```
// ATMInfra.java - ATM INFRASTRUCTURE EMULATOR
//
// MODULE INDEX
// NAME                          CONTENTS
// ATMInfra.genRandPix           Generate random pixels
// ATMInfra.changePix            Change a small percentage of the pixels in an
//                               array
// ATMInfra.emulateCardIn        Emulate insertion of a card into the reader
// ATMInfra.readScreenPixels     Read the current screen
// ATMInfra.readScreenDump       Read a saved screen dump
// ATMInfra.emulateTouch         Emulate touching the screen
// ATMInfra.getAmountPaid        Get the amount paid out by the bill dispenser
//
// MAINTENANCE HISTORY
// DATE         PROGRAMMER AND DETAILS
// 19-08-07     JS      Original
//
//-----------------------------------------------------------------------------

// IMPORTS

import java.util.*;

//-----------------------------------------------------------------------------

public class ATMInfra {

    //-------------------------------------------------------------------------

    // DEFINITIONS

    static private final int PIX_ARR_SIZE = 307200;
                                // Pixel array size
    static private final int MAX_CHG_PIX = 200;
                                // Maximum changed pixels
    static private final int ATM_PIN_SIZE = 6;
                                // PIN size
    static private final int MAX_DLR_LEN = 4;
                                // Maximum dollar value length
    static private final int CNT_LEN = 2;
                                // Length of the cents field
    static private final int DISP_UNIT = 1000;
                                // Dispensing unit (cents)
    static private final int MAX_AMT = 100000;
                                // Maximum amount to dispense (cents)
    static private final int MIN_AMT = 1000;
                                // Minimum amount to dispense (cents)

    //-------------------------------------------------------------------------

    // ATM STATE

    static private final int ATM_IDLE = 0;
                                // The ATM is between transactions
    static private final int ATM_RECV_PIN = 1;
                                // The ATM is receiving a PIN
    static private final int ATM_RECV_ACC = 2;
                                // The ATM is receiving an account selection
    static private final int ATM_RECV_DOLLARS = 3;
                                // The ATM is receiving the dollars component
                                // of the amount to be withdrawn
    static private final int ATM_RECV_CENTS = 4;
                                // The ATM is receiving the cents component
                                // of the amount to be withdrawn
```

```
static private final int ATM_DISPENSING = 5;
                            // The ATM is dispensing cash

//---------------------------------------------------------------------------

// STATIC VARIABLES

static private int  atmState = ATM_IDLE;    // ATM state
static private String atmCardNumber;        // Current card number
static private String atmPinVal;            // PIN value
static private String atmAccVal;            // Account code
static private int  atmAmtVal;              // Amount value
static private int  atmDlrLen;              // Length of dollars field
static private int  atmCntLen;              // Length of cents field
static private int  atmDlrRem = 0;          // Dollars remaining to be
                                            // dispensed
static private Random atmRandom = new Random(20360);
                                            // Random number generator
static private long[] atmAcpCrdPix = genRandPix();
                                            // Card accepted pixels
static private long[] atmAcpPinPix = genRandPix();
                                            // PIN accepted pixels
static private long[] atmAcpAccPix = genRandPix();
                                            // Account accepted pixels
static private long[] atmRejectPix = genRandPix();
                                            // Request rejected pixels

//---------------------------------------------------------------------------

// GENERATE RANDOM PIXELS

static private long[]
genRandPix ()
{
    long[]          pixArray;       // Pixel array
    int             i;              // General purpose index

    pixArray = new long [ PIX_ARR_SIZE ];
    for (i = 0; i < PIX_ARR_SIZE; i++)
        pixArray[i] = atmRandom.nextInt ();
    return pixArray;
}

//---------------------------------------------------------------------------

// CHANGE A SMALL PERCENTAGE OF THE PIXELS IN AN ARRAY

static private long[]
changePix (
    long[]          orgPix)         // Original pixels
{
    long[]          chgPix;         // Changed pixels
    int             i;              // General purpose index
    int             chgRem;         // Pixel changes remaining
    int             chgRand;        // Pixel change random number

    chgPix = new long [ PIX_ARR_SIZE ];
    chgRem = atmRandom.nextInt(MAX_CHG_PIX + 1);
    for (i = 0; i < PIX_ARR_SIZE; i++) {
            chgRand = atmRandom.nextInt(PIX_ARR_SIZE-i);
            if (chgRand < chgRem) {
                    chgPix[i] = atmRandom.nextLong();
                    chgRem -- ;
            } else {
```

```
                    chgPix[i] = orgPix[i];
                }
            }
        return chgPix;
    }

    //-------------------------------------------------------------------------

    // EMULATE INSERTION OF A CARD INTO THE READER

    static public void
    emulateCardIn (
        String          cardNumber)     // Card number
    {
        if (atmState != ATM_IDLE)
            throw new RuntimeException ("bad state " + atmState);
        atmCardNumber = cardNumber;
        System.out.println ("Card number " + atmCardNumber + " inserted");
        atmState = ATM_RECV_PIN;
        atmPinVal = "";
    }

    //-------------------------------------------------------------------------

    // READ THE CURRENT SCREEN

    static public long []
    readScreenPixels ()
    {
        long[]          pixArray;       // Pixel array

        switch (atmState) {
        case ATM_IDLE:
            pixArray = genRandPix();
            break;
        case ATM_RECV_PIN:
            pixArray = changePix(atmAcpCrdPix);
            break;
        case ATM_RECV_ACC:
            pixArray = changePix(atmAcpPinPix);
            break;
        case ATM_RECV_DOLLARS:
        case ATM_RECV_CENTS:
            pixArray = changePix(atmAcpAccPix);
            break;
        case ATM_DISPENSING:
            pixArray = genRandPix();
            break;
        default:
            throw new RuntimeException ("bad state " + atmState);
        }
        return pixArray;
    }

    //-------------------------------------------------------------------------

    // READ A SAVED SCREEN DUMP

    static public long []
    readScreenDump (
        String          fileName)       // File name
    {
        long[]          pixArray;       // Pixel array
```

```
    if (fileName.equals ("AcceptedCard.rgb"))
        pixArray = atmAcpCrdPix;
    else if (fileName.equals ("AcceptedPIN.rgb"))
        pixArray = atmAcpPinPix;
    else if (fileName.equals ("AcceptedAccount.rgb"))
        pixArray = atmAcpAccPix;
    else
        throw new RuntimeException (fileName + " does not exist");
    return pixArray;
}

//-----------------------------------------------------------------------

// EMULATE TOUCHING THE SCREEN

static public void
emulateTouch (
    int             x,              // Horizontal position
    int             y)              // Vertical position
{
    char            ch;             // Received character

    // Decode the character

    if (x == 246 && y == 406)
            ch = '0';
    else if (x == 394 && y == 406)
            ch = '.';
    else if (x == 246 && y == 332)
            ch = '1';
    else if (x == 320 && y == 332)
            ch = '2';
    else if (x == 394 && y == 332)
            ch = '3';
    else if (x == 246 && y == 258)
            ch = '4';
    else if (x == 320 && y == 258)
            ch = '5';
    else if (x == 394 && y == 258)
            ch = '6';
    else if (x == 246 && y == 184)
            ch = '7';
    else if (x == 320 && y == 184)
            ch = '8';
    else if (x == 394 && y == 184)
            ch = '9';
    else if (x == 560 && y == 184)
            ch = '\n';
    else if (x == 560 && y == 258)
            ch = '\b';
    else {
            throw new RuntimeException ("bad touch point x=" + x
                + " y=" + y);
    }

    // Process the character depending on the current state

    switch (atmState) {
    case ATM_RECV_PIN:
        if ( ! Character.isDigit (ch))
            throw new RuntimeException (
                "received non-digit while reading PIN");
        atmPinVal += ch;
        if (atmPinVal.length() >= ATM_PIN_SIZE) {
```

```
            System.out.println ("PIN " + atmPinVal + " entered");
            atmState = ATM_RECV_ACC;
        }
        break;
    case ATM_RECV_ACC:
        switch (ch) {
        case '8':
            atmAccVal = "Cheque";
            break;
        case '5':
            atmAccVal = "Savings";
            break;
        case '2':
            atmAccVal = "Credit";
            break;
        default:
            throw new RuntimeException ("invalid account selection");
        }
        System.out.println (atmAccVal + " selected");
        atmState = ATM_RECV_DOLLARS;
        atmAmtVal = 0;
        atmDlrLen = 0;
        atmCntLen = 0;
        break;
    case ATM_RECV_DOLLARS:
        if (Character.isDigit (ch)) {
            if (atmDlrLen >= MAX_DLR_LEN)
                throw new RuntimeException ("dollar value length too big");
            atmAmtVal = atmAmtVal * 10 + Character.digit (ch, 10);
            atmDlrLen ++ ;
        } else if (ch == '.') {
            atmState = ATM_RECV_CENTS;
        } else {
            throw new RuntimeException (
                "invalid character in dollars field");
        }
        break;
    case ATM_RECV_CENTS:
        if (Character.isDigit (ch)) {
            if (atmCntLen >= CNT_LEN)
                throw new RuntimeException ("cents length too big");
            atmAmtVal = atmAmtVal * 10 + Character.digit (ch, 10);
            atmCntLen ++ ;
        } else if (ch == '\n') {
            if (atmCntLen != CNT_LEN)
                throw new RuntimeException ("wrong number of cents digits");
            if (atmAmtVal % DISP_UNIT != 0)
                throw new RuntimeException (
                    "amount not multiple of dispensing unit");
            if (atmAmtVal > MAX_AMT)
                throw new RuntimeException ("too much");
            if (atmAmtVal < MIN_AMT)
                throw new RuntimeException ("too little");
            System.out.println ("Dispensing $" + atmAmtVal/100);
            atmState = ATM_DISPENSING;
            atmDlrRem += atmAmtVal;
        } else {
            throw new RuntimeException ("invalid character in cents field");
        }
        break;
    default:
        throw new RuntimeException ("bad state " + atmState);
    }
}
```

```
    //-------------------------------------------------------------------------

    // GET THE AMOUNT PAID OUT BY THE BILL DISPENSER

    static public int
    getAmountPaid ()
    {
        int             amtDisp;        // Amount dispensed

        if (atmDlrRem > 25000)
                amtDisp = 25000;
        else
                amtDisp = atmDlrRem;
        atmDlrRem -= amtDisp;
        if (amtDisp != 0 && atmDlrRem == 0) {
            System.out.println ("Dispensing finished");
            atmState = ATM_IDLE;
        }
        return amtDisp / 100;
    }
}
```