem.beginTransaction();
                Util.arrayCopyNonAtomic(bArray, bOffset, PIN, (short) 0, bLength);
                pin_size = bLength;
                JCSystem.commitTransaction();
        }

        private void verifyPin(byte[] bArray, short bOffset, byte bLength) {
                JCSystem.beginTransaction();
                if (bLength == pin_size
                        && Util.arrayCompare(PIN, (short) 0, bArray, bOffset,
                bLength) == 0) {
                        pin_tries = 0;
                        PIN_VERIFIED[0] = true;
                } else {
                        ++pin_tries;
                        PIN_VERIFIED[0] = false;
                        ISOException.throwIt(ISO7816.SW_WRONG_DATA);
                }
                JCSystem.commitTransaction();
        }

        private void checkPinStatus() {
                if (!PIN_VERIFIED[0]) {
                        ISOException.throwIt(
                                ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
                }
        }
}
```