```
//# tail -n 9999 `find . -type f | grep '.java'` > combined.txt
==> ./src/bank/account/SavingAccount.java <==
package bank.account;
public class SavingAccount extends Account {
    private double interestRate = 0.1 / 100d;
    /** constructors **/
    public SavingAccount(String theAccountNumber, String thePIN,
            double theAvailableBalance, double theTotalBalance) {
        super(theAccountNumber, thePIN, theAvailableBalance, theTotalBalance);
    }
    public SavingAccount(String theAccountNumber, String thePIN,
            double theAvailableBalance, double theTotalBalance, double theInterest) {
        super(theAccountNumber, thePIN, theAvailableBalance, theTotalBalance);
        interestRate = theInterest;
    }
    /** getters **/
    public double getInterestRate() {
        return interestRate;
    }
    public String getInterestRateString() {
        return interestRate * 100 + "%";
    }
}
==> ./src/bank/account/CurrentAccount.java <==
package bank.account;
public class CurrentAccount extends Account {
    public CurrentAccount(String theAccountNumber, String thePIN, double
    theAvailableBalance,
            double theTotalBalance) {
        super(theAccountNumber, thePIN, theAvailableBalance, theTotalBalance);
        overdrawnLimit = 10000;
    }
    public CurrentAccount(String theAccountNumber, String thePIN, double
    theAvailableBalance,
            double theTotalBalance, double theOverdrawnLimit) {
        super(theAccountNumber, thePIN, theAvailableBalance, theTotalBalance);
        overdrawnLimit = theOverdrawnLimit;
    }
}
==> ./src/bank/account/Account.java <==
package bank.account;

import java.util.Vector;

import javax.security.auth.login.AccountNotFoundException;

import atm.exception.OverdrawnException;
import atm.utils.MyStaticStuff;

// Account.java
// Represents a bank account

public class Account {
    protected String accountNumber; // account number
    private String pin; // PIN for authentication
    protected double availableBalance; // funds available for withdrawal
    protected double totalBalance; // funds available + pending deposits
    protected double overdrawnLimit;

    /** Account constructor initializes attributes **/
```

```
59      public Account(String theAccountNumber, String thePIN,
60              double theAvailableBalance, double theTotalBalance) {
61          accountNumber = theAccountNumber;
62          pin = thePIN;
63          availableBalance = theAvailableBalance;
64          totalBalance = theTotalBalance;
65          overdrawnLimit = 0.0;
66      } // end Account constructor
67
68      /** Static methods **/
69      public static Account getAccount(Vector<Account> accounts,
70              String accountNumber) throws AccountNotFoundException {
71          for (Account account : accounts)
72              if (account.getAccountNumber() == accountNumber)
73                  return account;
74          throw new AccountNotFoundException();
75      }
76
77      public static boolean isMyBankAccount(String accountNumber) {
78          return accountNumber.charAt(0) == '1';
79      }
80
81      /** instance methods **/
82
83      // determines whether a user-specified PIN matches PIN in Account
84      public boolean validatePIN(String userPIN) {
85          System.out.println("right pin:" + pin);
86          System.out.println("trying pin:" + userPIN);
87          return pin.equals(userPIN);
88      } // end method validatePIN
89
90      public boolean isMyBankAccount() {
91          String accountStr = String.valueOf(accountNumber);
92          return accountStr.charAt(0) == '1';
93      }
94
95      public boolean isEnough(double amount) {
96          double requiredAmount = amount;
97          if (!isMyBankAccount())
98              requiredAmount += MyStaticStuff.EXTRA_CHARGE;
99          return ((availableBalance + overdrawnLimit) >= requiredAmount);
100     }
101
102     // credits an amount to the account
103     public void credit(double amount) {
104         availableBalance += amount; // add to available balance
105         totalBalance += amount; // add to total balance
106     } // end method credit
107
108     // debits an amount from the account
109     public void debit(double amount) throws OverdrawnException {
110         if ((availableBalance + overdrawnLimit) < amount)
111             throw new OverdrawnException();
112         availableBalance -= amount; // subtract from available balance
113         totalBalance -= amount; // subtract from total balance
114     } // end method debit
115
116     /** getters **/
117     // returns available balance
118     public double getAvailableBalance() {
```

```
        return availableBalance;
    } // end getAvailableBalance

    // returns the total balance
    public double getTotalBalance() {
        return totalBalance;
    } // end method getTotalBalance
        // returns account number

    public String getAccountNumber() {
        return accountNumber;
    } // end method getAccountNumber

    public double getOverdrawnLimit() {
        return overdrawnLimit;
    }

} // end class Account
==> ./src/bank/operation/Transaction.java <==
package bank.operation;
import javax.security.auth.login.AccountNotFoundException;
import bank.BankDatabase;
import atm.core.ATM;
import atm.core.Keypad;
import atm.core.Screen;
import atm.core.UI;
import atm.exception.CardOutException;
import atm.exception.CashNotesNotSupportedException;
import atm.exception.WrongInputException;
// Transaction.java
// Abstract superclass Transaction represents an ATM transaction
public abstract class Transaction {
    private String accountNumber; // indicates account involved
    private UI ui; // ATM's screen and keypad
    private BankDatabase bankDatabase; // account info database
    // Transaction constructor invoked by subclasses using super()
    public Transaction(ATM atm) {
        accountNumber = atm.getCurrentAccountNumber();
        ui = atm.getUI();
        bankDatabase = atm.getBankDatabase();
    } // end Transaction constructor
    // return account number
    public String getAccountNumber() {
        return accountNumber;
    } // end method getAccountNumber
    // return reference to screen
    public Screen getScreen() {
        return ui.screen;
    } // end method getScreen
    // return reference to keypad
    public Keypad getKeypad() {
        return ui.keypad;
    } // end method getKeypad
    // return reference to bank database
    public BankDatabase getBankDatabase() {
        return bankDatabase;
    } // end method getBankDatabase
    // perform the transaction (overridden by each subclass)
    public abstract void execute() throws WrongInputException,
            AccountNotFoundException, CardOutException,
```

```
            CashNotesNotSupportedException;
} // end class Transaction
==> ./src/bank/operation/BalanceInquiry.java <==
package bank.operation;

import javax.security.auth.login.AccountNotFoundException;

import bank.BankDatabase;
import atm.core.ATM;

// BalanceInquiry.java
// Represents a balance inquiry ATM transaction

public class BalanceInquiry extends Transaction {
    // BalanceInquiry constructor
    public BalanceInquiry(ATM atm) {
        super(atm);
    } // end BalanceInquiry constructor

    @Override
    // performs the transaction
    // get references to bank database and screen from parameters
    public void execute() throws AccountNotFoundException {
        getScreen().displayMessageLine("\nBalance Information:");

        getBankDatabase();
        // get & display the balance for the account on the screen
        double availableBalance = BankDatabase
                .getAvailableBalance(getAccountNumber());
        getBankDatabase();
        double totalBalance = BankDatabase.getTotalBalance(getAccountNumber());
        getScreen().displayMessage(" - Available balance: ");
        getScreen().displayDollarAmount(availableBalance);
        getScreen().displayMessageLine();
        getScreen().displayMessage(" - Total balance:     ");
        getScreen().displayDollarAmount(totalBalance);
        getScreen().displayMessageLine();

        getBankDatabase();
        // check if the account has interest rate
        if (BankDatabase.IsSavingAccount(getAccountNumber())) {
            getScreen().displayMessage(" - Interest rate:     ");
            getScreen().displayMessage(
                    BankDatabase.getInterestRateString(getAccountNumber()));
            getScreen().displayMessageLine();
        }

        getBankDatabase();
        // check if the account has overdraw limit
        if (BankDatabase.IsCurrentAccount(getAccountNumber())) {
            getScreen().displayMessage(" - Overdraw limit:     ");
            getScreen().displayMessage(
                    BankDatabase.getOverdrawLimit(getAccountNumber()));
            getScreen().displayMessageLine();
        }

        getScreen().displayMessageLine();
    } // end method execute
} // end class BalanceInquiry
==> ./src/bank/operation/Withdrawal.java <==
```

```java
package bank.operation;

import java.util.Vector;

import javax.security.auth.login.AccountNotFoundException;

import com.thoughtworks.xstream.InitializationException;

import bank.BankDatabase;
import bank.account.Account;
import atm.core.ATM;
import atm.core.CashDispenser;
import atm.core.Screen;
import atm.core.UI;
import atm.exception.CardOutException;
import atm.exception.CashNotEnoughException;
import atm.exception.CashNotesNotSupportedException;
import atm.exception.CashOutException;
import atm.exception.OverdrawnException;
import atm.exception.WrongInputException;
import atm.utils.CashCount;
import atm.utils.MyInputHandler;
import atm.utils.MyStaticStuff;
import atm.utils.MyStrings;

// Withdrawal.java
// Represents a withdrawal ATM transaction

public class Withdrawal extends Transaction {
    private int amount; // amount to withdraw
    private ATM atm;

    // constant corresponding to menu option to cancel
    private static int CANCELED;
    private boolean commandMode = true;

    // Withdrawal constructor
    // get references to keypad and cash dispenser from atm
    public Withdrawal(ATM atm) {
        // initialize superclass variables
        super(atm);
        this.atm = atm;
        CANCELED = MyStaticStuff.MenuCashValue.length + 2;
    } // end Withdrawal constructor

    public void setAmount(String amountStr) throws NumberFormatException {
        commandMode = false;
        this.amount = Integer.parseInt(amountStr);
    }

    @Override
    // perform transaction
    public void execute() throws WrongInputException, AccountNotFoundException,
            CardOutException, CashNotesNotSupportedException {
        boolean cashDispensed = false; // cash was not dispensed yet
        int tryCount = 0;
        // loop until cash is dispensed or the user cancels
        do {
            tryCount++;
            // obtain a chosen withdrawal amount from the user
```

```
299             amount = displayMenuOfAmounts(atm.getUI());
300
301             // check whether user chose a withdrawal amount or canceled
302             if (amount == CANCELED)
303                 return;
304
305             // auto check whether the user has enough money in the account
306             // check whether the cash dispenser has enough money
307             try {
308                 try {
309                     if (!Account.isMyBankAccount(getAccountNumber()))
310                         if (!BankDatabase.getAccount(getAccountNumber())
311                                 .isEnough(amount))
312                             throw new OverdrawnException();
313                     Vector<CashCount> cashPop = CashDispenser
314                             .dispenseCash(amount);
315                     if (!Account.isMyBankAccount(getAccountNumber()))
316                         BankDatabase.debit(getAccountNumber(),
317                                 MyStaticStuff.EXTRA_CHARGE);
318                     BankDatabase.debit(getAccountNumber(), amount);
319                     cashDispensed = true; // cash was dispensed
320                     atm.popCash(cashPop);
321                 } catch (OverdrawnException e) {
322                     getScreen().displayMessageLine(
323                             MyStrings.getOverDrawnMessage(BankDatabase
324                                     .getAccount(getAccountNumber())
325                                     .getOverdrawnLimit()));
326                     MyStaticStuff.sleep();
327                 }
328             } catch (CashNotEnoughException e) {
329                 // cash dispenser does not have enough cash
330                 getScreen().displayMessageLine(
331                         "\nInsufficient cash available in the ATM."
332                                 + "\n Avaliabe cash:"
333                                 + CashDispenser.getAmount()
334                                 + "\n\nPlease choose a smaller amount.");
335                 MyStaticStuff.sleep();
336             } // dispense cash
337         } while ((!cashDispensed)
338                 && (tryCount < MyInputHandler.MAX_WRONG_INPUT));
339     } // end method execute
340
341     public void executeGUI() throws AccountNotFoundException,
342             OverdrawnException, CashNotEnoughException, CashOutException,
            CashNotesNotSupportedException {
343         // throw overdrawexception, cashnotenoughexception
344         if (commandMode)
345             throw new InitializationException(
346                     "WithDrawal: amount has not be initialized");
347
348         int withExtraCharge = amount;
349         if (!Account.isMyBankAccount(getAccountNumber()))
350             withExtraCharge += MyStaticStuff.EXTRA_CHARGE;
351
352         if (!BankDatabase.getAccount(getAccountNumber()).isEnough(
353                 withExtraCharge))
354             throw new OverdrawnException();
355         Vector<CashCount> cashPop = CashDispenser.dispenseCash(amount);
356         if (!Account.isMyBankAccount(getAccountNumber()))
357             BankDatabase.debit(getAccountNumber(), MyStaticStuff.EXTRA_CHARGE);
```

```
358             BankDatabase.debit(getAccountNumber(), amount);
359
360             throw new CashOutException(cashPop);
361         }
362
363         // display a menu of withdrawal amounts and the option to cancel;
364         // return the chosen amount or 0 if the user chooses to cancel
365         private int displayMenuOfAmounts(UI ui) throws WrongInputException {
366             int userChoice = 0; // local variable to store return value
367
368             // loop while no valid choice has been made
369             while (userChoice == 0) {
370                 // display the menu
371                 String msg = "\nWithdrawal Menu:";
372                 int i = 0;
373                 for (Integer cashValue : MyStaticStuff.MenuCashValue)
374                     msg += "\n" + (++i) + " - " + Screen.getDollarAmount(cashValue);
375                 msg += "\n" + (++i) + " - Other";
376                 msg += "\n" + CANCELED + " - Cancel withdrawal";
377                 msg += "\n\nChoose a withdrawal amount: ";
378                 // get user input through keypad
379                 int input = ui.keypad.getInputInt(msg);
380
381                 // determine how to proceed based on the input value
382                 if (input == CANCELED)
383                     userChoice = CANCELED;
384                 else if (input == CANCELED - 1)
385                     userChoice = manualInputAmount(ui);
386                 else if ((input >= 1)
387                         && (input <= MyStaticStuff.MenuCashValue.length))
388                     userChoice = MyStaticStuff.MenuCashValue[input - 1];
389                 else
390                     ui.screen.displayMessageLine("\nIvalid selection. Try again.");
391             } // end while
392             return userChoice; // return withdrawal amount or CANCELED
393         } // end method displayMenuOfAmounts
394
395         private int manualInputAmount(UI ui) throws WrongInputException {
396             int amount = CANCELED;
397             int wrongInputCount = 0;
398             boolean ok;
399             do {
400                 ok = true;
401                 String msg = "We provide " + MyStaticStuff.getCashValuesStrings()
402                         + " note"
403                         + (MyStaticStuff.CashValues.length > 1 ? "s" : "")
404                         + " only";
405                 msg += "\nInput the amount to withdraw (input 0 to cancel): ";
406                 try {
407                     amount = ui.keypad.getInputInt(msg);
408                     if (amount == 0)
409                         return CANCELED;
410                     else if ((amount < 0) || ((amount % 100) != 0))
411                         throw new WrongInputException();
412                 } catch (WrongInputException e) {
413                     ok = false;
414                     ui.screen.displayMessageLine("Invalid input");
415                     MyStaticStuff.sleep();
416                 }
417             } while ((wrongInputCount <= MyInputHandler.MAX_WRONG_INPUT) && (!ok));
```

```
418             if (!ok)
419                 throw new WrongInputException();
420             return amount;
421         }
422     } // end class Withdrawal
423     ==> ./src/bank/operation/Transfer.java <==
424     package bank.operation;
425     import java.util.Vector;
426     import javax.security.auth.login.AccountNotFoundException;
427     import bank.BankDatabase;
428     import bank.account.Account;
429     import atm.core.ATM;
430     import atm.core.UI;
431     import atm.exception.OverdrawnException;
432     import atm.exception.TransferSameAccountException;
433     import atm.exception.WrongInputException;
434     import atm.utils.MyInputHandler;
435     import atm.utils.MyStaticStuff;
436     import atm.utils.MyStrings;
437     public class Transfer {
438         public static Vector<Transaction> transfer(ATM atm)
439                 throws WrongInputException, AccountNotFoundException {
440             Vector<Transaction> result = new Vector<Transaction>();
441             UI ui = atm.getUI();
442             String accountNumberTo;
443             Account accountFrom = BankDatabase.getAccount(atm
444                     .getCurrentAccountNumber());
445             Account accountTo = null;
446             double amount = 0;
447             boolean ok;
448             int wrongCount = 0;
449             // get target account to be transfered from user
450             do {
451                 ok = true;
452                 accountNumberTo = String
453                         .valueOf(ui.keypad
454                                 .getInputInt("\nPlease input the account number of the receiver: "));
455                 try {
456                     accountTo = BankDatabase.getAccount(accountNumberTo);
457                     if (accountFrom.getAccountNumber() == accountTo
458                             .getAccountNumber()) {
459                         wrongCount++;
460                         ok = false;
461                         ui.screen
462                                 .displayMessageLine(MyStrings.TRANSFER_SAME_ACCOUNT);
463                     }
464                 } catch (AccountNotFoundException e) {
465                     wrongCount++;
466                     ok = false;
467                     ui.screen.displayMessageLine(MyStrings.ACCOUNT_NOT_FOUND);
468                     MyStaticStuff.sleep();
469                 }
470             } while ((wrongCount <= MyInputHandler.MAX_WRONG_INPUT) && (!ok));
471             if (!ok)
472                 throw new WrongInputException();
473             // get amount to be transfered from user
474             // auto throw OverdrawnException if the accountFrom has not enough
475             // available balance
476             wrongCount = 0;
```

```
477             do {
478                 ok = true;
479                 try {
480                     amount = ui.keypad
481                             .getInputDouble("\nPlease the amount to transfer (input 0
                            to cancel): ");
482                     if (amount == 0)
483                         return null;
484                     else if (amount < 0) {
485                         wrongCount++;
486                         ok = false;
487                         ui.screen
488                                 .displayMessageLine("The amount should be positive.
                                Please try again.");
489                     } else if (accountFrom.isEnough(amount)) {
490                         accountFrom.debit(amount);
491                         accountTo.credit(amount);
492                     } else {
493                         throw new OverdrawnException();
494                     }
495                 } catch (OverdrawnException e) {
496                     wrongCount++;
497                     ok = false;
498                     ui.screen.displayMessageLine(MyStrings
499                             .getOverDrawnMessage(BankDatabase.getAccount(
500                                     atm.getCurrentAccountNumber())
501                                     .getOverdrawnLimit()));
502                     MyStaticStuff.sleep();
503                 }
504             } while ((wrongCount <= MyInputHandler.MAX_WRONG_INPUT) && (!ok));
505             if (!ok)
506                 throw new WrongInputException();
507             else {
508                 if (!accountFrom.isMyBankAccount())
509                     ui.screen.displayMessageLine(MyStaticStuff
510                             .getExtraChargeString());
511                 ui.screen.displayMessageLine(MyStrings.TRANSFER_SUCCEED);
512             }
513             return result;
514         }
515         public static void transferGUI(ATM atm, String accountNumberTo,
516                 double amount) throws AccountNotFoundException,
517                 TransferSameAccountException, OverdrawnException {
518             Account accountFrom = BankDatabase.getAccount(atm
519                     .getCurrentAccountNumber());
520             Account accountTo = BankDatabase.getAccount(accountNumberTo);
521             if (accountFrom.getAccountNumber() == accountTo.getAccountNumber())
522                 throw new TransferSameAccountException();
523             // auto throw OverdrawnException if the accountFrom has not enough
524             // available balance
525             double withExtraCharge = amount;
526             if (!accountFrom.isMyBankAccount())
527                 withExtraCharge += MyStaticStuff.EXTRA_CHARGE;
528             try {
529                 if (accountFrom.isEnough(withExtraCharge)) {
530                     accountFrom.debit(withExtraCharge);
531                     accountTo.credit(amount);
532                 } else {
533                     throw new OverdrawnException();
534                 }
```

```
535             } catch (OverdrawnException overdrawnException) {
536                 throw overdrawnException;
537             }
538         }
539     }
540     ==> ./src/bank/BankDatabase.java <==
541     package bank;
542     import java.util.Vector;
543     import javax.security.auth.login.AccountNotFoundException;
544     import atm.exception.OverdrawnException;
545     import atm.gui.virtualslots.cardslot.CardInsideJPanel;
546     import bank.account.Account;
547     import bank.account.CurrentAccount;
548     import bank.account.SavingAccount;
549     public class BankDatabase {
550         // Vector of Accounts
551         private static Vector<Account> accounts = new Vector<Account>();
552         // no-argument BankDatabase constructor initializes accounts
553         public static void init() {
554             accounts.add(new Account("12345", "02345", 5000.0, 5000.0));
555             accounts.add(new CurrentAccount("12356", "02356", 9000.0, 10000.0));
556             accounts.add(new SavingAccount("12369", "02369", 23000.0, 23000.0));
557             accounts.add(new Account("45678", "05678", 2000.0, 2000.0));
558         } // end no-argument BankDatabase constructor
559         /** getters **/
560         // retrieve Account object containing specified account number
561         public static Account getAccount(String accountNumber) throws ↵
            AccountNotFoundException {
562             // loop through accounts searching for matching account number
563             for (Account account : accounts) {
564                 // return current account if match found
565                 if (account.getAccountNumber().equals(accountNumber))
566                     return account;
567             } // end for
568             throw new AccountNotFoundException(); // if no matching account was
569                                                   // found, throw exception
570         } // end method getAccount
571         public static Vector<Account> getAccounts() {
572             return accounts;
573         }
574         // return available balance of Account with specified account number
575         public static double getAvailableBalance(String userAccountNumber) throws ↵
            AccountNotFoundException {
576             return getAccount(userAccountNumber).getAvailableBalance();
577         } // end method getAvailableBalance
578         // return total balance of Account with specified account number
579         public static double getTotalBalance(String userAccountNumber) throws ↵
            AccountNotFoundException {
580             return getAccount(userAccountNumber).getAvailableBalance();
581         } // end method getTotalBalance
582         // return interest rate of Account with specified account number
583         public static double getInterestRate(String userAccountNumber) throws ↵
            AccountNotFoundException {
584             return ((SavingAccount) getAccount(userAccountNumber)).getInterestRate();
585         } // end method getInterestRate
586         // return interest rate of Account with specified account number
587         // in unit of %
588         public static String getInterestRateString(String userAccountNumber) throws ↵
            AccountNotFoundException {
589             return ((SavingAccount) getAccount(userAccountNumber)). ↵
```

```
        getInterestRateString();
590     } // end method getInterestRateString
591     // return overdraw limit of Account with specified account number
592     public static double getOverdrawLimit(String userAccountNumber) throws
        AccountNotFoundException {
593         return getAccount(userAccountNumber).getOverdrawnLimit();
594     } // end method getOverdrawLimit
595     /** instance methods **/
596     // determine whether the account is saving account
597     public static boolean IsSavingAccount(String userAccountNumber) throws
        AccountNotFoundException {
598         Account account = getAccount(userAccountNumber);
599         return account instanceof SavingAccount;
600     }
601     // determine whether the account is current account
602     public static boolean IsCurrentAccount(String userAccountNumber) throws
        AccountNotFoundException {
603         Account account = getAccount(userAccountNumber);
604         return account instanceof CurrentAccount;
605     }
606     // determine whether user-specified account number and PIN match
607     // those of an account in the database
608     @Deprecated
609     public static boolean authenticateUser_old(String userAccountNumber, String
        userPIN)
610             throws AccountNotFoundException {
611         System.out.println("authenticateUser_old");
612         // attempt to retrieve the account with the account number
613         Account userAccount;
614         userAccount = getAccount(userAccountNumber);
615         // if account exists, return result of Account method validateIN
616         return userAccount.validatePIN(userPIN);
617     } // end method authenticateUser
618     public static boolean authenticateUser(String userPIN) throws
        AccountNotFoundException {
619         System.out.println("authenticating User: " + CardInsideJPanel.getCard().
            accountNumber);
620         // attempt to retrieve the account with the account number
621         Account userAccount;
622         userAccount = getAccount(CardInsideJPanel.getCard().accountNumber);
623         // if account exists, return result of Account method validateIN
624         return userAccount.validatePIN(userPIN);
625     } // end method authenticateUser
626     // credit an amount to Account with specified account number
627     public static void credit(String userAccountNumber, double amount) throws
        AccountNotFoundException {
628         getAccount(userAccountNumber).credit(amount);
629     } // end method credit
630     // debit an amount from of Account with specified account number
631     public static void debit(String userAccountNumber, double amount) throws
        OverdrawnException,
632             AccountNotFoundException {
633         getAccount(userAccountNumber).debit(amount);
634     } // end method debit
635 } // end class BankDatabase
636 ==> ./src/webs/layout/CenterLayout.java <==
637 package webs.layout;
638 /*
639  * JCommon : a free general purpose class library for the Java(tm) platform
640  *
```

```
 *
 * (C) Copyright 2000-2005, by Object Refinery Limited and Contributors.
 *
 * Project Info:  http://www.jfree.org/jcommon/index.html
 *
 * This library is free software; you can redistribute it and/or modify it
 * under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation; either version 2.1 of the License, or
 * (at your option) any later version.
 *
 */
import java.awt.Component;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.Insets;
import java.awt.LayoutManager;
import java.io.Serializable;
/**
 * A layout manager that displays a single component in the center of its
 * container.
 *
 * @author David Gilbert
 */
public class CenterLayout implements LayoutManager, Serializable {
    /** For serialization. */
    private static final long serialVersionUID = 4693195323330015042L;
    /** Creates a new layout manager.*/
    public CenterLayout() {
    }
    /** Returns the preferred size.
     * @param parent
     *            the parent.
     * @return the preferred size.*/
    public Dimension preferredLayoutSize(final Container parent) {
        synchronized (parent.getTreeLock()) {
            final Insets insets = parent.getInsets();
            if (parent.getComponentCount() > 0) {
                final Component component = parent.getComponent(0);
                final Dimension d = component.getPreferredSize();
                return new Dimension((int) d.getWidth() + insets.left
                        + insets.right, (int) d.getHeight() + insets.top
                        + insets.bottom);
            } else {
                return new Dimension(insets.left + insets.right, insets.top
                        + insets.bottom);
            }
        }
    }
    /** Returns the minimum size.
     * @param parent
     *            the parent.
     * @return the minimum size.*/
    public Dimension minimumLayoutSize(final Container parent) {
        synchronized (parent.getTreeLock()) {
            final Insets insets = parent.getInsets();
            if (parent.getComponentCount() > 0) {
                final Component component = parent.getComponent(0);
                final Dimension d = component.getMinimumSize();
                return new Dimension(d.width + insets.left + insets.right,
                        d.height + insets.top + insets.bottom);
```

```
            } else {
                return new Dimension(insets.left + insets.right, insets.top
                        + insets.bottom);
            }
        }
    }
    /** Lays out the components.
     * @param parent
     *            the parent.*/
    public void layoutContainer(final Container parent) {
        synchronized (parent.getTreeLock()) {
            if (parent.getComponentCount() > 0) {
                final Insets insets = parent.getInsets();
                final Dimension parentSize = parent.getSize();
                final Component component = parent.getComponent(0);
                final Dimension componentSize = component.getPreferredSize();
                final int xx = insets.left
                        + (Math.max((parentSize.width - insets.left
                                - insets.right - componentSize.width) / 2, 0));
                final int yy = insets.top
                        + (Math.max((parentSize.height - insets.top
                                - insets.bottom - componentSize.height) / 2, 0));
                component.setBounds(xx, yy, componentSize.width,
                        componentSize.height);
            }
        }
    }
    /** Not used.
     * @param comp
     *            the component.*/
    public void addLayoutComponent(final Component comp) {
        // not used.
    }
    /** Not used.
     * @param comp
     *            the component.*/
    public void removeLayoutComponent(final Component comp) {
        // not used
    }
    /** Not used.
     * @param name
     *            the component name.
     * @param comp
     *            the component.*/
    public void addLayoutComponent(final String name, final Component comp) {
        // not used
    }
    /** Not used.
     * @param name
     *            the component name.
     * @param comp
     *            the component.*/
    public void removeLayoutComponent(final String name, final Component comp) {
        // not used
    }
}
==> ./src/webs/layout/WrapLayout.java <==
package webs.layout;

import java.awt.*;
```

```
761
762 import javax.swing.JScrollPane;
763 import javax.swing.SwingUtilities;
764
765 /**
766  *  FlowLayout subclass that fully supports wrapping of components.
767  */
768 @SuppressWarnings("serial")
769 public class WrapLayout extends FlowLayout
770 {
771     @SuppressWarnings("unused")
772     private Dimension preferredLayoutSize;
773
774     /**
775     * Constructs a new <code>WrapLayout</code> with a left
776     * alignment and a default 5-unit horizontal and vertical gap.
777     */
778     public WrapLayout()
779     {
780         super();
781     }
782
783     /**
784     * Constructs a new <code>FlowLayout</code> with the specified
785     * alignment and a default 5-unit horizontal and vertical gap.
786     * The value of the alignment argument must be one of
787     * <code>WrapLayout</code>, <code>WrapLayout</code>,
788     * or <code>WrapLayout</code>.
789     * @param align the alignment value
790     */
791     public WrapLayout(int align)
792     {
793         super(align);
794     }
795
796     /**
797     * Creates a new flow layout manager with the indicated alignment
798     * and the indicated horizontal and vertical gaps.
799     * <p>
800     * The value of the alignment argument must be one of
801     * <code>WrapLayout</code>, <code>WrapLayout</code>,
802     * or <code>WrapLayout</code>.
803     * @param align the alignment value
804     * @param hgap the horizontal gap between components
805     * @param vgap the vertical gap between components
806     */
807     public WrapLayout(int align, int hgap, int vgap)
808     {
809         super(align, hgap, vgap);
810     }
811
812     /**
813     * Returns the preferred dimensions for this layout given the
814     * <i>visible</i> components in the specified target container.
815     * @param target the component which needs to be laid out
816     * @return the preferred dimensions to lay out the
817     * subcomponents of the specified container
818     */
819     @Override
820     public Dimension preferredLayoutSize(Container target)
```

```
821         {
822             return layoutSize(target, true);
823         }
824 
825         /**
826         * Returns the minimum dimensions needed to layout the <i>visible</i>
827         * components contained in the specified target container.
828         * @param target the component which needs to be laid out
829         * @return the minimum dimensions to lay out the
830         * subcomponents of the specified container
831         */
832         @Override
833         public Dimension minimumLayoutSize(Container target)
834         {
835             Dimension minimum = layoutSize(target, false);
836             minimum.width -= (getHgap() + 1);
837             return minimum;
838         }
839 
840         /**
841         * Returns the minimum or preferred dimension needed to layout the target
842         * container.
843         *
844         * @param target target to get layout size for
845         * @param preferred should preferred size be calculated
846         * @return the dimension to layout the target container
847         */
848         private Dimension layoutSize(Container target, boolean preferred)
849         {
850         synchronized (target.getTreeLock())
851         {
852             //  Each row must fit with the width allocated to the containter.
853             //  When the container width = 0, the preferred width of the container
854             //  has not yet been calculated so lets ask for the maximum.
855 
856             int targetWidth = target.getSize().width;
857 
858             if (targetWidth == 0)
859                 targetWidth = Integer.MAX_VALUE;
860 
861             int hgap = getHgap();
862             int vgap = getVgap();
863             Insets insets = target.getInsets();
864             int horizontalInsetsAndGap = insets.left + insets.right + (hgap * 2);
865             int maxWidth = targetWidth - horizontalInsetsAndGap;
866 
867             //  Fit components into the allowed width
868 
869             Dimension dim = new Dimension(0, 0);
870             int rowWidth = 0;
871             int rowHeight = 0;
872 
873             int nmembers = target.getComponentCount();
874 
875             for (int i = 0; i < nmembers; i++)
876             {
877                 Component m = target.getComponent(i);
878 
879                 if (m.isVisible())
880                 {
```

```
881                 Dimension d = preferred ? m.getPreferredSize() : m.getMinimumSize();
882
883                 //  Can't add the component to current row. Start a new row.
884
885                 if (rowWidth + d.width > maxWidth)
886                 {
887                     addRow(dim, rowWidth, rowHeight);
888                     rowWidth = 0;
889                     rowHeight = 0;
890                 }
891
892                 //  Add a horizontal gap for all components after the first
893
894                 if (rowWidth != 0)
895                 {
896                     rowWidth += hgap;
897                 }
898
899                 rowWidth += d.width;
900                 rowHeight = Math.max(rowHeight, d.height);
901             }
902         }
903
904         addRow(dim, rowWidth, rowHeight);
905
906         dim.width += horizontalInsetsAndGap;
907         dim.height += insets.top + insets.bottom + vgap * 2;
908
909         //  When using a scroll pane or the DecoratedLookAndFeel we need to
910         //  make sure the preferred size is less than the size of the
911         //  target containter so shrinking the container size works
912         //  correctly. Removing the horizontal gap is an easy way to do this.
913
914         Container scrollPane = SwingUtilities.getAncestorOfClass(JScrollPane.class, ↵
             target);
915
916         if (scrollPane != null && target.isValid())
917         {
918             dim.width -= (hgap + 1);
919         }
920
921         return dim;
922     }
923     }
924
925     /*
926      *  A new row has been completed. Use the dimensions of this row
927      *  to update the preferred size for the container.
928      *
929      *  @param dim update the width and height when appropriate
930      *  @param rowWidth the width of the row to add
931      *  @param rowHeight the height of the row to add
932  */
933     private void addRow(Dimension dim, int rowWidth, int rowHeight)
934     {
935         dim.width = Math.max(dim.width, rowWidth);
936
937         if (dim.height > 0)
938         {
939             dim.height += getVgap();
```

```
        }

        dim.height += rowHeight;
    }
}
==> ./src/webs/layout/CircleLayout.java <==
package webs.layout;
/*
 This program is a part of the companion code for Core Java 8th ed.
 (http://horstmann.com/corejava)
 This program is free software: you can redistribute it and/or modify
 it under the terms of the GNU General Public License as published by
 the Free Software Foundation, either version 3 of the License, or
 (at your option) any later version.
 This program is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 GNU General Public License for more details.
 You should have received a copy of the GNU General Public License
 along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */
import java.awt.Component;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.Insets;
import java.awt.LayoutManager;
/**
 * @version 1.32 2007-06-12
 * @author Cay Horstmann
 */
/**
 * A frame that shows buttons arranged along a circle.
 */
/**
 * A layout manager that lays out components along a circle.
 */
public class CircleLayout implements LayoutManager {
    public void addLayoutComponent(String name, Component comp) {
    }
    public void removeLayoutComponent(Component comp) {
    }
    public void setSizes(Container parent) {
        if (sizesSet)
            return;
        int n = parent.getComponentCount();
        preferredWidth = 0;
        preferredHeight = 0;
        minWidth = 0;
        minHeight = 0;
        maxComponentWidth = 0;
        maxComponentHeight = 0;
        // compute the maximum component widths and heights
        // and set the preferred size to the sum of the component sizes.
        for (int i = 0; i < n; i++) {
            Component c = parent.getComponent(i);
            if (c.isVisible()) {
                Dimension d = c.getPreferredSize();
                maxComponentWidth = Math.max(maxComponentWidth, d.width);
                maxComponentHeight = Math.max(maxComponentHeight, d.height);
                preferredWidth += d.width;
```

```
                preferredHeight += d.height;
            }
        }
        minWidth = preferredWidth / 2;
        minHeight = preferredHeight / 2;
        sizesSet = true;
    }
    public Dimension preferredLayoutSize(Container parent) {
        setSizes(parent);
        Insets insets = parent.getInsets();
        int width = preferredWidth + insets.left + insets.right;
        int height = preferredHeight + insets.top + insets.bottom;
        return new Dimension(width, height);
    }
    public Dimension minimumLayoutSize(Container parent) {
        setSizes(parent);
        Insets insets = parent.getInsets();
        int width = minWidth + insets.left + insets.right;
        int height = minHeight + insets.top + insets.bottom;
        return new Dimension(width, height);
    }
    public void layoutContainer(Container parent) {
        setSizes(parent);
        // compute center of the circle
        Insets insets = parent.getInsets();
        int containerWidth = parent.getSize().width - insets.left - insets.right;
        int containerHeight = parent.getSize().height - insets.top - insets.bottom;
        int xcenter = insets.left + containerWidth / 2;
        int ycenter = insets.top + containerHeight / 2;
        // compute radius of the circle
        int xradius = (containerWidth - maxComponentWidth) / 2;
        int yradius = (containerHeight - maxComponentHeight) / 2;
        int radius = Math.min(xradius, yradius);
        // lay out components along the circle
        int n = parent.getComponentCount();
        for (int i = 0; i < n; i++) {
            Component c = parent.getComponent(i);
            if (c.isVisible()) {
                double angle = 2 * Math.PI * i / n;
                // center point of component
                int x = xcenter + (int) (Math.cos(angle) * radius);
                int y = ycenter + (int) (Math.sin(angle) * radius);
                // move component so that its center is (x, y)
                // and its size is its preferred size
                Dimension d = c.getPreferredSize();
                c.setBounds(x - d.width / 2, y - d.height / 2, d.width, d.height);
            }
        }
    }
    private int minWidth = 0;
    private int minHeight = 0;
    private int preferredWidth = 0;
    private int preferredHeight = 0;
    private boolean sizesSet = false;
    private int maxComponentWidth = 0;
    private int maxComponentHeight = 0;
}
==> ./src/atm/exception/CashOutException.java <==
package atm.exception;
import java.util.Vector;
```

```
import atm.utils.CashCount;
public class CashOutException extends Exception {
    /***/
    private static final long serialVersionUID = 1L;
    private Vector<CashCount> popCashCounts;
    public CashOutException(Vector<CashCount> popCashCounts) {
        this.popCashCounts = popCashCounts;
        System.out.println(toString() + "==>" + popCashCounts.toString());
    }
    @Override
    public String toString() {
        return "Cash Out Exception";
    }
    public Vector<CashCount> getCashCounts() {
        return popCashCounts;
    }
}
==> ./src/atm/exception/TransferSameAccountException.java <==
package atm.exception;
public class TransferSameAccountException extends Exception {
    /***/
    private static final long serialVersionUID = 1L;
    public TransferSameAccountException() {
        System.out.println(toString());
    }
    @Override
    public String toString() {
        return "Transfer Same Account Exception";
    }
}
==> ./src/atm/exception/OverdrawnException.java <==
package atm.exception;
public class OverdrawnException extends Exception {
    /***/
    private static final long serialVersionUID = 1L;
    public OverdrawnException() {
        System.out.println(toString());
    }
    @Override
    public String toString() {
        return "Overdrawn Exception";
    }
}
==> ./src/atm/exception/WrongInputException.java <==
package atm.exception;
public class WrongInputException extends Exception {
    /***/
    private static final long serialVersionUID = 1L;
    public WrongInputException() {
        System.out.println(toString());
    }
    @Override
    public String toString() {
        return "WrongInput Exception";
    }
}
==> ./src/atm/exception/CashNotesNotSupportedException.java <==
package atm.exception;
import java.util.Vector;
import atm.utils.CashCount;
```

```
1120  public class CashNotesNotSupportedException extends Exception {
1121      /***/
1122      private static final long serialVersionUID = 1L;
1123      public CashNotesNotSupportedException() {
1124          System.out.println(toString());
1125      }
1126      public CashNotesNotSupportedException(int amountRequired,
1127              Vector<CashCount> result) {
1128          System.out.println("CashNotesNotSupportedException!");
1129          System.out.println("amountRequired: " + amountRequired);
1130          System.out.println("result set: " + result.toString());
1131      }
1132      @Override
1133      public String toString() {
1134          return "CashNotEnough Exception";
1135      }
1136  }
1137  ==> ./src/atm/exception/CashNotEnoughException.java <==
1138  package atm.exception;
1139  public class CashNotEnoughException extends Exception {
1140      /***/
1141      private static final long serialVersionUID = 1L;
1142      public CashNotEnoughException() {
1143          System.out.println(toString());
1144      }
1145      @Override
1146      public String toString() {
1147          return "CashNotEnoughException Exception";
1148      }
1149  }
1150  ==> ./src/atm/exception/CardOutException.java <==
1151  package atm.exception;
1152  public class CardOutException extends Exception {
1153      /***/
1154      private static final long serialVersionUID = 1L;
1155      public CardOutException() {
1156          System.out.println(toString());
1157      }
1158      @Override
1159      public String toString() {
1160          return "Card Out Exception";
1161      }
1162  }
1163  ==> ./src/atm/utils/MyInputHandler.java <==
1164  package atm.utils;
1165  public class MyInputHandler {
1166      public static int MAX_WRONG_INPUT = 3;
1167      public static String CARDOUT = "CARDOUT";
1168  }
1169  ==> ./src/atm/utils/MyImages.java <==
1170  package atm.utils;
1171  import java.io.IOException;
1172  import java.net.URL;
1173  import javax.swing.ImageIcon;
1174  import myutils.gui.MyImageUtils;
1175  import atm.gui.MyGUISettings;
1176  public class MyImages implements FetchImageNeeder {
1177      public static ImageIcon banner;
1178      public static ImageIcon viewBalance;
1179      public static ImageIcon transfer;
```

```
1180        public static ImageIcon extraCharge;
1181        public static ImageIcon cashNotEnough;
1182        private static boolean inited = false;
1183        public static void init() throws IOException {
1184            if (inited)
1185                return;
1186            banner = MyImageUtils.scaleImageIconByHeight(new ImageIcon(new URL(
1187                    MyURLs.IMAGE_BANNER)), MyGUISettings.MONITOR_TOP_MARGIN);
1188            viewBalance = MyImageUtils.scaleImageIconByHeight(new ImageIcon(
1189                    new URL(MyURLs.IMAGE_VIEW_BALANCE)),
1190                    MyGUISettings.MONITOR_TOP_MARGIN);
1191            transfer = MyImageUtils.scaleImageIconByHeight(new ImageIcon(new URL(
1192                    MyURLs.IMAGE_TRANSFER)), MyGUISettings.MONITOR_TOP_MARGIN);
1193            extraCharge = MyImageUtils.scaleImageIconByHeight(new ImageIcon(
1194                    new URL(MyURLs.IMAGE_EXTRA_CHARGE)),
1195                    MyGUISettings.MONITOR_TOP_MARGIN);
1196            cashNotEnough = MyImageUtils.scaleImageIconByHeight(new ImageIcon(
1197                    new URL(MyURLs.IMAGE_CASH_NOT_ENOUGH)),
1198                    MyGUISettings.MONITOR_TOP_MARGIN);
1199            inited = true;
1200        }
1201        @Override
1202        public void fetchImage() throws IOException {
1203            init();
1204        }
1205    }
1206    ==> ./src/atm/utils/MyURLs.java <==
1207    package atm.utils;
1208    public class MyURLs {
1209        public static final String IMAGE_TRIANGLE_POINT_LEFT =
            "http://xtupload.com/image.php?id=1AEE_54755B4E";
1210        public static final String IMAGE_TRIANGLE_POINT_RIGHT =
            "http://xtupload.com/image.php?id=ED71_54755B4E";
1211        public static final String IMAGE_BANNER =
            "http://xtupload.com/image.php?id=6E8A_5475599F";
1212        public static final String IMAGE_CARD1 =
            "http://xtupload.com/image.php?id=C4C5_5475599F&gif";
1213        public static final String IMAGE_CARD2 =
            "http://xtupload.com/image.php?id=E041_54755AFF&gif";
1214        public static final String IMAGE_CARD3 =
            "http://xtupload.com/image.php?id=5E5E_54755AFF&gif";
1215        public static final String IMAGE_CARD4 =
            "http://xtupload.com/image.php?id=899A_54755B4E&gif";
1216        public static final String IMAGE_NOTE100 =
            "http://xtupload.com/image.php?id=F1F4_54755BB3&gif";
1217        public static final String IMAGE_NOTE500 =
            "http://xtupload.com/image.php?id=1499_54755BB3&gif";
1218        public static final String IMAGE_NOTE1000 =
            "http://xtupload.com/image.php?id=B5A2_54755BB3&gif";
1219        public static final String IMAGE_COMFIRM_TRANSFER =
            "http://xtupload.com/image.php?id=8A92_5475599F&gif";
1220        public static final String IMAGE_CANCEL =
            "http://xtupload.com/image.php?id=C7B4_5475599F&gif";
1221        public static final String IMAGE_CLEAR =
            "http://xtupload.com/image.php?id=75FE_54755B4E&gif";
1222        public static final String IMAGE_ENTER =
            "http://xtupload.com/image.php?id=A967_54755B4E&gif";
1223        public static final String IMAGE_CARD1_DARK =
            "http://xtupload.com/image.php?id=AD5C_5475599F&gif";
1224        public static final String IMAGE_CARD2_DARK =
```

```
        "http://xtupload.com/image.php?id=EF6B_54755AFF&gif";
1225    public static final String IMAGE_CARD3_DARK =
        "http://xtupload.com/image.php?id=5389_54755AFF&gif";
1226    public static final String IMAGE_CARD4_DARK =
        "http://xtupload.com/share.php?id=281B_54755AFF";
1227    public static final String IMAGE_MENU =
        "http://xtupload.com/image.php?id=D2B1_54755BB3&gif";
1228    public static final String IMAGE_WITHDRAW_MENU =
        "http://xtupload.com/image.php?id=93F0_54755BB3&gif";
1229    @Deprecated
1230    public static final String IMAGE_PIX =
        "http://xtupload.com/image.php?id=A5EC_54755C00";
1231    public static final String IMAGE_VIEW_BALANCE =
        "http://xtupload.com/image.php?id=7510_5476AAEB";
1232    public static final String IMAGE_TRANSFER =
        "http://xtupload.com/image.php?id=F85F_547B1FA3";
1233    public static final String IMAGE_EXTRA_CHARGE_THREE_Line =
        "http://xtupload.com/image.php?id=19BB_547B2F97";
1234    public static final String IMAGE_EXTRA_CHARGE =
        "http://xtupload.com/image.php?id=E99F_547B3824";
1235    public static final String IMAGE_CASH_NOT_ENOUGH =
        "http://xtupload.com/image.php?id=DCDC_547B3D18";
1236 }
1237 class MyURLs_OLD {
1238    public static final String IMAGE_TRIANGLE_POINT_LEFT =
        "http://aabbcc1241.freeiz.com/HKCC/oop/publish/images/triangle_point_left.png";
1239    public static final String IMAGE_TRIANGLE_POINT_RIGHT =
        "http://aabbcc1241.freeiz.com/HKCC/oop/publish/images/triangle_point_right.png";
1240    public static final String IMAGE_BANNER =
        "http://aabbcc1241.freeiz.com/HKCC/oop/publish/images/banner.png";
1241    public static final String IMAGE_CARD1 =
        "http://aabbcc1241.freeiz.com/HKCC/oop/publish/images/CARD1.gif";
1242    public static final String IMAGE_CARD1_DARK =
        "http://xtupload.com/image.php?id=5390_54752750&gif";
1243    public static final String IMAGE_CARD2 =
        "http://aabbcc1241.freeiz.com/HKCC/oop/publish/images/CARD2.gif";
1244    public static final String IMAGE_CARD2_DARK =
        "http://xtupload.com/image.php?id=BDCC_54752750&gif";
1245    public static final String IMAGE_CARD3 =
        "http://aabbcc1241.freeiz.com/HKCC/oop/publish/images/CARD3.gif";
1246    public static final String IMAGE_CARD3_DARK =
        "http://xtupload.com/image.php?id=2DB9_547525A6&gif";
1247    public static final String IMAGE_CARD4 =
        "http://aabbcc1241.freeiz.com/HKCC/oop/publish/images/CARD4.gif";
1248    public static final String IMAGE_CARD4_DARK =
        "http://xtupload.com/image.php?id=5A81_547525A6&gif";
1249    public static final String IMAGE_NOTE100 =
        "http://aabbcc1241.freeiz.com/HKCC/oop/publish/images/dollar100.gif";
1250    public static final String IMAGE_NOTE500 =
        "http://aabbcc1241.freeiz.com/HKCC/oop/publish/images/dollar500.gif";
1251    public static final String IMAGE_NOTE1000 =
        "http://aabbcc1241.freeiz.com/HKCC/oop/publish/images/dollar1000.gif";
1252 }
1253 ==> ./src/atm/utils/FetchImageRunnable.java <==
1254 package atm.utils;
1255 import java.io.IOException;
1256 public class FetchImageRunnable implements Runnable {
1257    private FetchImageNeeder needer;
1258    private boolean finished = false;
1259    public FetchImageRunnable(FetchImageNeeder needer) {
```

```
        this.needer = needer;
    }
    @Override
    public void run() {
        try {
            needer.fetchImage();
        } catch (IOException e) {
            System.out.println("Failed to get to internet");
            e.printStackTrace();
        }
        finished = true;
    }
    public boolean isFinished() {
        return finished;
    }
}
==> ./src/atm/utils/FetchImageNeeder.java <==
package atm.utils;
import java.io.IOException;
public interface FetchImageNeeder {
    void fetchImage() throws IOException;
}
==> ./src/atm/utils/CashCount.java <==
package atm.utils;
import atm.exception.CashNotEnoughException;
public class CashCount {
    private final int value;
    private int count;
    public CashCount(int value, int count) {
        this.value = value;
        this.count = count;
    }
    public int getValue() {
        return value;
    }
    public int getCount() {
        return count;
    }
    public void remove(int number) throws CashNotEnoughException {
        if (number > count)
            throw new CashNotEnoughException();
        count -= number;
    }
    public void add(int number) {
        count += number;
    }
    @Override
    public String toString() {
        return MyStrings.DOLLAR_SIGN + " " + value + " x" + count;
    }
}
==> ./src/atm/utils/MyStaticStuff.java <==
package atm.utils;
import java.util.Vector;
import atm.core.Screen;
public class MyStaticStuff {
    public static int[] CashValues = { 100, 500, 1000 };
    public static final int[] MenuCashValue = { 200, 400, 800, 1000 };
    public static double EXTRA_CHARGE = 20;
    public static void sleep() {
```

```
1320            MyStaticStuff.sleep(1000);
1321        }
1322        public static void sleep(long millis) {
1323            try {
1324                Thread.sleep(millis);
1325            } catch (InterruptedException e) {
1326            }
1327        }
1328        public static String getCashValuesStrings() {
1329            Vector<String> cashValuesStrings = new Vector<String>();
1330            for (int i = 0; i < CashValues.length; i++)
1331                cashValuesStrings.add(Screen.getDollarAmount(CashValues[i]));
1332            return cashValuesStrings.toString();
1333        }
1334        public static String getExtraChargeString() {
1335            return "Extra charge: " + Screen.getDollarAmount(EXTRA_CHARGE)
1336                    + " will be charged for successful transaction";
1337        }
1338        public static String getCashValuesStrings(Vector<CashCount> cashCounts) {
1339            Vector<String> cashValuesStrings = new Vector<String>();
1340            for (CashCount cashCount : cashCounts)
1341                if (cashCount.getCount() > 0)
1342                    cashValuesStrings.add(Screen.getDollarAmount(cashCount.getValue())
                    + "x"
1343                            + cashCount.getCount());
1344            return cashValuesStrings.toString();
1345        }
1346    }
1347    ==> ./src/atm/utils/MyStrings.java <==
1348    package atm.utils;
1349    import java.util.Vector;
1350    import javax.swing.JLabel;
1351    import atm.core.Screen;
1352    public class MyStrings {
1353        /** Business part **/
1354        public static final String DOLLAR_SIGN = "HKD $";
1355        public static final String TAKE_CARD = "Please take your card";
1356        public static final String TAKE_CASH = "Please take your cash";
1357        public static final String TAKE_RECEIPT = "Please take your receipt";
1358        public static final String CONTACTS_US = "Please contact CC Bank (9876-5432)";
1359        public static final String WRONG_INPUT = "Invalid inputs";
1360        public static final String BYE1 = "Thank you for using our service.";
1361        public static final String BYE2 = "Have a good day!";
1362        @Deprecated
1363        public static final String BYE = BYE1 + " " + BYE2;
1364        public static final String ACCOUNT_NOT_FOUND = "The account is not found";
1365        public static final String TRANSFER_SAME_ACCOUNT = "Please do not transfer to
        the same account";
1366        public static final String TRANSFER_SUCCEED = "The transfer has been done";
1367        public static final String CARD_NOT_VALID = "Your card is not identified";
1368        @Deprecated
1369        public static final String LOGIN_FAIL = "Invalid account number or PIN. Please
        try again.";
1370        /** System part **/
1371        public static final String INTERNET_ERROR = "Internet connection is not stable";
1372        public static String getOverDrawnMessage(Double limit) {
1373            String msg = "Overdrawn (Insufficient funds in your account)";
1374            if (limit > 0)
1375                msg += ", your overdrawn limit is: "
1376                        + Screen.getDollarAmount(limit);
```

```
1377            return msg;
1378        }
1379        public static Vector<JLabel> getOverDrawnMessageLabels(Double limit) {
1380            Vector<JLabel> result = new Vector<JLabel>();
1381            result.add(new JLabel("Overdrawn (Insufficient funds in your account)"));
1382            if (limit > 0)
1383                result.add(new JLabel("your overdrawn limit is: "
1384                        + Screen.getDollarAmount(limit)));
1385            return result;
1386        }
1387    }
1388    ==> ./src/atm/core/ATM.java <==
1389    package atm.core;
1390    import java.util.Vector;
1391    import javax.security.auth.login.AccountNotFoundException;
1392    import bank.BankDatabase;
1393    import bank.account.Account;
1394    import bank.operation.BalanceInquiry;
1395    import bank.operation.Transaction;
1396    import bank.operation.Transfer;
1397    import bank.operation.Withdrawal;
1398    import atm.exception.CardOutException;
1399    import atm.exception.CashNotesNotSupportedException;
1400    import atm.exception.WrongInputException;
1401    import atm.gui.monitor.mainscreen.CardNotValidJPanel;
1402    import atm.gui.monitor.mainscreen.LoginJPanel;
1403    import atm.gui.monitor.mainscreen.MainMenuJPanel;
1404    import atm.gui.monitor.mainscreen.MainScreenCardJPanel;
1405    import atm.gui.monitor.mainscreen.MaxWrongTryJPanel;
1406    import atm.gui.virtualslots.cardslot.Card;
1407    import atm.gui.virtualslots.cardslot.CardInsideJPanel;
1408    import atm.gui.virtualslots.cardslot.CardSlotCardJPanel;
1409    import atm.utils.CashCount;
1410    import atm.utils.MyInputHandler;
1411    import atm.utils.MyStaticStuff;
1412    import atm.utils.MyStrings;
1413    // ATM.java
1414    // Represents an automated teller machine
1415    public class ATM {
1416        private static ATM atm = null;
1417        public static boolean userAuthenticated; // whether user is authenticated
1418        public static String currentAccountNumber; // current user's account number
1419        private Screen screen; // ATM's screen
1420        private Keypad keypad; // ATM's keypad
1421        private CashDispenser cashDispenser; // ATM's cash dispenser
1422        private BankDatabase bankDatabase; // account information database
1423        private UI ui;
1424        private int wrongCount = 0;
1425        // constants corresponding to main menu options
1426        public static final int BALANCE_INQUIRY = 1;
1427        public static final int WITHDRAWAL = 2;
1428        public static final int TRANSFER = 3;
1429        public static final int EXIT = 4;
1430        // no-argument ATM constructor initializes instance variables
1431        private ATM() {
1432            screen = new Screen(); // create screen
1433            keypad = new Keypad(screen); // create keypad
1434            cashDispenser = new CashDispenser(); // create cash dispenser
1435            bankDatabase = new BankDatabase(); // create acct info database
1436            ui = new UI(screen, bankDatabase, keypad);
```

```
1437            init();
1438        } // end no-argument ATM constructor
1439        public void init() {
1440            wrongCount = 0;
1441        }
1442        /** getters **/
1443        public static ATM getATM() {
1444            if (atm == null)
1445                atm = new ATM();
1446            return atm;
1447        }
1448        public UI getUI() {
1449            return ui;
1450        }
1451        public BankDatabase getBankDatabase() {
1452            return bankDatabase;
1453        }
1454        public CashDispenser getCashDispenser() {
1455            return cashDispenser;
1456        }
1457        public String getCurrentAccountNumber() {
1458            try {
1459                return CardInsideJPanel.getCard().accountNumber;
1460            } catch (NullPointerException e) {
1461                return "0";
1462            }
1463        }
1464        /** setters **/
1465        public void removeAuthentication() {
1466            userAuthenticated = false;
1467        }
1468        /** instance methods **/
1469        // start ATM
1470        public void run() {
1471            screen.clear();
1472            // welcome and authenticate user; perform transactions
1473            while (true) {
1474                // loop while user is not yet authenticated
1475                while (!userAuthenticated) {
1476                    screen.displayMessageLine("\nWelcome to CC Bank ATM!");
1477                    authenticateUser_old();
1478                } // end while
1479                try {
1480                    try {
1481                        try {
1482                            performTransactions();
1483                        } catch (AccountNotFoundException e) {
1484                            screen.displayMessageLine(MyStrings.ACCOUNT_NOT_FOUND);
1485                        }
1486                    } catch (WrongInputException e) {
1487                        screen.displayMessageLine(MyStrings.WRONG_INPUT);
1488                    }
1489                } // user is now
1490                catch (CardOutException e) {
1491                    userAuthenticated = false;
1492                }
1493                // authenticated
1494                userAuthenticated = false; // reset before next ATM session
1495                ATM.currentAccountNumber = "0"; // reset before next ATM session
1496                popCard();
```

```
1497                showBye();
1498            } // end while
1499        } // end method run
1500        // attempts to authenticate user against database
1501        @Deprecated
1502        private void authenticateUser_old() {
1503            // get account number from user
1504            String accountNumber = "";
1505            String pin = "";
1506            int wrongCount = 0;
1507            boolean ok;
1508            do {
1509                ok = true;
1510                try {
1511                    accountNumber = String.valueOf(keypad
1512                            .getInputInt("\nPlease enter your account number: "));
1513                } catch (WrongInputException e) {
1514                    ok = false;
1515                    wrongCount++;
1516                    screen.displayMessageLine();
1517                }
1518            } while ((wrongCount <= MyInputHandler.MAX_WRONG_INPUT) && (!ok));
1519            if (!ok) {
1520                userAuthenticated = false;
1521                return;
1522            }
1523            // end of input account number
1524            // prompt for PIN
1525            wrongCount = 0;
1526            do {
1527                try {
1528                    pin = String.valueOf(keypad.getInputInt("\nEnter your PIN: "));//    ↵
                        input
1529                                                                                    //  ↵
    PIN
1530                } catch (WrongInputException e) {
1531                    ok = false;
1532                    wrongCount++;
1533                    screen.displayMessageLine();
1534                }
1535            } while ((wrongCount <= MyInputHandler.MAX_WRONG_INPUT) && (!ok));
1536            if (!ok) {
1537                userAuthenticated = false;
1538                return;
1539            }
1540            // set userAuthenticated to boolean value returned by database
1541            try {
1542                userAuthenticated = BankDatabase.authenticateUser_old(
1543                        accountNumber, pin);
1544            } catch (AccountNotFoundException e) {
1545                System.out.println("Account not Found");
1546            }
1547            // check whether authentication succeeded
1548            if (userAuthenticated) {
1549                ATM.currentAccountNumber = accountNumber; // save user's account #
1550            } // end if
1551            else {
1552                screen.displayMessageLine("Invalid account number or PIN. Please try    ↵
                    again.");
1553                MyStaticStuff.sleep();
```

```
1554            }
1555        } // end method authenticateUser
1556        public void authenticateUser(String pin) {
1557            System.out.println("attend to login");
1558            try {
1559                // userAuthenticated =
1560                // BankDatabase.authenticateUser_old(ATM.currentAccountNumber, pin);
1561                userAuthenticated = BankDatabase.authenticateUser(pin);
1562            } catch (AccountNotFoundException e) {
1563                CardNotValidJPanel.showMe();
1564            }
1565            if (!userAuthenticated) {
1566                wrongCount++;
1567                if (wrongCount <= MyInputHandler.MAX_WRONG_INPUT) {
1568                    System.out.println("wrong pin");
1569                    LoginJPanel.showMeWrongStatic(wrongCount);
1570                } else {
1571                    System.out.println("too many wrong try");
1572                    MaxWrongTryJPanel.showMe();
1573                }
1574            } else {
1575                System.out.println("logged in");
1576                MainMenuJPanel.showMe();
1577            }
1578        }
1579        // display the main menu and perform transactions
1580        private void performTransactions() throws CardOutException,
1581                WrongInputException, AccountNotFoundException {
1582            // local variable to store transaction currently being processed
1583            Vector<Transaction> currentTransactions = null;
1584            boolean userExited = false; // user has not chosen to exit
1585            // loop while user has not chosen option to exit system
1586            while (!userExited) {
1587                // show main menu and get user selection
1588                int mainMenuSelection = -1;
1589                int wrongCount = 0;
1590                try {
1591                    mainMenuSelection = displayMainMenu();
1592                } catch (WrongInputException e) {
1593                    if (++wrongCount > MyInputHandler.MAX_WRONG_INPUT)
1594                        mainMenuSelection = EXIT;
1595                }
1596                // decide how to proceed based on user's menu selection
1597                switch (mainMenuSelection) {
1598                // user chose to perform one of three transaction types
1599                case BALANCE_INQUIRY:
1600                case WITHDRAWAL:
1601                case TRANSFER:
1602                    currentTransactions = createTransactions(mainMenuSelection);
1603                    if (currentTransactions == null) {
1604                        userExited = false;
1605                        break;
1606                    }
1607                    // execute transaction
1608                    for (Transaction currentTransaction : currentTransactions)
1609                        try {
1610                            currentTransaction.execute();
1611                        } catch (CashNotesNotSupportedException e) {
1612                            // unimplemented
1613                            System.out
```

```
1614                                    .println("Cash Notes Not Supported by this ATM");
1615                     }
1616                 if (mainMenuSelection == TRANSFER)
1617                     throw new CardOutException();
1618                 // auto finish the transaction if with WITHDRAWAL success (card
1619                 // out expection)
1620                 break;
1621             case EXIT: // user chose to terminate session
1622                 userExited = true; // this ATM session should end
1623                 break;
1624             default: // user did not enter an integer from 1-4
1625                 screen.displayMessageLine("\nYou did not enter a valid selection.
                    Please try again.");
1626                 break;
1627             } // end switch
1628             MyStaticStuff.sleep();
1629         } // end while
1630     } // end method performTransactions
1631     // display the main menu and return an input selection
1632     private int displayMainMenu() throws WrongInputException {
1633         screen.clear();
1634         if (!Account.isMyBankAccount(ATM.currentAccountNumber))
1635             screen.displayMessageLine(MyStaticStuff.getExtraChargeString());
1636         String msg = "\nMain menu:";
1637         msg += "\n1 - View my balance";
1638         msg += "\n2 - Withdraw cash";
1639         msg += "\n3 - Transfer funds";
1640         msg += "\n4 - Exit";
1641         msg += "\n\nEnter a choice: ";
1642         return keypad.getInputInt(msg); // return user's selection
1643     } // end method displayMainMenu
1644     // return object of specified Transaction subclass
1645     public Vector<Transaction> createTransactions(int type)
1646             throws CardOutException, WrongInputException,
1647             AccountNotFoundException {
1648         System.out.println("ATM:createTransactions (type:" + type + ")");
1649         // temporary Transaction variable
1650         Vector<Transaction> result = new Vector<Transaction>();
1651         // determine which type of Transaction to create
1652         switch (type) {
1653         case BALANCE_INQUIRY: // create new BalanceInquiry transaction
1654             result.add(new BalanceInquiry(this));
1655             break;
1656         case WITHDRAWAL: // create new Withdrawal transaction
1657             result.add(new Withdrawal(this));
1658             break;
1659         case TRANSFER: // create new Deposit transaction
1660             result = Transfer.transfer(this);
1661             break;
1662         } // end switch
1663         return result; // return the newly created object
1664     } // end method createTransaction
1665     /** Skipped featured (will be added in GUI) **/
1666     // instruct user to take card
1667     public void popCard() {
1668         screen.displayMessageLine(MyStrings.TAKE_CARD);
1669         MyStaticStuff.sleep();
1670     }
1671     // instruct user to take cash
1672     public void popCash(Vector<CashCount> cashPop) throws CardOutException {
```

```
1673            screen.displayMessageLine(MyStrings.TAKE_CASH + " "
1674                    + MyStaticStuff.getCashValuesStrings(cashPop));
1675            MyStaticStuff.sleep();
1676            throw new CardOutException();
1677        }
1678        @SuppressWarnings("deprecation")
1679        public void showBye() {
1680            screen.displayMessageLine("\n" + MyStrings.BYE);
1681            MyStaticStuff.sleep();
1682        }
1683        public static void readCard(Card card) {
1684            System.out.println("reading inserted card:" + card.accountNumber);
1685            MainScreenCardJPanel
1686                    .switchToCardStatic(MainScreenCardJPanel.STRING_READCARD);
1687            (new WaitReadCard(card)).start();
1688        }
1689        public static void checkCard(Card card) {
1690            try {
1691                ATM.currentAccountNumber = String.valueOf(Integer
1692                        .parseInt(card.accountNumber));
1693                System.out.println("the card [" + ATM.currentAccountNumber
1694                        + "] is valid");
1695                LoginJPanel.showMeStatic();
1696            } catch (NumberFormatException e) {
1697                CardNotValidJPanel.showMe();
1698            }
1699        }
1700        public static void popCardStatic() {
1701            MainScreenCardJPanel
1702                    .switchToCardStatic(MainScreenCardJPanel.STRING_CARD_NOT_VALID);
1703            CardSlotCardJPanel.popCardStatic();
1704        }
1705        /** private class **/
1706        private static class WaitReadCard extends Thread {
1707            private Card card;
1708            public WaitReadCard(Card card) {
1709                this.card = card;
1710            }
1711            @Override
1712            public void run() {
1713                try {
1714                    Thread.sleep(2000);
1715                } catch (InterruptedException e) {
1716                }
1717                ATM.checkCard(card);
1718            }
1719        }
1720        /** static connectors to instance methods **/
1721        public static void initStatic() {
1722            ATM.atm = new ATM();
1723            // user is not authenticated to start/restart
1724            ATM.userAuthenticated = false;
1725            // no current account number to start/restart
1726            ATM.currentAccountNumber = "0";
1727            if (CardSlotCardJPanel.hasCard())
1728                currentAccountNumber = CardInsideJPanel.getCard().accountNumber;
1729            atm.init();
1730        }
1731    } // end class ATM
1732    ==> ./src/atm/core/UI.java <==
```

```
package atm.core;
import bank.BankDatabase;
public class UI {
    public BankDatabase bankDatabase;
    public Screen screen;
    public Keypad keypad;
    public UI(Screen screen, BankDatabase bankDatabase, Keypad keypad) {
        this.screen = screen;
        this.bankDatabase = bankDatabase;
        this.keypad = keypad;
    }
}
==> ./src/atm/core/CashDispenser.java <==
package atm.core;

import java.util.Vector;

import atm.exception.CashNotEnoughException;
import atm.exception.CashNotesNotSupportedException;
import atm.utils.CashCount;

// CashDispenser.java
// Represents the cash dispenser of the ATM

public class CashDispenser {
    // number of cash bills remaining
    public static Vector<CashCount> cashCounts = new Vector<CashCount>();
    public static Vector<CashCount> lastTransaction = null;

    // no-argument CashDispenser constructor initializes count to default
    public static void init() {
        // set count attribute to default
        cashCounts = new Vector<CashCount>();
        cashCounts.add(new CashCount(100, 15));
        cashCounts.add(new CashCount(500, 8));
        cashCounts.add(new CashCount(1000, 2));
    } // end CashDispenser constructor

    /**
     * static methods
     *
     * @throws CashNotesNotSupportedException
     **/
    // simulates dispensing of specified amount of cash
    public static Vector<CashCount> dispenseCash(int amountRemain)
            throws CashNotEnoughException, CashNotesNotSupportedException {
        Vector<CashCount> result = new Vector<CashCount>();
        if (!isSufficientCashAvailable(amountRemain))
            throw new CashNotEnoughException();
        for (int i = cashCounts.size() - 1; i >= 0; i--) {
            result.add(new CashCount(cashCounts.get(i).getValue(), 0));
            while ((amountRemain >= cashCounts.get(i).getValue())
                    && (cashCounts.get(i).getCount() > 0)) {
                amountRemain -= cashCounts.get(i).getValue();
                cashCounts.get(i).remove(1);
                result.get(result.size() - 1).add(1);
            }
        }
        if (amountRemain != 0)
            throw new CashNotesNotSupportedException(amountRemain, result);
```

```
1793            lastTransaction = result;
1794            return result;
1795        } // end method dispenseCash
1796
1797        public static void rollback() {
1798            System.out.println("CashDispenser: rollback (cash not taken)");
1799            if (lastTransaction == null)
1800                return;
1801            for (CashCount lastTransactionIterator : lastTransaction) {
1802                for (CashCount cashCountIterator : cashCounts) {
1803                    if (lastTransactionIterator.getValue() == cashCountIterator
1804                            .getValue())
1805                        cashCountIterator.add(lastTransactionIterator.getCount());
1806                }
1807            }
1808        }
1809
1810        // comfirm pop cash
1811        public static void commit() {
1812            System.out.println("CashDispenser: commit (confirm cash taken)");
1813            lastTransaction = null;
1814        }
1815
1816        public static Vector<CashCount> getCash() {
1817            return lastTransaction;
1818        }
1819
1820        // indicates whether cash dispenser can dispense desired amount
1821        public static boolean isSufficientCashAvailable(double amountRequired) {
1822            return (getAmount() >= amountRequired);
1823        } // end method isSufficientCashAvailable
1824
1825        public static double getAmount() {
1826            double amount = 0;
1827            for (CashCount cashCount : cashCounts)
1828                amount += cashCount.getValue() * cashCount.getCount();
1829            return amount;
1830        }
1831
1832    } // end class CashDispenser
1833
1834    ==> ./src/atm/core/Keypad.java <==
1835    package atm.core;
1836    // Keypad.java
1837    // Represents the keypad of the ATM
1838    import java.util.Scanner; // program uses Scanner to obtain user input
1839    import atm.exception.WrongInputException;
1840    import atm.utils.MyInputHandler;
1841    import atm.utils.MyStaticStuff;
1842    public class Keypad {
1843        private Screen screen;
1844        private Scanner input; // reads data from the command line
1845        // no-argument constructor initializes the Scanner
1846        public Keypad(Screen screen) {
1847            this.screen = screen;
1848            input = new Scanner(System.in);
1849        } // end no-argument Keypad constructor
1850        // return an integer value entered by user
1851        public int getInputInt(String msg) throws WrongInputException {
1852            int result = 0;
```

```
1853            int wrongCount = 0;
1854            boolean ok;
1855            do {
1856                ok = true;
1857                try {
1858                    screen.displayMessage(msg);
1859                    result = Integer.valueOf(input.next());
1860                } catch (NumberFormatException e) {
1861                    screen.displayMessageLine("Please input an integer only.");
1862                    wrongCount++;
1863                    ok = false;
1864                    MyStaticStuff.sleep();
1865                }
1866            } while ((wrongCount <= MyInputHandler.MAX_WRONG_INPUT) && (!ok));
1867            if (!ok)
1868                throw new WrongInputException();
1869            else
1870                return result; // we don't assume that user enters an integer
1871        } // end method getInput
1872        // return an positive integer value entered by user
1873        public int getInputIntPositive(String msg) throws WrongInputException {
1874            int result = 0;
1875            int wrongCount = 0;
1876            boolean ok;
1877            do {
1878                ok = true;
1879                try {
1880                    screen.displayMessage(msg);
1881                    result = Integer.valueOf(input.next());
1882                    if (result <= 0)
1883                        throw new NumberFormatException();
1884                } catch (NumberFormatException e) {
1885                    screen.displayMessageLine("Please input an positive integer only.");
1886                    wrongCount++;
1887                    ok = false;
1888                    MyStaticStuff.sleep();
1889                }
1890            } while ((wrongCount <= MyInputHandler.MAX_WRONG_INPUT) && (!ok));
1891            if (!ok)
1892                throw new WrongInputException();
1893            else
1894                return result; // we don't assume that user enters an integer
1895        } // end method getInputIntPositive
1896        public double getInputDouble(String msg) throws WrongInputException {
1897            double result = 0;
1898            int wrongCount = 0;
1899            boolean ok;
1900            do {
1901                ok = true;
1902                try {
1903                    screen.displayMessage(msg);
1904                    result = Double.valueOf(input.next());
1905                } catch (NumberFormatException e) {
1906                    System.out.println("Please input an real number only.");
1907                    wrongCount++;
1908                    ok = false;
1909                    MyStaticStuff.sleep();
1910                }
1911            } while ((wrongCount <= MyInputHandler.MAX_WRONG_INPUT) && (!ok));
1912            if (!ok)
```

```
            throw new WrongInputException();
        else
            return result;
    }
    public double getInputDoublePositive(String msg) throws WrongInputException {
        double result = 0;
        int wrongCount = 0;
        boolean ok;
        do {
            ok = true;
            try {
                screen.displayMessage(msg);
                result = Double.valueOf(input.next());
                if (result <= 0)
                    throw new NumberFormatException();
            } catch (NumberFormatException e) {
                System.out.println("Please input a positive real number only.");
                wrongCount++;
                ok = false;
                MyStaticStuff.sleep();
            }
        } while ((wrongCount <= MyInputHandler.MAX_WRONG_INPUT) && (!ok));
        if (!ok)
            throw new WrongInputException();
        else
            return result;
    }
} // end class Keypad
==> ./src/atm/core/Screen.java <==
package atm.core;
import atm.utils.MyStrings;
// Screen.java
// Represents the screen of the ATM
public class Screen {
    public void clear() {
        for (int i = 0; i < 5; i++)
            displayMessageLine();
    }
    // displays a message without a carriage return
    public void displayMessage(String message) {
        System.out.print(message);
    } // end method displayMessage
    // display a message with a carriage return
    public void displayMessageLine(String message) {
        System.out.println(message);
    }
    public void displayMessageLine(int i) {
        displayMessageLine(i + "");
    }
    public void displayMessage(double d) {
        displayMessageLine(d + "");
    }
    public void displayMessageLine() {
        displayMessageLine("");
    } // end method displayMessageLine
    // display a dollar amount
    public void displayDollarAmount(double amount) {
        displayMessage(Screen.getDollarAmount(amount));
    } // end method displayDollarAmount
    // display a dollar amount
```

```
    public void displayDollarAmount(int amount) {
        displayMessage(Screen.getDollarAmount(amount));
    } // end method displayDollarAmount
    // return a dollar amount as String
    public static String getDollarAmount(double amount) {
        return String.format(MyStrings.DOLLAR_SIGN + "%,.2f", amount);
    } // end method displayDollarAmount
    // return a dollar amount as String
    public static String getDollarAmount(int amount) {
        return MyStrings.DOLLAR_SIGN + amount;
    } // end method displayDollarAmount
} // end class Screen
==> ./src/atm/gui/MyGUISettings.java <==
package atm.gui;
import java.awt.Color;
import java.awt.Font;
public class MyGUISettings {
    public static int MONITOR_TOP_MARGIN = 75;
    public static int MONITOR_FRAME_WIDTH = 600;
    public static int MONITOR_FRAME_HEIGHT = 400;
    public static int VIRTUAL_SLOTS_FRAME_WIDTH = 400;
    public static int VIRTUAL_SLOTS_FRAME_HEIGHT = 300;
    public static int SIDE_BUTTON_MARGIN = 75;
    public static int SIDE_BUTTON_SIZE = 50;
    public static int CARD_IMAGE_WIDTH = 128;
    public static int CARD_IMAGE_HEIGHT = 64;
    public static int FUNCTION_BUTTON_WIDTH = 64;
    public static int FUNCTION_BUTTON_HEIGHT = 32;
    public static Font getFont(int fontSize) {
        return new Font("Arial", Font.PLAIN, fontSize);
    }
    public static Font getBoldFont(int fontSize) {
        return new Font("Arial", Font.BOLD, fontSize);
    }
    public static Color getATMScreenBackGroundColor() {
        return new Color(135, 206, 250);
    }
    public static Color getATMShellColor() {
        return new Color(128, 128, 128);
    }
}
==> ./src/atm/gui/monitor/MonitorJFrame.java <==
package atm.gui.monitor;
import javax.swing.JFrame;
import atm.gui.MyGUISettings;
import atm.gui.monitor.mainscreen.CashNotEnoughJPanel;
import atm.gui.monitor.mainscreen.MainMenuJPanel;
import atm.gui.monitor.mainscreen.MainScreenCardJPanel;
import atm.gui.monitor.mainscreen.TransferJPanel;
import atm.gui.monitor.mainscreen.ViewBalanceJPanel;
import atm.gui.monitor.mainscreen.WithDrawalJPanel;
import atm.gui.monitor.sidebuttons.LeftSideButtonsJPanel;
import atm.gui.monitor.sidebuttons.RightSideButtonsJPanel;
import atm.gui.virtualslots.cardslot.CardInsideJPanel;
import atm.gui.virtualslots.cardslot.CardSlotCardJPanel;
import java.awt.BorderLayout;
import java.net.MalformedURLException;
public class MonitorJFrame extends JFrame {
    /***/
    private static final long serialVersionUID = 1L;
```

```
2033        private LeftSideButtonsJPanel leftSideButtonsJPanel;
2034        private RightSideButtonsJPanel rightSideButtonsJPanel;
2035        private MainScreenCardJPanel mainScreenJPanel;
2036        public static String STATE = "";
2037        public MonitorJFrame() throws MalformedURLException {
2038            setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
2039            setTitle("ATM Monitor");
2040            setVisible(false);
2041            setResizable(false);
2042            getContentPane().setLayout(new BorderLayout(0, 0));
2043            leftSideButtonsJPanel = new LeftSideButtonsJPanel();
2044            getContentPane().add(leftSideButtonsJPanel, BorderLayout.WEST);
2045            rightSideButtonsJPanel = new RightSideButtonsJPanel();
2046            getContentPane().add(rightSideButtonsJPanel, BorderLayout.EAST);
2047            mainScreenJPanel = new MainScreenCardJPanel();
2048            getContentPane().add(mainScreenJPanel, BorderLayout.CENTER);
2049            mainScreenJPanel.setBackground(MyGUISettings
2050                    .getATMScreenBackGroundColor());
2051        }
2052        public void calcBounds(int w, int h, int s) {
2053            setVisible(true);
2054            setBounds(10, 10, w + s * 2, h);
2055        }
2056        @Override
2057        public void dispose() {
2058            super.dispose();
2059        }
2060        public static void sideButtonClick(String command) {
2061            System.out.println("Side button clicked: " + command);
2062            switch (STATE) {
2063            case MainScreenCardJPanel.STRING_MAIN_MENU:
2064                System.out.println("MainScreenCardJPanel.STRING_MAIN_MENU:");
2065                switch (command) {
2066                case MainScreenCardJPanel.STRING_VIEW_BALANCE:
2067                    ViewBalanceJPanel.showMeStatic();
2068                    break;
2069                case MainScreenCardJPanel.STRING_TAKE_CARD:
2070                    CardSlotCardJPanel.popCardStatic();
2071                    break;
2072                case MainScreenCardJPanel.STRING_WITHDRAWAL:
2073                    WithDrawalJPanel.showMeStatic();
2074                    break;
2075                case MainScreenCardJPanel.STRING_TRANSFER:
2076                    TransferJPanel.showMeStatic();
2077                    break;
2078                }
2079                break;
2080            case MainScreenCardJPanel.STRING_VIEW_BALANCE:
2081                switch (command) {
2082                case ViewBalanceJPanel.STRING_MAIN_MENU:
2083                    MainMenuJPanel.showMe();
2084                    break;
2085                case ViewBalanceJPanel.STRING_TAKE_CARD:
2086                    CardSlotCardJPanel.popCardStatic();
2087                    break;
2088                }
2089                break;
2090            case MainScreenCardJPanel.STRING_WITHDRAWAL:
2091                switch (command) {
2092                case WithDrawalJPanel.STRING_MAIN_MENU:
```

```
                MainMenuJPanel.showMe();
                break;
            case WithDrawalJPanel.STRING_TAKE_CARD:
                CardSlotCardJPanel.popCardStatic();
                break;
            default:
                WithDrawalJPanel.sideButtonClickStatic(command);
                break;
            }
        case MainScreenCardJPanel.STRING_CASH_NOT_ENOUGH:
            switch (command) {
            case CashNotEnoughJPanel.STRING_MAIN_MENU:
                MainMenuJPanel.showMe();
                break;
            case CashNotEnoughJPanel.STRING_TAKE_CARD:
                CardSlotCardJPanel.popCardStatic();
                break;
            }
        case MainScreenCardJPanel.STRING_TRANSFER: {
        }
        default:
            break;
        }
    }
    public static void returnButtonClick() {
        if (!CardInsideJPanel.hasCard())
            return;
        switch (STATE) {
        case MainScreenCardJPanel.STRING_MAIN_MENU:
            CardSlotCardJPanel.popCardStatic();
            break;
        case MainScreenCardJPanel.STRING_VIEW_BALANCE:
            MainMenuJPanel.showMe();
            break;
        case MainScreenCardJPanel.STRING_TRANSFER:
            MainMenuJPanel.showMe();
            break;
        default:
            CardSlotCardJPanel.popCardStatic();
            break;
        }
    }
}
==> ./src/atm/gui/monitor/sidebuttons/SideButtons.java <==
package atm.gui.monitor.sidebuttons;
import java.awt.Image;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import javax.swing.ImageIcon;
import atm.gui.MyGUISettings;
import atm.gui.monitor.MonitorJFrame;
import atm.gui.virtualslots.cashdispenser.notes.CashNote;
import atm.utils.MyURLs;
public class SideButtons implements CashNote {
    public static ImageIcon triangle_point_left_imageIcon;
    public static ImageIcon triangle_point_right_imageIcon;
    public static String[] commands = new String[8];
    public static void init() throws MalformedURLException {
        triangle_point_right_imageIcon = new ImageIcon(new ImageIcon(new URL(
```

```
                MyURLs.IMAGE_TRIANGLE_POINT_RIGHT)).getImage()
                .getScaledInstance(MyGUISettings.SIDE_BUTTON_SIZE,
                        MyGUISettings.SIDE_BUTTON_SIZE, Image.SCALE_SMOOTH));
        triangle_point_left_imageIcon = new ImageIcon(new ImageIcon(new URL(
                MyURLs.IMAGE_TRIANGLE_POINT_LEFT)).getImage()
                .getScaledInstance(MyGUISettings.SIDE_BUTTON_SIZE,
                        MyGUISettings.SIDE_BUTTON_SIZE, Image.SCALE_SMOOTH));
    }
    public static void click(int id) {
        MonitorJFrame.sideButtonClick(commands[id - 1]);
    }
    @Override
    public void fetchImage() throws IOException {
        System.out.println("fetching images for layout");
        init();
    }
}
==> ./src/atm/gui/monitor/sidebuttons/LeftSideButtonsJPanel.java <==
package atm.gui.monitor.sidebuttons;
import java.awt.GridLayout;
import java.util.Vector;
import javax.swing.JButton;
import javax.swing.JPanel;
import atm.gui.MyGUISettings;
import javax.swing.BoxLayout;
import java.awt.Component;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.Box;
public class LeftSideButtonsJPanel extends JPanel {
    /***/
    private static final long serialVersionUID = 1L;
    public Vector<JButton> buttons;
    public LeftSideButtonsJPanel() {
        setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
        setBackground(MyGUISettings.getATMShellColor());
        Component topVerticalStrut = Box.createVerticalStrut(75);
        add(topVerticalStrut);
        JPanel centerPanel = new JPanel();
        add(centerPanel);
        centerPanel.setLayout(new GridLayout(4, 1, 0, 0));
        centerPanel.setBackground(MyGUISettings.getATMShellColor());
        Component bottomVerticalStrut = Box.createVerticalStrut(25);
        add(bottomVerticalStrut);
        buttons = new Vector<JButton>();
        for (int i = 0; i < 4; i++) {
            JButton button = new JButton(SideButtons.triangle_point_right_imageIcon);
            buttons.add(button);
            button.addActionListener(getActionListener(i));
            centerPanel.add(button);
        }
    }
    private ActionListener getActionListener(final int id) {
        return new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                SideButtons.click(id * 2 + 1);
            }
        };
    }
```

```
}
==> ./src/atm/gui/monitor/sidebuttons/RightSideButtonsJPanel.java <==
package atm.gui.monitor.sidebuttons;
import java.awt.GridLayout;
import java.util.Vector;
import javax.swing.JPanel;
import javax.swing.JButton;
import atm.gui.MyGUISettings;
import javax.swing.BoxLayout;
import java.awt.Component;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.Box;
public class RightSideButtonsJPanel extends JPanel {
    /***/
    private static final long serialVersionUID = 1L;
    public Vector<JButton> buttons;
    public RightSideButtonsJPanel() {
        setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
        setBackground(MyGUISettings.getATMShellColor());
        Component topVerticalStrut = Box.createVerticalStrut(75);
        add(topVerticalStrut);
        JPanel centerPanel = new JPanel();
        add(centerPanel);
        centerPanel.setLayout(new GridLayout(4, 1, 0, 0));
        centerPanel.setBackground(MyGUISettings.getATMShellColor());
        Component bottomVerticalStrut = Box.createVerticalStrut(25);
        add(bottomVerticalStrut);
        buttons = new Vector<JButton>();
        for (int i = 0; i < 4; i++) {
            JButton button = new JButton(SideButtons.triangle_point_left_imageIcon);
            buttons.add(button);
            centerPanel.add(button);
            button.addActionListener(getActionListener(i));
        }
    }
    private ActionListener getActionListener(final int id) {
        return new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                SideButtons.click((id + 1) * 2);
            }
        };
    }
}
==> ./src/atm/gui/monitor/mainscreen/ReadCardJPanel.java <==
package atm.gui.monitor.mainscreen;
import javax.swing.JPanel;
import javax.swing.JLabel;
import atm.gui.MyGUISettings;
import javax.swing.Box;
import javax.swing.BoxLayout;
import java.awt.Component;
public class ReadCardJPanel extends JPanel {
    /***/
    private static final long serialVersionUID = 1L;
    /** Create the panel.*/
    public ReadCardJPanel() {
        setLayout(new BoxLayout(this, BoxLayout.X_AXIS));
        Component glue = Box.createHorizontalGlue();
```

```
2273            add(glue);
2274            JLabel label = new JLabel("Reading Your Card...");
2275            add(label);
2276            label.setAlignmentX(Component.CENTER_ALIGNMENT);
2277            label.setFont(MyGUISettings.getFont(26));
2278            Component glue_1 = Box.createHorizontalGlue();
2279            add(glue_1);
2280        }
2281    }
2282    ==> ./src/atm/gui/monitor/mainscreen/MainScreenCardJPanel.java <==
2283    package atm.gui.monitor.mainscreen;
2284    import java.util.Vector;
2285    import java.awt.Component;
2286    import myutils.gui.cardlayout.AbstractCardJPanel;
2287    import atm.gui.MyGUISettings;
2288    import atm.gui.monitor.MonitorJFrame;
2289    public class MainScreenCardJPanel extends AbstractCardJPanel {
2290        /***/
2291        private static final long serialVersionUID = 1L;
2292        private static Vector<MainScreenCardJPanel> contents = new Vector<
          MainScreenCardJPanel>();
2293        public static final String STRING_WELCOME = "Welcome";
2294        public static final String STRING_READCARD = "Read Card";
2295        public static final String STRING_CARD_NOT_VALID = "Card Not Valid";
2296        public static final String STRING_LOGIN = "Login";
2297        public static final String STRING_MAIN_MENU = "Main Menu";
2298        public static final String STRING_VIEW_BALANCE = MainMenuJPanel.
          STRING_VIEW_BALANCE;
2299        public static final String STRING_TAKE_CARD = "Take Card";
2300        public static final String STRING_BYE = "Bye";
2301        public static final String STRING_MAX_WRONG_TRY = "Max Wrong Try";
2302        public static final String STRING_WITHDRAWAL = MainMenuJPanel.
          STRING_WITHDRAW_CASH;
2303        public static final String STRING_TRANSFER = MainMenuJPanel.
          STRING_TRANSFER_FUNDS;
2304        public static final String STRING_TAKE_CASH = "Take Cash";
2305        public static final String STRING_OVERDRAWN = "Overdrawn";
2306        public static final String STRING_CASH_NOT_ENOUGH = "Cash Not Enough";
2307        public static final String STRING_TRANSFER_RECEIVER_ACCOUNT_NOT_FOUND =
          "Transfer Receiver Account Not Found";
2308        public static final String STRING_TRANSFER_SAME_ACCOUNT = "Transfer Same
          Account";
2309        public static final String STRING_TRANSFER_SUCCESS = "Trasfer Success";
2310        public static final String STRING_CASH_REQUIRED_NOT_SUPPORTED = "Cash Required
          Not Supported";
2311        private ViewBalanceJPanel viewBalanceJPanel;
2312        public MainScreenCardJPanel() {
2313            super();
2314            contents.add(this);
2315        }
2316        @Override
2317        protected void myInit() {
2318            addToCards(new WelcomeJPanel(), STRING_WELCOME);
2319            addToCards(new ReadCardJPanel(), STRING_READCARD);
2320            addToCards(new CardNotValidJPanel(), STRING_CARD_NOT_VALID);
2321            addToCards(new LoginJPanel(), STRING_LOGIN);
2322            addToCards(new MainMenuJPanel(), STRING_MAIN_MENU);
2323            viewBalanceJPanel = new ViewBalanceJPanel();
2324            addToCards(new MaxWrongTryJPanel(), STRING_MAX_WRONG_TRY);
2325            addToCards(new TakeCardJPanel(), STRING_TAKE_CARD);
```

```
2326            addToCards(new ByeJPanel(), STRING_BYE);
2327            addToCards(new WithDrawalJPanel(), STRING_WITHDRAWAL);
2328            addToCards(new TransferJPanel(), STRING_TRANSFER);
2329            addToCards(new TakeCashJPanel(), STRING_TAKE_CASH);
2330            addToCards(new OverdrawnJPanel(), STRING_OVERDRAWN);
2331            addToCards(new CashNotEnoughJPanel(), STRING_CASH_NOT_ENOUGH);
2332            addToCards(new TransferReceiverAccountNotFoundJPanel(),
2333                    STRING_TRANSFER_RECEIVER_ACCOUNT_NOT_FOUND);
2334            addToCards(new TransferSameAccountJPanel(),
2335                    STRING_TRANSFER_SAME_ACCOUNT);
2336            addToCards(new TransferSuccessJPanel(), STRING_TRANSFER_SUCCESS);
2337            addToCards(new CashRequiredNotSupportedJPanel(),
2338                    STRING_CASH_REQUIRED_NOT_SUPPORTED);
2339            // switchToCard(STRING_WELCOME);
2340            WelcomeJPanel.showMeStatic();
2341        }
2342        public void renewViewBalanceJPanel() {
2343            cardLayout.removeLayoutComponent(viewBalanceJPanel);
2344            viewBalanceJPanel = new ViewBalanceJPanel();
2345            addToCards(viewBalanceJPanel, STRING_VIEW_BALANCE);
2346            viewBalanceJPanel.loadinfo();
2347        }
2348        @Override
2349        public void switchToCard(String label) {
2350            if (label.equals(STRING_VIEW_BALANCE))
2351                renewViewBalanceJPanel();
2352            MonitorJFrame.STATE = label;
2353            super.switchToCard(label);
2354        }
2355        /** static connector to instance stuff **/
2356        public static void switchToCardStatic(String label) {
2357            for (MainScreenCardJPanel content : contents)
2358                content.switchToCard(label);
2359        }
2360        @Override
2361        public void addToCards(Component component, String label) {
2362            super.addToCards(component, label);
2363            component.setBackground(MyGUISettings.getATMScreenBackGroundColor());
2364        }
2365    }
2366    ==> ./src/atm/gui/monitor/mainscreen/AvailableCashNotesJPanel.java <==
2367    package atm.gui.monitor.mainscreen;
2368    import java.util.Vector;
2369    import javax.swing.JPanel;
2370    import javax.swing.JLabel;
2371    import atm.core.CashDispenser;
2372    import atm.gui.MyGUISettings;
2373    import atm.gui.virtualslots.cashdispenser.notes.CashNote100;
2374    import atm.gui.virtualslots.cashdispenser.notes.CashNote1000;
2375    import atm.gui.virtualslots.cashdispenser.notes.CashNote500;
2376    import atm.utils.CashCount;
2377    import webs.layout.WrapLayout;
2378    public class AvailableCashNotesJPanel extends JPanel {
2379        /***/
2380        private static final long serialVersionUID = 1L;
2381        private static Vector<AvailableCashNotesJPanel> contents = new Vector<
            AvailableCashNotesJPanel>();
2382        private JLabel label;
2383        Vector<JLabel> cashNoteJLabels;
2384        public AvailableCashNotesJPanel() {
```

```
2385            contents.add(this);
2386            setBackground(MyGUISettings.getATMScreenBackGroundColor());
2387            // setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
2388            setLayout(new WrapLayout());
2389            label = new JLabel();
2390            add(label);
2391            cashNoteJLabels = new Vector<JLabel>();
2392            myUpdate();
2393        }
2394        public void myUpdate() {
2395            cashNoteJLabels.removeAllElements();
2396            for (CashCount cashCount : CashDispenser.cashCounts) {
2397                if (cashCount.getCount() > 0) {
2398                    switch (cashCount.getValue()) {
2399                    case 100:
2400                        cashNoteJLabels.add(CashNote100.jLabel);
2401                        break;
2402                    case 500:
2403                        cashNoteJLabels.add(CashNote500.jLabel);
2404                        break;
2405                    case 1000:
2406                        cashNoteJLabels.add(CashNote1000.jLabel);
2407                        break;
2408                    }
2409                }
2410            }
2411            if (cashNoteJLabels.size() == 0) {
2412                label.setText("This ATM does not provide any cash notes");
2413            } else {
2414                label.setText("This ATM provide the following types of cash note");
2415            }
2416            // label.setFont(MyGUISettings.getFont(26));
2417            label.setFont(MyGUISettings.getBoldFont(24));
2418            removeAll();
2419            add(label);
2420            for (JLabel cashNoteJLabel : cashNoteJLabels)
2421                add(cashNoteJLabel);
2422        }
2423        public static void myUpdateStatic() {
2424            for (AvailableCashNotesJPanel cashAvailableJPanel : contents) {
2425                cashAvailableJPanel.myUpdate();
2426            }
2427        }
2428    }
2429    ==> ./src/atm/gui/monitor/mainscreen/ViewBalanceJPanel.java <==
2430    package atm.gui.monitor.mainscreen;
2431    import javax.security.auth.login.AccountNotFoundException;
2432    import javax.swing.JPanel;
2433    import atm.core.ATM;
2434    import atm.exception.CardOutException;
2435    import atm.exception.CashNotesNotSupportedException;
2436    import atm.exception.WrongInputException;
2437    import atm.gui.MyGUISettings;
2438    import atm.gui.monitor.MonitorJFrame;
2439    import atm.gui.monitor.sidebuttons.SideButtons;
2440    import atm.gui.virtualslots.cardslot.CardSlotCardJPanel;
2441    import atm.utils.MyImages;
2442    import bank.operation.Transaction;
2443    import java.awt.Dimension;
2444    import java.awt.GridLayout;
```

```
2445  import java.awt.Button;
2446  import java.awt.Font;
2447  import java.awt.Component;
2448  import javax.swing.Box;
2449  import javax.swing.BoxLayout;
2450  import javax.swing.JLabel;
2451  import javax.swing.JTextArea;
2452  import java.awt.BorderLayout;
2453  import java.util.ConcurrentModificationException;
2454  import java.util.Vector;
2455  public class ViewBalanceJPanel extends JPanel {
2456      /***/
2457      private static final long serialVersionUID = 1L;
2458      private static Vector<ViewBalanceJPanel> contents = new Vector<
          ViewBalanceJPanel>();
2459      public static final String STRING_MAIN_MENU = "Main Menu";
2460      public static final String STRING_TAKE_CARD = "Take Card";
2461      public static final String[] commands = { "", "", "", "", STRING_MAIN_MENU,
2462              STRING_TAKE_CARD, STRING_MAIN_MENU, STRING_TAKE_CARD };
2463      GUIPrinter guiPrinter;
2464      private JTextArea text;
2465      public ViewBalanceJPanel() {
2466          contents.add(this);
2467          setBackground(MyGUISettings.getATMScreenBackGroundColor());
2468          setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
2469          JPanel topPanel = new JPanel();
2470          add(topPanel);
2471          topPanel.setLayout(new BoxLayout(topPanel, BoxLayout.X_AXIS));
2472          JLabel label = new JLabel(MyImages.viewBalance);
2473          topPanel.add(label);
2474          label.setAlignmentX(0.5f);
2475          JPanel contentPanel = new JPanel();
2476          contentPanel.setBackground(MyGUISettings.getATMScreenBackGroundColor());
2477          add(contentPanel);
2478          contentPanel.setLayout(new GridLayout(2, 1, 0, 0));
2479          JPanel infoPanel1 = new JPanel();
2480          contentPanel.add(infoPanel1);
2481          infoPanel1.setBackground(MyGUISettings.getATMScreenBackGroundColor());
2482          infoPanel1.setLayout(new BorderLayout(0, 0));
2483          text = new JTextArea();
2484          infoPanel1.add(text);
2485          text.setBackground(MyGUISettings.getATMScreenBackGroundColor());
2486          text.setFont(MyGUISettings.getFont(24));
2487          text.setSize(400, 150);
2488          infoPanel1.setSize(400, 150);
2489          text.setPreferredSize(new Dimension(400, 150));
2490          infoPanel1.setPreferredSize(new Dimension(40, 150));
2491          JPanel strucPanel = new JPanel();
2492          contentPanel.add(strucPanel);
2493          strucPanel.setBackground(MyGUISettings.getATMScreenBackGroundColor());
2494          strucPanel.setLayout(new GridLayout(2, 1, 0, 0));
2495          JPanel menuPanel1 = new JPanel();
2496          strucPanel.add(menuPanel1);
2497          menuPanel1.setLayout(new GridLayout(1, 2, 0, 0));
2498          menuPanel1.setBackground(MyGUISettings.getATMScreenBackGroundColor());
2499          Button button_1 = new Button(STRING_MAIN_MENU);
2500          menuPanel1.add(button_1);
2501          button_1.setFont(new Font("Arial", Font.PLAIN, 26));
2502          button_1.setBackground(MyGUISettings.getATMScreenBackGroundColor());
2503          Button button_2 = new Button(STRING_TAKE_CARD);
```

```
2504            menuPanel1.add(button_2);
2505            button_2.setFont(new Font("Arial", Font.PLAIN, 26));
2506            button_2.setBackground(MyGUISettings.getATMScreenBackGroundColor());
2507            JPanel menuPanel2 = new JPanel();
2508            strucPanel.add(menuPanel2);
2509            menuPanel2.setBackground(MyGUISettings.getATMScreenBackGroundColor());
2510            menuPanel2.setLayout(new GridLayout(1, 2, 0, 0));
2511            Component verticalStrut_1 = Box.createVerticalStrut(25);
2512            add(verticalStrut_1);
2513            guiPrinter = new GUIPrinter(text);
2514        }
2515        public void loadinfo() {
2516            text.setText("");
2517            try {
2518                Vector<Transaction> currentTransactions;
2519                currentTransactions = ATM.getATM().createTransactions(
2520                        ATM.BALANCE_INQUIRY);
2521                if (currentTransactions == null) {
2522                    MainMenuJPanel.showMe();
2523                }
2524                // execute transaction
2525                guiPrinter.start();
2526                for (Transaction currentTransaction : currentTransactions)
2527                    currentTransaction.execute();
2528            } catch (AccountNotFoundException e) {
2529                CardNotValidJPanel.showMe();
2530            } catch (CardOutException e) {
2531                CardSlotCardJPanel.popCardStatic();
2532            } catch (WrongInputException e) {
2533                MaxWrongTryJPanel.showMe();
2534            } catch (CashNotesNotSupportedException e) {
2535                // impossible
2536            }
2537            guiPrinter.stop();
2538        }
2539        private void showMe() {
2540            loadinfo();
2541            MonitorJFrame.STATE = MainScreenCardJPanel.STRING_VIEW_BALANCE;
2542            SideButtons.commands = ViewBalanceJPanel.commands;
2543            MainScreenCardJPanel
2544                    .switchToCardStatic(MainScreenCardJPanel.STRING_VIEW_BALANCE);
2545        }
2546        public static void showMeStatic() {
2547            try {
2548                // contents.removeAllElements();
2549                // new ViewBalanceJPanel();
2550                for (ViewBalanceJPanel content : contents) {
2551                    content.showMe();
2552                }
2553            } catch (ConcurrentModificationException e) {
2554                // this is expected to happen normally
2555            }
2556        }
2557    }
2558    ==> ./src/atm/gui/monitor/mainscreen/TakeCardJPanel.java <==
2559    package atm.gui.monitor.mainscreen;
2560    import javax.swing.JPanel;
2561    import javax.swing.JLabel;
2562    import java.awt.Font;
2563    import javax.swing.BoxLayout;
```

```
import java.awt.Component;
import javax.swing.Box;
import atm.gui.MyGUISettings;
import atm.gui.monitor.sidebuttons.SideButtons;
import myutils.Utils;
public class TakeCardJPanel extends JPanel {
    /***/
    private static final long serialVersionUID = 1L;
    private static JLabel codeLabel;
    public static final String[] commands = { "", "", "", "", "", "", "", "" };
    /** Create the panel.*/
    public TakeCardJPanel() {
        setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
        setBackground(MyGUISettings.getATMScreenBackGroundColor());
        Component verticalGlue_1 = Box.createVerticalGlue();
        add(verticalGlue_1);
        JPanel panel = new JPanel();
        add(panel);
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
        panel.setBackground(MyGUISettings.getATMScreenBackGroundColor());
        JLabel lblPleaseTakeYour = new JLabel("Please Take your Card");
        panel.add(lblPleaseTakeYour);
        lblPleaseTakeYour.setAlignmentX(Component.CENTER_ALIGNMENT);
        lblPleaseTakeYour.setFont(new Font("Arial", Font.PLAIN, 26));
        codeLabel = new JLabel("Reference Code: ");
        codeLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
        panel.add(codeLabel);
        codeLabel.setFont(new Font("Arial", Font.PLAIN, 26));
        Component verticalGlue = Box.createVerticalGlue();
        add(verticalGlue);
        genCode();
    }
    public static void genCode() {
        codeLabel.setText("Reference Code: "
                + (1000 + Utils.random.nextInt(8000)));
    }
    public static void showMe() {
        SideButtons.commands = TakeCardJPanel.commands;
        MainScreenCardJPanel
                .switchToCardStatic(MainScreenCardJPanel.STRING_TAKE_CARD);
    }
}
==> ./src/atm/gui/monitor/mainscreen/ByeJPanel.java <==
package atm.gui.monitor.mainscreen;
import javax.swing.JPanel;
import javax.swing.BoxLayout;
import javax.swing.JLabel;
import atm.gui.MyGUISettings;
import atm.utils.MyStrings;
import java.awt.Component;
import javax.swing.Box;
public class ByeJPanel extends JPanel {
    /***/
    private static final long serialVersionUID = 1L;
    public ByeJPanel() {
        setLayout(new BoxLayout(this, BoxLayout.X_AXIS));
        setBackground(MyGUISettings.getATMScreenBackGroundColor());
        Component horizontalGlue = Box.createHorizontalGlue();
        add(horizontalGlue);
        JPanel panel = new JPanel();
```

```
2624            add(panel);
2625            panel.setBackground(MyGUISettings.getATMScreenBackGroundColor());
2626            panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
2627            JLabel lblNewLabel1 = new JLabel(MyStrings.BYE1);
2628            lblNewLabel1.setAlignmentX(Component.CENTER_ALIGNMENT);
2629            panel.add(lblNewLabel1);
2630            lblNewLabel1.setFont(MyGUISettings.getFont(26));
2631            JLabel lblNewLabel2 = new JLabel(MyStrings.BYE2);
2632            lblNewLabel2.setAlignmentX(Component.CENTER_ALIGNMENT);
2633            panel.add(lblNewLabel2);
2634            lblNewLabel2.setFont(MyGUISettings.getFont(26));
2635            Component horizontalGlue_1 = Box.createHorizontalGlue();
2636            add(horizontalGlue_1);
2637        }
2638        public static void showMe() {
2639            MainScreenCardJPanel.switchToCardStatic(MainScreenCardJPanel.
                STRING_TAKE_CARD);
2640        }
2641    }
2642    ==> ./src/atm/gui/monitor/mainscreen/LoginJPanel.java <==
2643    package atm.gui.monitor.mainscreen;
2644    import javax.swing.JPanel;
2645    import javax.swing.JLabel;
2646    import javax.swing.JPasswordField;
2647    import java.awt.Component;
2648    import javax.swing.Box;
2649    import java.awt.BorderLayout;
2650    import java.util.Vector;
2651    import atm.core.ATM;
2652    import atm.gui.MyGUISettings;
2653    import atm.gui.keypad.KeypadJFrame;
2654    public class LoginJPanel extends JPanel {
2655        /***/
2656        private static final long serialVersionUID = 1L;
2657        private static Vector<LoginJPanel> contents = new Vector<LoginJPanel>();
2658        private JPasswordField passwordField;
2659        private JLabel lblWrongPassword;
2660        public LoginJPanel() {
2661            contents.add(this);
2662            setLayout(new BorderLayout(0, 0));
2663            Box horizontalBox = Box.createHorizontalBox();
2664            horizontalBox.setAlignmentY(Component.CENTER_ALIGNMENT);
2665            add(horizontalBox);
2666            Component horizontalGlue = Box.createHorizontalGlue();
2667            horizontalBox.add(horizontalGlue);
2668            Box verticalBox = Box.createVerticalBox();
2669            horizontalBox.add(verticalBox);
2670            lblWrongPassword = new JLabel("Wrong PIN");
2671            verticalBox.add(lblWrongPassword);
2672            JLabel lblInputPassword = new JLabel("Please input the password");
2673            verticalBox.add(lblInputPassword);
2674            lblInputPassword.setFont(MyGUISettings.getFont(26));
2675            Component horizontalGlue_1 = Box.createHorizontalGlue();
2676            horizontalBox.add(horizontalGlue_1);
2677            passwordField = new JPasswordField();
2678            add(passwordField, BorderLayout.SOUTH);
2679            passwordField.setFont(MyGUISettings.getFont(26));
2680            passwordField
2681                    .setBackground(MyGUISettings.getATMScreenBackGroundColor());
2682        }
```

```
2683        /** instance methods **/
2684        public void showMe() {
2685            lblWrongPassword.setVisible(false);
2686            passwordField.setText("");
2687            KeypadJFrame.switchTargetStatic(passwordField,
2688                    KeypadJFrame.STRING_MODE_PASSWORD);
2689            MainScreenCardJPanel
2690                    .switchToCardStatic(MainScreenCardJPanel.STRING_LOGIN);
2691        }
2692        public void showMeWrong(int wrongCount) {
2693            lblWrongPassword.setText("Wrong PIN (" + wrongCount + ")");
2694            lblWrongPassword.setVisible(true);
2695            passwordField.setText("");
2696            KeypadJFrame.switchTargetStatic(passwordField,
2697                    KeypadJFrame.STRING_MODE_PASSWORD);
2698            MainScreenCardJPanel
2699                    .switchToCardStatic(MainScreenCardJPanel.STRING_LOGIN);
2700        }
2701        /** static connectors to instance methods **/
2702        public static void showMeStatic() {
2703            ATM.initStatic();
2704            for (LoginJPanel loginJPanel : contents) {
2705                loginJPanel.showMe();
2706            }
2707        }
2708        public static void showMeWrongStatic(int wrongCount) {
2709            for (LoginJPanel loginJPanel : contents) {
2710                loginJPanel.showMeWrong(wrongCount);
2711            }
2712        }
2713    }
2714    ==> ./src/atm/gui/monitor/mainscreen/BannerJPanel.java <==
2715    package atm.gui.monitor.mainscreen;
2716    import javax.swing.JPanel;
2717    import javax.swing.JLabel;
2718    import atm.gui.MyGUISettings;
2719    import atm.utils.MyImages;
2720    import javax.swing.BoxLayout;
2721    import java.awt.Component;
2722    public class BannerJPanel extends JPanel {
2723        /***/
2724        private static final long serialVersionUID = 1L;
2725        public BannerJPanel() {
2726            setBackground(MyGUISettings.getATMScreenBackGroundColor());
2727            setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
2728            JLabel lblBanner = new JLabel(MyImages.banner);
2729            lblBanner.setAlignmentY(Component.TOP_ALIGNMENT);
2730            lblBanner.setAlignmentX(Component.CENTER_ALIGNMENT);
2731            add(lblBanner);
2732        }
2733    }
2734    ==> ./src/atm/gui/monitor/mainscreen/CashNotEnoughJPanel.java <==
2735    package atm.gui.monitor.mainscreen;
2736    import javax.security.auth.login.AccountNotFoundException;
2737    import javax.swing.JPanel;
2738    import javax.swing.BoxLayout;
2739    import javax.swing.JLabel;
2740    import bank.BankDatabase;
2741    import atm.core.ATM;
2742    import atm.gui.MyGUISettings;
```

```
2743  import atm.gui.monitor.MonitorJFrame;
2744  import atm.gui.monitor.sidebuttons.SideButtons;
2745  import atm.utils.MyImages;
2746  import atm.utils.MyStrings;
2747  import java.awt.Button;
2748  import java.awt.Component;
2749  import java.awt.Font;
2750  import java.util.Vector;
2751  import javax.swing.Box;
2752  import java.awt.GridLayout;
2753  public class CashNotEnoughJPanel extends JPanel {
2754      /***/
2755      private static final long serialVersionUID = 1L;
2756      private static Vector<CashNotEnoughJPanel> contents = new Vector<
          CashNotEnoughJPanel>();
2757      public static final String STRING_MAIN_MENU = "Main Menu";
2758      public static final String STRING_TAKE_CARD = "Take Card";
2759      public static final String[] commands = { "", "", "", "", STRING_MAIN_MENU,
2760              STRING_TAKE_CARD, "", "" };
2761      public CashNotEnoughJPanel() {
2762          contents.add(this);
2763          setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
2764          setBackground(MyGUISettings.getATMScreenBackGroundColor());
2765          JLabel label = new JLabel(MyImages.cashNotEnough);
2766          label.setAlignmentX(Component.CENTER_ALIGNMENT);
2767          add(label);
2768          JPanel panel = new JPanel();
2769          add(panel);
2770          panel.setBackground(MyGUISettings.getATMScreenBackGroundColor());
2771          panel.setLayout(new GridLayout(4, 2, 0, 0));
2772          for (String string : commands) {
2773              Button button = new Button(string);
2774              button.setFont(new Font("Arial", Font.PLAIN, 26));
2775              panel.add(button);
2776              button.setBackground(MyGUISettings.getATMScreenBackGroundColor());
2777          }
2778          Component verticalStrut = Box.createVerticalStrut(25);
2779          add(verticalStrut);
2780      }
2781      public void myUpdate() throws AccountNotFoundException {
2782          double overdrawnLimit = BankDatabase.getAccount(
2783                  ATM.getATM().getCurrentAccountNumber()).getOverdrawnLimit();
2784          Vector<JLabel> overDrawnMessageLabels = MyStrings
2785                  .getOverDrawnMessageLabels(overdrawnLimit);
2786          removeAll();
2787          Component verticalGlue = Box.createVerticalGlue();
2788          add(verticalGlue);
2789          for (JLabel jLabel : overDrawnMessageLabels) {
2790              add(jLabel);
2791              jLabel.setFont(MyGUISettings.getFont(26));
2792              jLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
2793              Component verticalGlue_1 = Box.createVerticalGlue();
2794              add(verticalGlue_1);
2795          }
2796      }
2797      public void showMe() {
2798          SideButtons.commands = WithDrawalJPanel.commands;
2799          MonitorJFrame.STATE = MainScreenCardJPanel.STRING_CASH_NOT_ENOUGH;
2800          MainScreenCardJPanel
2801                  .switchToCardStatic(MainScreenCardJPanel.STRING_CASH_NOT_ENOUGH);
```

```
2802        }
2803        public static void showMeStatic() {
2804            for (CashNotEnoughJPanel overdrawnExceptionJPanel : contents) {
2805                overdrawnExceptionJPanel.showMe();
2806            }
2807        }
2808    }
2809    ==> ./src/atm/gui/monitor/mainscreen/TakeCashJPanel.java <==
2810    package atm.gui.monitor.mainscreen;
2811    import javax.swing.JPanel;
2812    import javax.swing.JLabel;
2813    import java.awt.Font;
2814    import webs.layout.CenterLayout;
2815    import atm.gui.MyGUISettings;
2816    import atm.gui.monitor.sidebuttons.SideButtons;
2817    import atm.gui.virtualslots.cashdispenser.CashDispenserJPanel;
2818    public class TakeCashJPanel extends JPanel {
2819        /***/
2820        private static final long serialVersionUID = 1L;
2821        public static final String[] commands = { "", "", "", "", "", "", "", "" };
2822        /** Create the panel.*/
2823        public TakeCashJPanel() {
2824            // setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
2825            setLayout(new CenterLayout());
2826            setBackground(MyGUISettings.getATMScreenBackGroundColor());
2827            JLabel lblPleaseTakeYour = new JLabel("Please take your cash");
2828            add(lblPleaseTakeYour);
2829            // lblPleaseTakeYour.setAlignmentX(Component.CENTER_ALIGNMENT);
2830            lblPleaseTakeYour.setFont(new Font("Arial", Font.PLAIN, 26));
2831        }
2832        public static void showMe() {
2833            SideButtons.commands = TakeCashJPanel.commands;
2834            MainScreenCardJPanel
2835                    .switchToCardStatic(MainScreenCardJPanel.STRING_TAKE_CASH);
2836            CashDispenserJPanel.showMeStatic();
2837        }
2838    }
2839    ==> ./src/atm/gui/monitor/mainscreen/MaxWrongTryJPanel.java <==
2840    package atm.gui.monitor.mainscreen;
2841    import javax.swing.JPanel;
2842    import javax.swing.JLabel;
2843    import javax.swing.BoxLayout;
2844    import java.awt.Component;
2845    import java.awt.Font;
2846    import javax.swing.Box;
2847    import atm.gui.MyGUISettings;
2848    import atm.gui.virtualslots.cardslot.CardSlotCardJPanel;
2849    public class MaxWrongTryJPanel extends JPanel {
2850        /***/
2851        private static final long serialVersionUID = 1L;
2852        /** Create the panel.*/
2853        public MaxWrongTryJPanel() {
2854            setLayout(new BoxLayout(this, BoxLayout.X_AXIS));
2855            setBackground(MyGUISettings.getATMScreenBackGroundColor());
2856            Component horizontalGlue_1 = Box.createHorizontalGlue();
2857            add(horizontalGlue_1);
2858            Box verticalBox = Box.createVerticalBox();
2859            verticalBox.setAlignmentX(Component.CENTER_ALIGNMENT);
2860            add(verticalBox);
2861            JLabel label = new JLabel("Too many wrong try");
```

```
2862            label.setFont(new Font("Arial", Font.PLAIN, 26));
2863            label.setAlignmentX(0.5f);
2864            verticalBox.add(label);
2865            JLabel label_1 = new JLabel("Please contact CC Bank (9876-5432)");
2866            label_1.setFont(new Font("Arial", Font.PLAIN, 26));
2867            label_1.setAlignmentX(0.5f);
2868            verticalBox.add(label_1);
2869            Component horizontalGlue = Box.createHorizontalGlue();
2870            add(horizontalGlue);
2871        }
2872        public static void showMe() {
2873            MainScreenCardJPanel.switchToCardStatic(MainScreenCardJPanel.
                STRING_MAX_WRONG_TRY);
2874            CardSlotCardJPanel.waitPopCardStatic();
2875        }
2876    }
2877    ==> ./src/atm/gui/monitor/mainscreen/OverdrawnJPanel.java <==
2878    package atm.gui.monitor.mainscreen;
2879    import javax.security.auth.login.AccountNotFoundException;
2880    import javax.swing.JPanel;
2881    import javax.swing.BoxLayout;
2882    import javax.swing.JLabel;
2883    import bank.BankDatabase;
2884    import atm.core.ATM;
2885    import atm.gui.MyGUISettings;
2886    import atm.utils.MyStrings;
2887    import java.awt.Component;
2888    import java.util.Vector;
2889    import javax.swing.Box;
2890    public class OverdrawnJPanel extends JPanel {
2891        /***/
2892        private static final long serialVersionUID = 1L;
2893        private static Vector<OverdrawnJPanel> contents = new Vector<OverdrawnJPanel>();
2894        public OverdrawnJPanel() {
2895            contents.add(this);
2896            setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
2897            setBackground(MyGUISettings.getATMScreenBackGroundColor());
2898            Component verticalGlue = Box.createVerticalGlue();
2899            add(verticalGlue);
2900            JLabel label = new JLabel("Overdrawn: Loading...");
2901            label.setFont(MyGUISettings.getFont(26));
2902            label.setAlignmentX(Component.CENTER_ALIGNMENT);
2903            add(label);
2904            Component verticalGlue_1 = Box.createVerticalGlue();
2905            add(verticalGlue_1);
2906        }
2907        public void myUpdate() throws AccountNotFoundException {
2908            double overdrawnLimit = BankDatabase.getAccount(
2909                    ATM.getATM().getCurrentAccountNumber()).getOverdrawnLimit();
2910            Vector<JLabel> overDrawnMessageLabels = MyStrings
2911                    .getOverDrawnMessageLabels(overdrawnLimit);
2912            removeAll();
2913            Component verticalGlue = Box.createVerticalGlue();
2914            add(verticalGlue);
2915            for (JLabel jLabel : overDrawnMessageLabels) {
2916                add(jLabel);
2917                jLabel.setFont(MyGUISettings.getFont(26));
2918                jLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
2919                Component verticalGlue_1 = Box.createVerticalGlue();
2920                add(verticalGlue_1);
```

```
        }
    }
    public void showMe(String parent) {
        try {
            myUpdate();
            MainScreenCardJPanel
                    .switchToCardStatic(MainScreenCardJPanel.STRING_OVERDRAWN);
            switch (parent) {
            case MainScreenCardJPanel.STRING_WITHDRAWAL:
                WithDrawalJPanel.waitReturnFromWrongStatic();
                break;
            case MainScreenCardJPanel.STRING_TRANSFER:
                TransferJPanel.waitReturnFromWrongStatic();
                break;
            }
        } catch (AccountNotFoundException e) {
            CardNotValidJPanel.showMe();
        }
    }
    public static void showMeStatic(String parent) {
        for (OverdrawnJPanel overdrawnExceptionJPanel : contents) {
            overdrawnExceptionJPanel.showMe(parent);
        }
    }
}
==> ./src/atm/gui/monitor/mainscreen/CardNotValidJPanel.java <==
package atm.gui.monitor.mainscreen;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.BoxLayout;
import java.awt.Component;
import java.awt.Font;
import javax.swing.Box;
import atm.gui.MyGUISettings;
import atm.gui.virtualslots.cardslot.CardSlotCardJPanel;
public class CardNotValidJPanel extends JPanel {
    /***/
    private static final long serialVersionUID = 1L;
    /** Create the panel.*/
    public CardNotValidJPanel() {
        setLayout(new BoxLayout(this, BoxLayout.X_AXIS));
        setBackground(MyGUISettings.getATMScreenBackGroundColor());
        Component horizontalGlue_1 = Box.createHorizontalGlue();
        add(horizontalGlue_1);
        Box verticalBox = Box.createVerticalBox();
        verticalBox.setAlignmentX(Component.CENTER_ALIGNMENT);
        add(verticalBox);
        JLabel label = new JLabel("Your card is not identified");
        label.setFont(new Font("Arial", Font.PLAIN, 26));
        label.setAlignmentX(0.5f);
        verticalBox.add(label);
        JLabel label_1 = new JLabel("Please contact CC Bank (9876-5432)");
        label_1.setFont(new Font("Arial", Font.PLAIN, 26));
        label_1.setAlignmentX(0.5f);
        verticalBox.add(label_1);
        Component horizontalGlue = Box.createHorizontalGlue();
        add(horizontalGlue);
    }
    public static void showMe() {
        MainScreenCardJPanel
```

```
                .switchToCardStatic(MainScreenCardJPanel.STRING_CARD_NOT_VALID);
        CardSlotCardJPanel.waitPopCardStatic();
    }
}
==> ./src/atm/gui/monitor/mainscreen/CashRequiredNotSupportedJPanel.java <==
package atm.gui.monitor.mainscreen;
import java.awt.Component;
import java.util.Vector;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JLabel;
import javax.swing.JPanel;
import atm.gui.MyGUISettings;
public class CashRequiredNotSupportedJPanel extends JPanel {
    /***/
    private static final long serialVersionUID = 1L;
    private static Vector<CashRequiredNotSupportedJPanel> contents = new Vector<
    CashRequiredNotSupportedJPanel>();
    public CashRequiredNotSupportedJPanel() {
        contents.add(this);
        setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
        setBackground(MyGUISettings.getATMScreenBackGroundColor());
        Component verticalGlue = Box.createVerticalGlue();
        add(verticalGlue);
        JLabel label = new JLabel(
                "Cash Notes required is not supported by this ATM");
        label.setFont(MyGUISettings.getFont(26));
        label.setAlignmentX(Component.CENTER_ALIGNMENT);
        add(label);
        Component verticalGlue_1 = Box.createVerticalGlue();
        add(verticalGlue_1);
    }
    public static void showMe() {
        MainScreenCardJPanel
                .switchToCardStatic(MainScreenCardJPanel.
                STRING_CASH_REQUIRED_NOT_SUPPORTED);
        WithDrawalJPanel.waitReturnFromWrongStatic();
    }
}
==> ./src/atm/gui/monitor/mainscreen/TransferJPanel.java <==
package atm.gui.monitor.mainscreen;
import javax.security.auth.login.AccountNotFoundException;
import javax.swing.JPanel;
import java.util.Vector;
import javax.swing.BoxLayout;
import java.awt.Component;
import atm.core.ATM;
import atm.exception.OverdrawnException;
import atm.exception.TransferSameAccountException;
import atm.gui.MyGUISettings;
import atm.gui.keypad.KeypadJFrame;
import atm.gui.monitor.MonitorJFrame;
import atm.gui.monitor.sidebuttons.SideButtons;
import atm.utils.MyInputHandler;
import atm.utils.MyStaticStuff;
import bank.operation.Transfer;
import javax.swing.JLabel;
import javax.swing.JTextField;
public class TransferJPanel extends JPanel {
    /***/
```

```
3039        private static final long serialVersionUID = 1L;
3040        private static Vector<TransferJPanel> contents = new Vector<TransferJPanel>();
3041        public static final String STRING_VIEW_BALANCE = "View Balance";
3042        public static final String STRING_WITHDRAW_CASH = "Withdraw Cash";
3043        public static final String STRING_TRANSFER_FUNDS = "Transfer Funds";
3044        public static final String STRING_TAKE_CARD = "Take Card";
3045        public static final String[] commands = { "", "", STRING_VIEW_BALANCE,
3046                STRING_WITHDRAW_CASH, STRING_TRANSFER_FUNDS, STRING_TAKE_CARD, "",
3047                "" };
3048        private int wrongTry;
3049        private JTextField receiverAccountNumberTextField;
3050        private JTextField amountTextField;
3051        public TransferJPanel() {
3052            contents.add(this);
3053            setBackground(MyGUISettings.getATMScreenBackGroundColor());
3054            setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
3055            JLabel lblExtraCharge = new JLabel(MyStaticStuff.getExtraChargeString());
3056            lblExtraCharge.setAlignmentX(Component.CENTER_ALIGNMENT);
3057            add(lblExtraCharge);
3058            lblExtraCharge.setFont(MyGUISettings.getFont(18));
3059            JLabel lblNewLabel = new JLabel("Receiver Account Number:");
3060            lblNewLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
3061            add(lblNewLabel);
3062            lblNewLabel.setFont(MyGUISettings.getFont(26));
3063            receiverAccountNumberTextField = new JTextField();
3064            add(receiverAccountNumberTextField);
3065            receiverAccountNumberTextField.setColumns(10);
3066            receiverAccountNumberTextField.setBackground(MyGUISettings
3067                    .getATMScreenBackGroundColor());
3068            JLabel lblNewLabel_1 = new JLabel("Amount to transfer:");
3069            lblNewLabel_1.setAlignmentX(Component.CENTER_ALIGNMENT);
3070            add(lblNewLabel_1);
3071            lblNewLabel_1.setFont(MyGUISettings.getFont(26));
3072            amountTextField = new JTextField();
3073            add(amountTextField);
3074            amountTextField.setColumns(10);
3075            amountTextField.setBackground(MyGUISettings
3076                    .getATMScreenBackGroundColor());
3077            wrongTry = 0;
3078        }
3079        /** instance methods **/
3080        public void enterKeyPressed() {
3081            System.out.println("enter");
3082            switch (KeypadJFrame.getModeStatic()) {
3083            case KeypadJFrame.STRING_MODE_ACCOUNTNUMBER:
3084                System.out.println("next");
3085                KeypadJFrame.switchTargetStatic(amountTextField,
3086                        KeypadJFrame.STRING_MODE_AMOUNT);
3087                break;
3088            case KeypadJFrame.STRING_MODE_AMOUNT:
3089                System.out.println("try");
3090                tryTransfer();
3091                break;
3092            }
3093        }
3094        private void tryTransfer() {
3095            boolean transferSuccess = false;
3096            try {
3097                System.out.println("check Transfer");
3098                double amount = Double.parseDouble(amountTextField.getText());
```

```
3099                Transfer.transferGUI(ATM.getATM(),
3100                        receiverAccountNumberTextField.getText(), amount);
3101                transferSuccess = true;
3102            } catch (NumberFormatException e) {
3103                System.out.println("not double?");
3104                MainMenuJPanel.showMe();
3105            } catch (AccountNotFoundException e) {
3106                System.out.println("account not found");
3107                TransferReceiverAccountNotFoundJPanel.showMeStatic();
3108            } catch (TransferSameAccountException e) {
3109                System.out.println("tranfer same account");
3110                TransferSameAccountJPanel.showMeStatic();
3111            } catch (OverdrawnException e) {
3112                System.out.println("overdrawn");
3113                OverdrawnJPanel.showMeStatic(MainScreenCardJPanel.STRING_TRANSFER);
3114            }
3115            if (transferSuccess) {
3116                // transfer success
3117                System.out.println("transfer success");
3118                TransferSuccessJPanel.showMeStatic();
3119            }
3120        }
3121        public void showMe() {
3122            System.out.println("show transfer jpanel");
3123            MonitorJFrame.STATE = MainScreenCardJPanel.STRING_TRANSFER;
3124            SideButtons.commands = TransferJPanel.commands;
3125            MainScreenCardJPanel
3126                    .switchToCardStatic(MainScreenCardJPanel.STRING_TRANSFER);
3127            receiverAccountNumberTextField.setText("");
3128            amountTextField.setText("");
3129            KeypadJFrame.switchTargetStatic(receiverAccountNumberTextField,
3130                    KeypadJFrame.STRING_MODE_ACCOUNTNUMBER);
3131        }
3132        public void showMeWrong() {
3133            int oldWrongTry = wrongTry;
3134            showMe();
3135            wrongTry = oldWrongTry + 1;
3136            if (wrongTry > MyInputHandler.MAX_WRONG_INPUT)
3137                MaxWrongTryJPanel.showMe();
3138        }
3139        /** static connector to instance stuff **/
3140        public static void showMeStatic() {
3141            for (TransferJPanel content : contents) {
3142                content.showMe();
3143            }
3144        }
3145        public static void showMeWrongStatic() {
3146            for (TransferJPanel content : contents) {
3147                content.showMeWrong();
3148            }
3149        }
3150        public static void enterKeyPressedStatic() {
3151            for (TransferJPanel transferJPanel : contents) {
3152                transferJPanel.enterKeyPressed();
3153            }
3154        }
3155        /** static methods **/
3156        public static void waitReturnFromWrongStatic() {
3157            WaitReturnFromWrongThread returnFromWrongThread = new
                WaitReturnFromWrongThread();
```

```
3158            returnFromWrongThread.start();
3159        }
3160        /** private class **/
3161        private static class WaitReturnFromWrongThread extends Thread {
3162            @Override
3163            public void run() {
3164                try {
3165                    Thread.sleep(2000);
3166                } catch (InterruptedException e) {
3167                }
3168                showMeWrongStatic();
3169            }
3170        }
3171    }
3172    ==> ./src/atm/gui/monitor/mainscreen/TransferSuccessJPanel.java <==
3173    package atm.gui.monitor.mainscreen;
3174    import java.awt.Component;
3175    import java.util.Vector;
3176    import javax.swing.Box;
3177    import javax.swing.BoxLayout;
3178    import javax.swing.JLabel;
3179    import javax.swing.JPanel;
3180    import atm.gui.MyGUISettings;
3181    public class TransferSuccessJPanel extends JPanel {
3182        /***/
3183        private static final long serialVersionUID = 1L;
3184        private static Vector<TransferSuccessJPanel> contents = new Vector<
          TransferSuccessJPanel>();
3185        public TransferSuccessJPanel() {
3186            contents.add(this);
3187            setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
3188            setBackground(MyGUISettings.getATMScreenBackGroundColor());
3189            Component verticalGlue = Box.createVerticalGlue();
3190            add(verticalGlue);
3191            JLabel label = new JLabel("Transfer Success");
3192            label.setFont(MyGUISettings.getFont(26));
3193            label.setAlignmentX(Component.CENTER_ALIGNMENT);
3194            add(label);
3195            Component verticalGlue_1 = Box.createVerticalGlue();
3196            add(verticalGlue_1);
3197        }
3198        public void showMe() {
3199            MainScreenCardJPanel
3200                    .switchToCardStatic(MainScreenCardJPanel.STRING_TRANSFER_SUCCESS);
3201            MainMenuJPanel.waitShowMeStatic();
3202        }
3203        public static void showMeStatic() {
3204            for (TransferSuccessJPanel content : contents) {
3205                content.showMe();
3206            }
3207        }
3208    }
3209    ==> ./src/atm/gui/monitor/mainscreen/MainMenuJPanel.java <==
3210    package atm.gui.monitor.mainscreen;
3211    import javax.swing.JPanel;
3212    import java.awt.GridLayout;
3213    import java.util.Vector;
3214    import java.awt.Button;
3215    import javax.swing.BoxLayout;
3216    import java.awt.Component;
```

```java
import javax.swing.Box;
import atm.core.ATM;
import atm.gui.MyGUISettings;
import atm.gui.keypad.KeypadJFrame;
import atm.gui.monitor.MonitorJFrame;
import atm.gui.monitor.sidebuttons.SideButtons;
import java.awt.Font;
public class MainMenuJPanel extends JPanel {
    /***/
    private static final long serialVersionUID = 1L;
    private static Vector<MainMenuJPanel> contents = new Vector<MainMenuJPanel>();
    public static final String STRING_VIEW_BALANCE = "View Balance";
    public static final String STRING_WITHDRAW_CASH = "Withdraw Cash";
    public static final String STRING_TRANSFER_FUNDS = "Transfer Funds";
    public static final String STRING_TAKE_CARD = "Take Card";
    public static final String[] commands = { "", "", STRING_VIEW_BALANCE,
            STRING_WITHDRAW_CASH, STRING_TRANSFER_FUNDS, STRING_TAKE_CARD, "",
            "" };
    public MainMenuJPanel() {
        contents.add(this);
        setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
        setBackground(MyGUISettings.getATMScreenBackGroundColor());
        Component verticalStrut_1 = Box.createVerticalStrut(75);
        add(verticalStrut_1);
        JPanel panel = new JPanel();
        add(panel);
        panel.setLayout(new GridLayout(4, 2, 0, 0));
        panel.setBackground(MyGUISettings.getATMScreenBackGroundColor());
        for (String string : commands) {
            Button button = new Button(string);
            button.setFont(new Font("Arial", Font.PLAIN, 26));
            panel.add(button);
            button.setBackground(MyGUISettings.getATMScreenBackGroundColor());
        }
        Component verticalStrut = Box.createVerticalStrut(25);
        add(verticalStrut);
    }
    public static void showMe() {
        ATM.getATM().init();
        MonitorJFrame.STATE = MainScreenCardJPanel.STRING_MAIN_MENU;
        SideButtons.commands = MainMenuJPanel.commands;
        KeypadJFrame.switchModeStatic(KeypadJFrame.STRING_MODE_NULL);
        MainScreenCardJPanel
                .switchToCardStatic(MainScreenCardJPanel.STRING_MAIN_MENU);
    }
    public static void waitShowMeStatic() {
        WaitReturnThread waitReturnThread = new WaitReturnThread();
        waitReturnThread.start();
    }
    /** private class **/
    private static class WaitReturnThread extends Thread {
        @Override
        public void run() {
            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
            }
            showMe();
        }
    }
```

```
}
==> ./src/atm/gui/monitor/mainscreen/TransferReceiverAccountNotFoundJPanel.java <==
package atm.gui.monitor.mainscreen;
import java.awt.Component;
import java.util.Vector;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JLabel;
import javax.swing.JPanel;
import atm.gui.MyGUISettings;
public class TransferReceiverAccountNotFoundJPanel extends JPanel {
    /***/
    private static final long serialVersionUID = 1L;
    private static Vector<TransferReceiverAccountNotFoundJPanel> contents = new
    Vector<TransferReceiverAccountNotFoundJPanel>();
    public TransferReceiverAccountNotFoundJPanel() {
        contents.add(this);
        setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
        setBackground(MyGUISettings.getATMScreenBackGroundColor());
        Component verticalGlue = Box.createVerticalGlue();
        add(verticalGlue);
        JLabel label = new JLabel("Receiver Account Not Found");
        label.setFont(MyGUISettings.getFont(26));
        label.setAlignmentX(Component.CENTER_ALIGNMENT);
        add(label);
        Component verticalGlue_1 = Box.createVerticalGlue();
        add(verticalGlue_1);
    }
    public void showMe() {
        MainScreenCardJPanel
                .switchToCardStatic(MainScreenCardJPanel.
                STRING_TRANSFER_RECEIVER_ACCOUNT_NOT_FOUND);
        TransferJPanel.waitReturnFromWrongStatic();
    }
    public static void showMeStatic() {
        for (TransferReceiverAccountNotFoundJPanel content : contents) {
            content.showMe();
        }
    }
}
==> ./src/atm/gui/monitor/mainscreen/ShowPopCashNotesJPanel.java <==
package atm.gui.monitor.mainscreen;
import java.util.Vector;
import javax.swing.JPanel;
import javax.swing.JLabel;
import atm.gui.MyGUISettings;
import atm.gui.virtualslots.cashdispenser.notes.CashNote100;
import atm.gui.virtualslots.cashdispenser.notes.CashNote1000;
import atm.gui.virtualslots.cashdispenser.notes.CashNote500;
import atm.utils.CashCount;
import webs.layout.CircleLayout;
public class ShowPopCashNotesJPanel extends JPanel {
    /***/
    private static final long serialVersionUID = 1L;
    private static Vector<ShowPopCashNotesJPanel> contents = new Vector<
    ShowPopCashNotesJPanel>();
    public ShowPopCashNotesJPanel() {
        contents.add(this);
        setBackground(MyGUISettings.getATMScreenBackGroundColor());
        setLayout(new CircleLayout());
```

```
    }
    public void myUpdate(Vector<CashCount> popCashCounts) {
        // reset layout (shown-part GUI)
        removeAll();
        for (CashCount cashCount : popCashCounts) {
            if (cashCount.getCount() > 0) {
                switch (cashCount.getValue()) {
                case 100:
                    for (int i = 0; i < cashCount.getCount(); i++)
                        add(new JLabel(CashNote100.imageIcon));
                    break;
                case 500:
                    for (int i = 0; i < cashCount.getCount(); i++)
                        add(new JLabel(CashNote500.imageIcon));
                    break;
                case 1000:
                    for (int i = 0; i < cashCount.getCount(); i++)
                        add(new JLabel(CashNote1000.imageIcon));
                    break;
                }
            }
        }
    }
    public static void myUpdateStatic(Vector<CashCount> popCashCounts) {
        for (ShowPopCashNotesJPanel content : contents) {
            content.myUpdate(popCashCounts);
        }
    }
}
==> ./src/atm/gui/monitor/mainscreen/WithDrawalJPanel.java <==
package atm.gui.monitor.mainscreen;
import javax.security.auth.login.AccountNotFoundException;
import javax.swing.JPanel;
import java.awt.GridLayout;
import java.util.Vector;
import java.awt.Button;
import javax.swing.BoxLayout;
import java.awt.Component;
import javax.swing.Box;
import atm.core.ATM;
import atm.core.CashDispenser;
import atm.exception.CashNotEnoughException;
import atm.exception.CashNotesNotSupportedException;
import atm.exception.CashOutException;
import atm.exception.OverdrawnException;
import atm.gui.MyGUISettings;
import atm.gui.keypad.KeypadJFrame;
import atm.gui.monitor.MonitorJFrame;
import atm.gui.monitor.sidebuttons.SideButtons;
import atm.gui.virtualslots.cardslot.CardSlotCardJPanel;
import atm.gui.virtualslots.cashdispenser.CashDispenserJPanel;
import atm.utils.MyImages;
import atm.utils.MyInputHandler;
import atm.utils.MyStaticStuff;
import atm.utils.MyStrings;
import bank.account.Account;
import bank.operation.Withdrawal;
import java.awt.Font;
import javax.swing.JTextField;
import java.awt.Color;
```

```
3394  import java.awt.Dimension;
3395  import java.awt.BorderLayout;
3396  import javax.swing.JLabel;
3397  public class WithDrawalJPanel extends JPanel {
3398      /***/
3399      private static final long serialVersionUID = 1L;
3400      private static Vector<WithDrawalJPanel> contents = new Vector<WithDrawalJPanel
      >();
3401      public static final String STRING_MAIN_MENU = "Main Menu";
3402      public static final String STRING_TAKE_CARD = "Take Card";
3403      public static final String[] commands = {
3404              String.valueOf(MyStaticStuff.MenuCashValue[0]),
3405              String.valueOf(MyStaticStuff.MenuCashValue[1]),
3406              String.valueOf(MyStaticStuff.MenuCashValue[2]),
3407              String.valueOf(MyStaticStuff.MenuCashValue[3]), STRING_MAIN_MENU,
3408              STRING_TAKE_CARD, "", "" };
3409      private JTextField textField;
3410      private Withdrawal withdrawalOperation;
3411      private int wrongTry;
3412      private JPanel topPanel;
3413      private Component verticalStrut;
3414      private JLabel extraChargeLabel;
3415      /** constructor **/
3416      public WithDrawalJPanel() {
3417          contents.add(this);
3418          setBackground(MyGUISettings.getATMScreenBackGroundColor());
3419          setLayout(new BorderLayout(0, 0));
3420          JPanel strucPanel = new JPanel();
3421          add(strucPanel);
3422          strucPanel.setBackground(MyGUISettings.getATMScreenBackGroundColor());
3423          strucPanel.setLayout(new BoxLayout(strucPanel, BoxLayout.Y_AXIS));
3424          topPanel = new JPanel();
3425          strucPanel.add(topPanel);
3426          topPanel.setBackground(MyGUISettings.getATMScreenBackGroundColor());
3427          topPanel.setLayout(new BoxLayout(topPanel, BoxLayout.X_AXIS));
3428          verticalStrut = Box.createVerticalStrut(75);
3429          topPanel.add(verticalStrut);
3430          extraChargeLabel = new JLabel(MyImages.extraCharge);
3431          topPanel.add(extraChargeLabel);
3432          JPanel contentPanel = new JPanel();
3433          strucPanel.add(contentPanel);
3434          contentPanel.setLayout(new GridLayout(4, 2, 0, 0));
3435          contentPanel.setBackground(MyGUISettings.getATMScreenBackGroundColor());
3436          Box horizontalBox = Box.createHorizontalBox();
3437          add(horizontalBox, BorderLayout.SOUTH);
3438          JLabel lblNewLabel = new JLabel("HKD $ ");
3439          horizontalBox.add(lblNewLabel);
3440          textField = new JTextField("");
3441          horizontalBox.add(textField);
3442          textField.setPreferredSize(new Dimension(400, 25));
3443          textField.setColumns(10);
3444          textField.setBackground(new Color(135, 206, 250));
3445          for (int i = 0; i < 4; i++) {
3446              Button button = new Button(MyStrings.DOLLAR_SIGN + " "
3447                      + commands[i]);
3448              button.setFont(new Font("Arial", Font.PLAIN, 26));
3449              contentPanel.add(button);
3450              button.setBackground(MyGUISettings.getATMScreenBackGroundColor());
3451          }
3452          for (int i = 4; i < 8; i++) {
```

```
3453                Button button = new Button(commands[i]);
3454                button.setFont(new Font("Arial", Font.PLAIN, 26));
3455                contentPanel.add(button);
3456                button.setBackground(MyGUISettings.getATMScreenBackGroundColor());
3457            }
3458        }
3459        /** instance methods **/
3460        public void sideButtonClick(String command) {
3461            textField.setText(command);
3462            tryWithDrawal();
3463        }
3464        public void enterButtonClick() {
3465            tryWithDrawal();
3466        }
3467        public void showMe() {
3468            if (Account.isMyBankAccount(ATM.getATM().getCurrentAccountNumber())) {
3469                topPanel.removeAll();
3470                topPanel.add(verticalStrut);
3471                verticalStrut.setVisible(true);
3472                extraChargeLabel.setVisible(false);
3473            } else {
3474                topPanel.removeAll();
3475                topPanel.add(extraChargeLabel);
3476                verticalStrut.setVisible(false);
3477                extraChargeLabel.setVisible(true);
3478            }
3479            wrongTry = 0;
3480            textField.setText("");
3481            withdrawalOperation = new Withdrawal(ATM.getATM());
3482            ATM.getATM().init();
3483            MonitorJFrame.STATE = MainScreenCardJPanel.STRING_WITHDRAWAL;
3484            SideButtons.commands = WithDrawalJPanel.commands;
3485            KeypadJFrame.switchTargetStatic(textField,
3486                    KeypadJFrame.STRING_MODE_CASH_AMOUNT);
3487            MainScreenCardJPanel
3488                    .switchToCardStatic(MainScreenCardJPanel.STRING_WITHDRAWAL);
3489        }
3490        public void showMeWrong() {
3491            int oldWrongTry = wrongTry;
3492            CashDispenser.rollback();
3493            showMe();
3494            wrongTry = oldWrongTry + 1;
3495            if (wrongTry > MyInputHandler.MAX_WRONG_INPUT)
3496                MaxWrongTryJPanel.showMe();
3497        }
3498        public void tryWithDrawal() {
3499            try {
3500                withdrawalOperation.setAmount(textField.getText());
3501                withdrawalOperation.executeGUI();
3502                // if success it will throw cash out exception
3503                // withdrawal failed
3504                CashDispenser.rollback();
3505            } catch (NumberFormatException e) {
3506                CashDispenser.rollback();
3507                System.out.println("Error! cash amount is not int?");
3508                wrongTry++;
3509                MainMenuJPanel.showMe();
3510            } catch (AccountNotFoundException e) {
3511                CashDispenser.rollback();
3512                CardNotValidJPanel.showMe();
```

```
3513            } catch (OverdrawnException e) {
3514                CashDispenser.rollback();
3515                OverdrawnJPanel
3516                        .showMeStatic(MainScreenCardJPanel.STRING_WITHDRAWAL);
3517            } catch (CashNotEnoughException e) {
3518                CashDispenser.rollback();
3519                CashNotEnoughJPanel.showMeStatic();
3520            } catch (CashOutException e) {
3521                CashDispenserJPanel.setPopCashCountsStatic(e.getCashCounts());
3522                CardSlotCardJPanel.popCardStatic();
3523            } catch (CashNotesNotSupportedException e) {
3524                CashRequiredNotSupportedJPanel.showMe();
3525            }
3526        }
3527        /** static methods **/
3528        public static void waitReturnFromWrongStatic() {
3529            WaitReturnFromWrongThread returnFromWrongThread = new
                WaitReturnFromWrongThread();
3530            returnFromWrongThread.start();
3531        }
3532        /** static connector to instance stuff **/
3533        public static void sideButtonClickStatic(String command) {
3534            for (WithDrawalJPanel withDrawalJPanel : contents) {
3535                withDrawalJPanel.sideButtonClick(command);
3536            }
3537        }
3538        public static void showMeStatic() {
3539            for (WithDrawalJPanel withDrawalJPanel : contents) {
3540                withDrawalJPanel.showMe();
3541            }
3542        }
3543        public static void showMeWrongStatic() {
3544            for (WithDrawalJPanel withDrawalJPanel : contents) {
3545                withDrawalJPanel.showMeWrong();
3546            }
3547        }
3548        public static void enterButtonClickStatic() {
3549            for (WithDrawalJPanel withDrawalJPanel : contents) {
3550                withDrawalJPanel.enterButtonClick();
3551            }
3552        }
3553        /** private class **/
3554        private static class WaitReturnFromWrongThread extends Thread {
3555            @Override
3556            public void run() {
3557                try {
3558                    Thread.sleep(2000);
3559                } catch (InterruptedException e) {
3560                }
3561                showMeWrongStatic();
3562            }
3563        }
3564    }
3565    ==> ./src/atm/gui/monitor/mainscreen/TransferSameAccountJPanel.java <==
3566    package atm.gui.monitor.mainscreen;
3567    import java.awt.Component;
3568    import java.util.Vector;
3569    import javax.swing.Box;
3570    import javax.swing.BoxLayout;
3571    import javax.swing.JLabel;
```

```
3572 import javax.swing.JPanel;
3573 import atm.gui.MyGUISettings;
3574 import atm.utils.MyStrings;
3575 public class TransferSameAccountJPanel extends JPanel {
3576     /***/
3577     private static final long serialVersionUID = 1L;
3578     private static Vector<TransferSameAccountJPanel> contents = new Vector<
      TransferSameAccountJPanel>();
3579     public TransferSameAccountJPanel() {
3580         contents.add(this);
3581         setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
3582         setBackground(MyGUISettings.getATMScreenBackGroundColor());
3583         Component verticalGlue = Box.createVerticalGlue();
3584         add(verticalGlue);
3585         JLabel label = new JLabel(MyStrings.TRANSFER_SAME_ACCOUNT);
3586         label.setFont(MyGUISettings.getFont(26));
3587         label.setAlignmentX(Component.CENTER_ALIGNMENT);
3588         add(label);
3589         Component verticalGlue_1 = Box.createVerticalGlue();
3590         add(verticalGlue_1);
3591     }
3592     public void showMe() {
3593         MainScreenCardJPanel
3594                 .switchToCardStatic(MainScreenCardJPanel.
                    STRING_TRANSFER_SAME_ACCOUNT);
3595         TransferJPanel.waitReturnFromWrongStatic();
3596     }
3597     public static void showMeStatic() {
3598         for (TransferSameAccountJPanel content : contents) {
3599             content.showMe();
3600         }
3601     }
3602 }
3603 ==> ./src/atm/gui/monitor/mainscreen/GUIPrinter.java <==
3604 package atm.gui.monitor.mainscreen;
3605 import java.io.IOException;
3606 import java.io.OutputStream;
3607 import java.io.PrintStream;
3608 import java.util.Vector;
3609 import javax.swing.JTextArea;
3610 import javax.swing.SwingUtilities;
3611 public class GUIPrinter extends OutputStream {
3612     private static Vector<GUIPrinter> contents = new Vector<GUIPrinter>();
3613     public JTextArea textArea;
3614     private static PrintStream systemPrintStream = System.out;
3615     public GUIPrinter(JTextArea textArea) {
3616         contents.add(this);
3617         this.textArea = textArea;
3618     }
3619     @Override
3620     public void write(byte[] bytes, int offset, int length) throws IOException {
3621         final String text = new String(bytes, offset, length);
3622         SwingUtilities.invokeLater(new Runnable() {
3623             @Override
3624             public void run() {
3625                 textArea.append(text);
3626             }
3627         });
3628     }
3629     @Override
```

```
3630        public void write(int b) throws IOException {
3631            write(new byte[] { (byte) b }, 0, 1);
3632        }
3633        public void start() {
3634            for (GUIPrinter guiPrinter : contents) {
3635                System.setOut(new PrintStream(guiPrinter));
3636            }
3637        }
3638        public void stop() {
3639            System.setOut(systemPrintStream);
3640        }
3641    }
3642    ==> ./src/atm/gui/monitor/mainscreen/WelcomeJPanel.java <==
3643    package atm.gui.monitor.mainscreen;
3644    import java.util.Vector;
3645    import javax.swing.JPanel;
3646    import javax.swing.BoxLayout;
3647    import atm.core.ATM;
3648    import atm.gui.MyGUISettings;
3649    import java.awt.Component;
3650    public class WelcomeJPanel extends JPanel {
3651        /***/
3652        private static final long serialVersionUID = 1L;
3653        private static Vector<WelcomeJPanel> contents = new Vector<WelcomeJPanel>();
3654        private BannerJPanel bannerJPanel;
3655        private AvailableCashNotesJPanel availableCashNotesJPanel;
3656        public WelcomeJPanel() {
3657            contents.add(this);
3658            setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
3659            setBackground(MyGUISettings.getATMScreenBackGroundColor());
3660            bannerJPanel = new BannerJPanel();
3661            bannerJPanel.setAlignmentX(Component.CENTER_ALIGNMENT);
3662            add(bannerJPanel);
3663            availableCashNotesJPanel = new AvailableCashNotesJPanel();
3664            add(availableCashNotesJPanel);
3665        }
3666        public void showMe() {
3667            System.out.println("WelcomeJPanel.showMe()");
3668            availableCashNotesJPanel.myUpdate();
3669            ATM.initStatic();
3670            MainScreenCardJPanel
3671                    .switchToCardStatic(MainScreenCardJPanel.STRING_WELCOME);
3672        }
3673        public static void showMeStatic() {
3674            System.out.println("WelcomeJPanel.showMeStatic()");
3675            for (WelcomeJPanel welcomeJPanel : contents) {
3676                welcomeJPanel.showMe();
3677            }
3678        }
3679    }
3680    ==> ./src/atm/gui/virtualslots/cashdispenser/CashDispenserJPanel.java <==
3681    package atm.gui.virtualslots.cashdispenser;
3682    import java.awt.BorderLayout;
3683    import java.awt.event.ActionEvent;
3684    import java.awt.event.ActionListener;
3685    import java.util.Vector;
3686    import javax.swing.JButton;
3687    import javax.swing.JLabel;
3688    import javax.swing.JPanel;
3689    import webs.layout.WrapLayout;
```

```
3690  import atm.core.CashDispenser;
3691  import atm.gui.virtualslots.VirtualSlotsJFrame;
3692  import atm.gui.virtualslots.cashdispenser.notes.CashNote100;
3693  import atm.gui.virtualslots.cashdispenser.notes.CashNote1000;
3694  import atm.gui.virtualslots.cashdispenser.notes.CashNote500;
3695  import atm.utils.CashCount;
3696  import atm.utils.MyStrings;
3697  import javax.swing.BoxLayout;
3698  import java.awt.Component;
3699  public class CashDispenserJPanel extends JPanel {
3700      /***/
3701      private static final long serialVersionUID = 1L;
3702      private static Vector<CashDispenserJPanel> contents = new Vector<
      CashDispenserJPanel>();
3703      private JButton takeCashJButton;
3704      private JPanel cashPanel;
3705      public static boolean hasCashToBePopped = false;
3706      private Vector<CashCount> popCashCounts;
3707      public CashDispenserJPanel() {
3708          contents.add(this);
3709          setLayout(new BorderLayout(0, 0));
3710          JPanel northPanel = new JPanel();
3711          add(northPanel, BorderLayout.NORTH);
3712          northPanel.setLayout(new BoxLayout(northPanel, BoxLayout.Y_AXIS));
3713          takeCashJButton = new JButton("Take all cash");
3714          takeCashJButton.setAlignmentX(Component.CENTER_ALIGNMENT);
3715          northPanel.add(takeCashJButton);
3716          takeCashJButton.setVisible(false);
3717          takeCashJButton.addActionListener(getButtonActionListener());
3718          //cashPanel = new JPanel(new CircleLayout());
3719          cashPanel = new JPanel(new WrapLayout());
3720          add(cashPanel, BorderLayout.CENTER);
3721          cashPanel.setAlignmentX(Component.CENTER_ALIGNMENT);
3722          cashPanel.setVisible(false);
3723      }
3724      private ActionListener getButtonActionListener() {
3725          return new ActionListener() {
3726              @Override
3727              public void actionPerformed(ActionEvent e) {
3728                  System.out.println("cash taken by user");
3729                  hasCashToBePopped = false;
3730                  CashDispenser.commit();
3731                  takeCashJButton.setVisible(false);
3732                  cashPanel.setVisible(false);
3733                  VirtualSlotsJFrame.myResetStatic();
3734              }
3735          };
3736      }
3737      public void setPopCashCounts(Vector<CashCount> popCashCounts) {
3738          hasCashToBePopped = true;
3739          this.popCashCounts = popCashCounts;
3740      }
3741      public void showMe() {
3742          takeCashJButton.setVisible(true);
3743          cashPanel.setVisible(true);
3744          // reset contentpanel layout (shown-part GUI)
3745          cashPanel.removeAll();
3746          for (CashCount cashCount : popCashCounts) {
3747              if (cashCount.getCount() > 0) {
3748                  switch (cashCount.getValue()) {
```

```
3749                    case 100:
3750                        for (int i = 0; i < cashCount.getCount(); i++) {
3751                            cashPanel.add(new JLabel(CashNote100.imageIcon));
3752                            System.out.println("popping " + MyStrings.DOLLAR_SIGN
3753                                    + " " + cashCount.getValue());
3754                        }
3755                        break;
3756                    case 500:
3757                        for (int i = 0; i < cashCount.getCount(); i++) {
3758                            cashPanel.add(new JLabel(CashNote500.imageIcon));
3759                            System.out.println("popping " + MyStrings.DOLLAR_SIGN
3760                                    + " " + cashCount.getValue());
3761                        }
3762                        break;
3763                    case 1000:
3764                        for (int i = 0; i < cashCount.getCount(); i++) {
3765                            cashPanel.add(new JLabel(CashNote1000.imageIcon));
3766                            System.out.println("popping " + MyStrings.DOLLAR_SIGN
3767                                    + " " + cashCount.getValue());
3768                        }
3769                        break;
3770                    }
3771                }
3772            }
3773        }
3774        /** static connector to instance stuff **/
3775        public static void setPopCashCountsStatic(Vector<CashCount> popCashCounts) {
3776            for (CashDispenserJPanel content : contents) {
3777                content.setPopCashCounts(popCashCounts);
3778            }
3779        }
3780        public static void showMeStatic() {
3781            for (CashDispenserJPanel content : contents) {
3782                content.showMe();
3783            }
3784        }
3785    }
3786    ==> ./src/atm/gui/virtualslots/cashdispenser/notes/CashNote500.java <==
3787    package atm.gui.virtualslots.cashdispenser.notes;
3788    import java.io.IOException;
3789    import java.net.MalformedURLException;
3790    import java.net.URL;
3791    import javax.swing.ImageIcon;
3792    import javax.swing.JLabel;
3793    import atm.utils.MyURLs;
3794    public class CashNote500 implements CashNote {
3795        public static ImageIcon imageIcon;
3796        public static JLabel jLabel;
3797        public static int value;
3798        public static void init() throws MalformedURLException {
3799            imageIcon = new ImageIcon(new URL(MyURLs.IMAGE_NOTE500));
3800            jLabel = new JLabel(imageIcon);
3801            value = 500;
3802        }
3803        @Override
3804        public void fetchImage() throws IOException {
3805            System.out.println("fetching images of cash note 500");
3806            init();
3807        }
3808    }
```

```
==> ./src/atm/gui/virtualslots/cashdispenser/notes/CashNote.java <==
package atm.gui.virtualslots.cashdispenser.notes;
import atm.utils.FetchImageNeeder;
public interface CashNote extends FetchImageNeeder {
}
==> ./src/atm/gui/virtualslots/cashdispenser/notes/CashNote100.java <==
package atm.gui.virtualslots.cashdispenser.notes;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import atm.utils.MyURLs;
public class CashNote100 implements CashNote {
    public static ImageIcon imageIcon;
    public static JLabel jLabel;
    public static int value;
    public static void init() throws MalformedURLException {
        imageIcon = new ImageIcon(new URL(MyURLs.IMAGE_NOTE100));
        jLabel = new JLabel(imageIcon);
        value = 100;
    }
    @Override
    public void fetchImage() throws IOException {
        System.out.println("fetching images of cash note 100");
        init();
    }
}
==> ./src/atm/gui/virtualslots/cashdispenser/notes/CashNote1000.java <==
package atm.gui.virtualslots.cashdispenser.notes;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import atm.utils.MyURLs;
public class CashNote1000 implements CashNote {
    public static ImageIcon imageIcon;
    public static JLabel jLabel;
    public static int value;
    public static void init() throws MalformedURLException {
        imageIcon = new ImageIcon(new URL(MyURLs.IMAGE_NOTE1000));
        jLabel = new JLabel(imageIcon);
        value = 1000;
    }
    @Override
    public void fetchImage() throws IOException {
        System.out.println("fetching images of cash note 1000");
        init();
    }
}
==> ./src/atm/gui/virtualslots/VirtualSlotsJFrame.java <==
package atm.gui.virtualslots;
import java.awt.GraphicsEnvironment;
import java.awt.Rectangle;
import java.util.Vector;
import javax.swing.BoxLayout;
import javax.swing.JFrame;
import atm.gui.MyGUISettings;
import atm.gui.monitor.mainscreen.WelcomeJPanel;
```

```
3869  import atm.gui.virtualslots.cardslot.CardSlotCardJPanel;
3870  import atm.gui.virtualslots.cashdispenser.CashDispenserJPanel;
3871  public class VirtualSlotsJFrame extends JFrame {
3872      /***/
3873      private static final long serialVersionUID = 1L;
3874      private static Vector<VirtualSlotsJFrame> contents = new Vector<
      VirtualSlotsJFrame>();
3875      private CardSlotCardJPanel cardSlotCardJPanel;
3876      private CashDispenserJPanel cashDispenserJPanel;
3877      public VirtualSlotsJFrame() {
3878          super("Virtual Slots");
3879          contents.add(this);
3880          setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
3881          setSize(400, 300);
3882          getContentPane().setLayout(
3883                  new BoxLayout(getContentPane(), BoxLayout.Y_AXIS));
3884          // getContentPane().setLayout(new WrapLayout());
3885          cardSlotCardJPanel = new CardSlotCardJPanel();
3886          getContentPane().add(cardSlotCardJPanel);
3887          cashDispenserJPanel = new CashDispenserJPanel();
3888          getContentPane().add(cashDispenserJPanel);
3889          myReset();
3890      }
3891      public void calcBounds() {
3892          setVisible(true);
3893          pack();
3894          Rectangle client = new Rectangle(
3895                  MyGUISettings.VIRTUAL_SLOTS_FRAME_WIDTH,
3896                  MyGUISettings.VIRTUAL_SLOTS_FRAME_HEIGHT);
3897          Rectangle screen = GraphicsEnvironment.getLocalGraphicsEnvironment()
3898                  .getMaximumWindowBounds().getBounds();
3899          int x = screen.width - client.width;
3900          int y = 0;
3901          setBounds(x, y, client.width, client.height);
3902      }
3903      public void calcBounds(float wRatio, float hRatio) {
3904          setVisible(true);
3905          pack();
3906          Rectangle client = new Rectangle(
3907                  Math.round(MyGUISettings.VIRTUAL_SLOTS_FRAME_WIDTH * wRatio),
3908                  Math.round(MyGUISettings.VIRTUAL_SLOTS_FRAME_HEIGHT * hRatio));
3909          Rectangle screen = GraphicsEnvironment.getLocalGraphicsEnvironment()
3910                  .getMaximumWindowBounds().getBounds();
3911          int x = screen.width - client.width;
3912          int y = 0;
3913          setBounds(x, y, client.width, client.height);
3914      }
3915      public void hideCardSlot() {
3916          cardSlotCardJPanel.setVisible(false);
3917          // calcBounds(1f, 2f);
3918      }
3919      public void myReset() {
3920          cardSlotCardJPanel.setVisible(true);
3921          cardSlotCardJPanel.switchToCard(CardSlotCardJPanel.STRING_SELECT_CARD);
3922          WelcomeJPanel.showMeStatic();
3923          calcBounds();
3924      }
3925      /** static connector to instance stuff **/
3926      public static void myResetStatic() {
3927          for (VirtualSlotsJFrame virtualSlotsJFrame : contents) {
```

```
3928                virtualSlotsJFrame.myReset();
3929            }
3930        }
3931        public static void hideCardSlotStatic() {
3932            for (VirtualSlotsJFrame virtualSlotsJFrame : contents) {
3933                virtualSlotsJFrame.hideCardSlot();
3934            }
3935        }
3936    }
3937    ==> ./src/atm/gui/virtualslots/cardslot/Card.java <==
3938    package atm.gui.virtualslots.cardslot;
3939    import java.awt.Image;
3940    import java.awt.event.ActionEvent;
3941    import java.awt.event.ActionListener;
3942    import java.io.IOException;
3943    import java.net.MalformedURLException;
3944    import java.net.URL;
3945    import java.util.Vector;
3946    import javax.swing.ImageIcon;
3947    import javax.swing.JButton;
3948    import javax.swing.JLabel;
3949    import atm.gui.MyGUISettings;
3950    import atm.utils.MyURLs;
3951    import atm.utils.FetchImageNeeder;
3952    public class Card implements FetchImageNeeder {
3953        private static Vector<Card> cards = new Vector<Card>();
3954        public ImageIcon imageIconBright;
3955        public ImageIcon imageIconDark;
3956        public JButton buttonInsert;
3957        public JButton buttonTake;
3958        public JLabel labelDark;
3959        public String accountNumber;
3960        /** static instances **/
3961        public static void init() throws MalformedURLException {
3962            cards.add(new Card(MyURLs.IMAGE_CARD1, MyURLs.IMAGE_CARD1_DARK, "12356"));
3963            cards.add(new Card(MyURLs.IMAGE_CARD2, MyURLs.IMAGE_CARD2_DARK, "12369"));
3964            cards.add(new Card(MyURLs.IMAGE_CARD3, MyURLs.IMAGE_CARD3_DARK, "45678"));
3965            cards.add(new Card(MyURLs.IMAGE_CARD4, MyURLs.IMAGE_CARD4_DARK, "E3545"));
3966        }
3967        public static Vector<Card> getCards() {
3968            return cards;
3969        }
3970        public static ActionListener getInsertActionListener(final Card card) {
3971            ActionListener insertCard = new ActionListener() {
3972                @Override
3973                public void actionPerformed(ActionEvent e) {
3974                    CardSlotCardJPanel.insertCardStatic(card);
3975                }
3976            };
3977            return insertCard;
3978        }
3979        public static ActionListener getTakeActionListener(final Card card) {
3980            ActionListener insertCard = new ActionListener() {
3981                @Override
3982                public void actionPerformed(ActionEvent e) {
3983                    CardSlotCardJPanel.takeCardStatic();
3984                }
3985            };
3986            return insertCard;
3987        }
```

```
3988        /** constructor **/
3989        public Card(String imageURLBright, String imageURLDark, String accountNumber)
3990                throws MalformedURLException {
3991            imageIconBright = new ImageIcon(new ImageIcon(new URL(imageURLBright))
3992                    .getImage().getScaledInstance(MyGUISettings.CARD_IMAGE_WIDTH,
3993                            MyGUISettings.CARD_IMAGE_HEIGHT, Image.SCALE_SMOOTH));
3994            imageIconDark = new ImageIcon(new ImageIcon(new URL(imageURLDark))
3995                    .getImage().getScaledInstance(MyGUISettings.CARD_IMAGE_WIDTH,
3996                            MyGUISettings.CARD_IMAGE_HEIGHT, Image.SCALE_SMOOTH));
3997            buttonInsert = new JButton(imageIconBright);
3998            buttonInsert.addActionListener(getInsertActionListener(this));
3999            buttonTake = new JButton(new ImageIcon(imageIconBright.getImage()));
4000            buttonTake.addActionListener(getTakeActionListener(this));
4001            labelDark = new JLabel(imageIconDark);
4002            this.accountNumber = accountNumber;
4003        }
4004        @Deprecated
4005        public Card() {
4006        }
4007        @Override
4008        public void fetchImage() throws IOException {
4009            System.out.println("fetching images of cards");
4010            init();
4011        }
4012    }
4013    ==> ./src/atm/gui/virtualslots/cardslot/SelectCardJPanel.java <==
4014    package atm.gui.virtualslots.cardslot;
4015    import javax.swing.JPanel;
4016    import webs.layout.CircleLayout;
4017    public class SelectCardJPanel extends JPanel {
4018        /***/
4019        private static final long serialVersionUID = 1L;
4020        public SelectCardJPanel() {
4021            setLayout(new CircleLayout());
4022            for (Card card : Card.getCards())
4023                add(card.buttonInsert);
4024            setPreferredSize(getMinimumSize());
4025        }
4026        public void myUpdateUI() {
4027            setPreferredSize(getMinimumSize());
4028            updateUI();
4029        }
4030    }
4031    ==> ./src/atm/gui/virtualslots/cardslot/CardInsideJPanel.java <==
4032    package atm.gui.virtualslots.cardslot;
4033    import java.util.Vector;
4034    import javax.swing.JPanel;
4035    import atm.gui.virtualslots.VirtualSlotsJFrame;
4036    public class CardInsideJPanel extends JPanel {
4037        /***/
4038        private static final long serialVersionUID = 1L;
4039        private static Vector<CardInsideJPanel> contents = new Vector<CardInsideJPanel
            >();
4040        private static Card card = null;
4041        /** constructor **/
4042        public CardInsideJPanel() {
4043            contents.add(this);
4044            setPreferredSize(getMinimumSize());
4045        }
4046        /** static methods **/
```

```
4047        public static boolean hasCard() {
4048            return (card != null);
4049        }
4050        public static void removeCard() {
4051            CardInsideJPanel.card = null;
4052            VirtualSlotsJFrame.hideCardSlotStatic();
4053        }
4054        public static Card getCard() {
4055            return card;
4056        }
4057        /** instance methods **/
4058        public void insertCard(Card card) {
4059            CardInsideJPanel.card = card;
4060            removeAll();
4061            add(card.labelDark);
4062            card.labelDark.setVisible(true);
4063            add(card.buttonTake);
4064            card.buttonTake.setVisible(false);
4065        }
4066        public void popCard() {
4067            card.labelDark.setVisible(false);
4068            card.buttonTake.setVisible(true);
4069        }
4070        public void myUpdateUI() {
4071            setPreferredSize(getMinimumSize());
4072            updateUI();
4073        }
4074        /** static connector to instance stuff **/
4075        public static void insertCardStatic(Card card) {
4076            for (CardInsideJPanel cardInsideJPanel : contents)
4077                cardInsideJPanel.insertCard(card);
4078        }
4079        public static void popCardStatic() {
4080            for (CardInsideJPanel cardInsideJPanel : contents)
4081                cardInsideJPanel.popCard();
4082        }
4083    }
4084    ==> ./src/atm/gui/virtualslots/cardslot/CardSlotCardJPanel.java <==
4085    package atm.gui.virtualslots.cardslot;
4086    import java.util.Vector;
4087    import atm.core.ATM;
4088    import atm.gui.monitor.mainscreen.MainScreenCardJPanel;
4089    import atm.gui.monitor.mainscreen.TakeCardJPanel;
4090    import atm.gui.monitor.mainscreen.TakeCashJPanel;
4091    import atm.gui.virtualslots.VirtualSlotsJFrame;
4092    import atm.gui.virtualslots.cashdispenser.CashDispenserJPanel;
4093    import myutils.gui.cardlayout.AbstractCardJPanel;
4094    public class CardSlotCardJPanel extends AbstractCardJPanel {
4095        /***/
4096        private static final long serialVersionUID = 1L;
4097        private static Vector<CardSlotCardJPanel> contents = new Vector<
          CardSlotCardJPanel>();
4098        public static final String STRING_SELECT_CARD = "Select Card";
4099        public static final String STRING_CARD_INSIDE = "Card Inside";
4100        public static final String STRING_EMPTY = "Empty";
4101        public static String STATE = "";
4102        private SelectCardJPanel selectCardJPanel;
4103        private CardInsideJPanel cardInsideJPanel;
4104        /** constructor **/
4105        public CardSlotCardJPanel() {
```

```
4106            contents.add(this);
4107        }
4108        @Override
4109        protected void myInit() {
4110            selectCardJPanel = new SelectCardJPanel();
4111            cardInsideJPanel = new CardInsideJPanel();
4112            addToCards(selectCardJPanel, STRING_SELECT_CARD);
4113            addToCards(cardInsideJPanel, STRING_CARD_INSIDE);
4114            addToCards(new EmptyJPanel(), STRING_EMPTY);
4115            switchToCard(STRING_SELECT_CARD);
4116        }
4117        /** static methods **/
4118        public static boolean hasCard() {
4119            return CardInsideJPanel.hasCard();
4120        }
4121        /** instance methods **/
4122        public void insertCard(Card card) {
4123            CardInsideJPanel.insertCardStatic(card);
4124            switchToCard(STRING_CARD_INSIDE);
4125            ATM.readCard(card);
4126        }
4127        public void popCard() {
4128            switchToCard(STRING_CARD_INSIDE);
4129            MainScreenCardJPanel
4130                    .switchToCardStatic(MainScreenCardJPanel.STRING_TAKE_CARD);
4131            CardInsideJPanel.popCardStatic();
4132        }
4133        @Override
4134        public void switchToCard(String label) {
4135            STATE = label;
4136            super.switchToCard(label);
4137            System.out.println(STATE);
4138            updateUI();
4139            myUpdateUI();
4140        }
4141        public void waitPopCard() {
4142            (new WaitPopCard()).start();
4143        }
4144        private void takeCard() {
4145            System.out.println("card taken by user");
4146            CardInsideJPanel.removeCard();
4147            if (CashDispenserJPanel.hasCashToBePopped) {
4148                System.out.println("going to pop cash");
4149                switchToCardStatic(STRING_EMPTY);
4150                TakeCashJPanel.showMe();
4151            } else {
4152                VirtualSlotsJFrame.myResetStatic();
4153            }
4154        }
4155        public void myUpdateUI() {
4156            if (selectCardJPanel != null)
4157                selectCardJPanel.myUpdateUI();
4158            if (cardInsideJPanel != null)
4159                cardInsideJPanel.myUpdateUI();
4160            setPreferredSize(getMinimumSize());
4161            updateUI();
4162        }
4163        /** private class **/
4164        private static class WaitPopCard extends Thread {
4165            @Override
```

```
4166            public void run() {
4167                try {
4168                    Thread.sleep(2000);
4169                } catch (InterruptedException e) {
4170                }
4171                CardSlotCardJPanel.popCardStatic();
4172            }
4173        }
4174        /** static-instance connector **/
4175        public static void switchToCardStatic(String label) {
4176            for (CardSlotCardJPanel cardSlotCardJPanel : contents) {
4177                cardSlotCardJPanel.switchToCard(label);
4178            }
4179        }
4180        public static void insertCardStatic(Card card) {
4181            for (CardSlotCardJPanel cardSlotCardJPanel : contents) {
4182                cardSlotCardJPanel.insertCard(card);
4183            }
4184        }
4185        public static void popCardStatic() {
4186            if (!CardInsideJPanel.hasCard())
4187                return;
4188            TakeCardJPanel.showMe();
4189            for (CardSlotCardJPanel cardSlotCardJPanel : contents) {
4190                cardSlotCardJPanel.popCard();
4191            }
4192        }
4193        public static void takeCardStatic() {
4194            for (CardSlotCardJPanel cardSlotCardJPanel : contents) {
4195                cardSlotCardJPanel.takeCard();
4196            }
4197        }
4198        public static void waitPopCardStatic() {
4199            for (CardSlotCardJPanel cardSlotCardJPanel : contents) {
4200                cardSlotCardJPanel.waitPopCard();
4201            }
4202        }
4203    }
4204    ==> ./src/atm/gui/virtualslots/cardslot/EmptyJPanel.java <==
4205    package atm.gui.virtualslots.cardslot;
4206    import javax.swing.JPanel;
4207    public class EmptyJPanel extends JPanel {
4208        /***/
4209        private static final long serialVersionUID = 1L;
4210        public EmptyJPanel() {
4211        }
4212    }
4213    ==> ./src/atm/gui/ATMGUILauncher.java <==
4214    package atm.gui;
4215    import java.io.IOException;
4216    import java.util.Vector;
4217    import javax.swing.UIManager;
4218    import javax.swing.UnsupportedLookAndFeelException;
4219    import atm.gui.keypad.KeyPadButtonIcons;
4220    import atm.gui.monitor.sidebuttons.SideButtons;
4221    import atm.gui.virtualslots.cardslot.Card;
4222    import atm.gui.virtualslots.cashdispenser.notes.CashNote100;
4223    import atm.gui.virtualslots.cashdispenser.notes.CashNote1000;
4224    import atm.gui.virtualslots.cashdispenser.notes.CashNote500;
4225    import atm.utils.FetchImageRunnable;
```

```
4226  import atm.utils.MyImages;
4227  public class ATMGUILauncher {
4228      JFrameManager frameManager;
4229      public ATMGUILauncher() throws IOException {
4230          System.out.println("fetching image in multi-thread mode");
4231          fetchImages();
4232          System.out.println("initializing GUI");
4233          setLookAndFeel();
4234          frameManager = new JFrameManager();
4235          frameManager.start();
4236      }
4237      private void setLookAndFeel() {
4238          String name = "";
4239          try {
4240              // name = "com.easynth.lookandfeel.EaSynthLookAndFeel";
4241              // name="com.alee.laf.WebLookAndFeel";
4242              UIManager.setLookAndFeel(name);
4243              System.out.println("loading native system look and feel (" + name
4244                      + ")");
4245          } catch (ClassNotFoundException | InstantiationException
4246                  | IllegalAccessException | UnsupportedLookAndFeelException e) {
4247              try {
4248                  name = "com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel";
4249                  UIManager.setLookAndFeel(name);
4250                  System.out.println("loading native system look and feel ("
4251                          + name + ")");
4252              } catch (ClassNotFoundException | InstantiationException
4253                      | IllegalAccessException | UnsupportedLookAndFeelException e1) {
4254                  try {
4255                      name = UIManager.getSystemLookAndFeelClassName();
4256                      UIManager.setLookAndFeel(name);
4257                      System.out.println("loading native system look and feel ("
4258                              + name + ")");
4259                  } catch (ClassNotFoundException | InstantiationException
4260                          | IllegalAccessException
4261                          | UnsupportedLookAndFeelException e2) {
4262                      System.out.println("loading cross platform look and feel ("
4263                              + UIManager.getCrossPlatformLookAndFeelClassName()
4264                              + ")");
4265                  }
4266              }
4267          }
4268      }
4269      /** Launch the application.*/
4270      public void start() {
4271          System.out.println();
4272          System.out.println("showing GUI");
4273          frameManager.start();
4274      }
4275      @SuppressWarnings("deprecation")
4276      public void fetchImages() throws IOException {
4277          // prepare
4278          Vector<FetchImageRunnable> runnables = new Vector<FetchImageRunnable>();
4279          Vector<Thread> threads = new Vector<Thread>();
4280          runnables.add(new FetchImageRunnable(new Card()));
4281          runnables.add(new FetchImageRunnable(new CashNote100()));
4282          runnables.add(new FetchImageRunnable(new CashNote500()));
4283          runnables.add(new FetchImageRunnable(new CashNote1000()));
4284          runnables.add(new FetchImageRunnable(new KeyPadButtonIcons()));
4285          runnables.add(new FetchImageRunnable(new SideButtons()));
```

```
4286            runnables.add(new FetchImageRunnable(new MyImages()));
4287            for (FetchImageRunnable fetchImageRunnable : runnables) {
4288                threads.add(new Thread(fetchImageRunnable));
4289            }
4290            // start all downloading thread
4291            for (Thread thread : threads) {
4292                thread.start();
4293            }
4294            // wait all downloading thread finished
4295            boolean allFinished;
4296            long startTime = System.currentTimeMillis();
4297            do {
4298                allFinished = true;
4299                for (FetchImageRunnable fetchImageRunnable : runnables) {
4300                    allFinished &= fetchImageRunnable.isFinished();
4301                }
4302                if (!allFinished) {
4303                    System.out.println("still downloading..."
4304                            + (System.currentTimeMillis() - startTime) / 1000.0
4305                            + " second(s) passed");
4306                    try {
4307                        Thread.sleep(1000);
4308                    } catch (InterruptedException e) {
4309                        // OS Level Error
4310                        System.out.println("OS Level Error");
4311                    }
4312                }
4313            } while (!allFinished);
4314            System.out.println("finished download==>"
4315                    + (System.currentTimeMillis() - startTime) / 1000.0
4316                    + " second(s) passed");
4317        }
4318    }
4319    ==> ./src/atm/gui/JFrameManager.java <==
4320    package atm.gui;
4321    import java.net.MalformedURLException;
4322    import atm.gui.keypad.KeypadJFrame;
4323    import atm.gui.monitor.MonitorJFrame;
4324    import atm.gui.virtualslots.VirtualSlotsJFrame;
4325    public class JFrameManager {
4326        private KeypadJFrame keypadJFrame;
4327        private MonitorJFrame monitorJFrame;
4328        private VirtualSlotsJFrame virtualSlotsJFrame;
4329        /** Create the application.
4330         * @throws MalformedURLException*/
4331        public JFrameManager() throws MalformedURLException {
4332            initialize();
4333        }
4334        /** Initialize the contents of the frame.
4335         * @throws MalformedURLException*/
4336        private void initialize() throws MalformedURLException {
4337            System.out.println("initializing keypad");
4338            keypadJFrame = new KeypadJFrame();
4339            System.out.println("initializing monitor");
4340            monitorJFrame = new MonitorJFrame();
4341            System.out.println("initializing virtual slots");
4342            virtualSlotsJFrame = new VirtualSlotsJFrame();
4343            // set size and location on screen
4344            System.out.println("showing keypad");
4345            keypadJFrame.calcBounds();
```

```
4346            System.out.println("showing virtual slots");
4347            virtualSlotsJFrame.calcBounds();
4348            System.out.println("showing monitor");
4349            monitorJFrame.calcBounds(MyGUISettings.MONITOR_FRAME_WIDTH,
4350                    MyGUISettings.MONITOR_FRAME_HEIGHT,
4351                    MyGUISettings.SIDE_BUTTON_MARGIN);
4352        }
4353        protected void start() {
4354        }
4355        public void end() {
4356            keypadJFrame.dispose();
4357            monitorJFrame.dispose();
4358            virtualSlotsJFrame.dispose();
4359            System.exit(0);
4360        }
4361    }
4362    ==> ./src/atm/gui/keypad/KeypadJFrame.java <==
4363    package atm.gui.keypad;
4364    import java.awt.GraphicsEnvironment;
4365    import java.awt.GridLayout;
4366    import java.awt.Rectangle;
4367    import java.awt.event.ActionEvent;
4368    import java.awt.event.ActionListener;
4369    import java.util.Vector;
4370    import javax.swing.JComponent;
4371    import javax.swing.JFrame;
4372    import javax.swing.JButton;
4373    import javax.swing.JPanel;
4374    import javax.swing.BoxLayout;
4375    import javax.swing.JPasswordField;
4376    import javax.swing.text.BadLocationException;
4377    import javax.swing.text.JTextComponent;
4378    import atm.core.ATM;
4379    import atm.gui.monitor.MonitorJFrame;
4380    import atm.gui.monitor.mainscreen.TransferJPanel;
4381    import atm.gui.monitor.mainscreen.WithDrawalJPanel;
4382    public class KeypadJFrame extends JFrame {
4383        /***/
4384        private static final long serialVersionUID = 1L;
4385        private static Vector<KeypadJFrame> contents = new Vector<KeypadJFrame>();
4386        public static final String STRING_MODE_PASSWORD = "Password";
4387        public static final String STRING_MODE_ACCOUNTNUMBER = "AccountNumber";
4388        public static final String STRING_MODE_AMOUNT = "Amount";
4389        public static final String STRING_MODE_CASH_AMOUNT = "Cash Amount";
4390        public static final String STRING_MODE_NULL = "Null";
4391        private String mode;
4392        private boolean dotEnable;
4393        private int maxLength;
4394        private JComponent keys[];
4395        private JPanel numberKeysJPanel;
4396        private JPanel functionKeysJPanel;
4397        private JTextComponent textComponent;
4398        // constructor sets up GUI
4399        public KeypadJFrame() {
4400            contents.add(this);
4401            setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
4402            setTitle("Keypad");
4403            getContentPane().setLayout(
4404                    new BoxLayout(getContentPane(), BoxLayout.X_AXIS));
4405            numberKeysJPanel = new JPanel(new GridLayout(4, 3));
```

```
4406            getContentPane().add(numberKeysJPanel);
4407            functionKeysJPanel = new JPanel(new GridLayout(4, 1));
4408            getContentPane().add(functionKeysJPanel);
4409            keys = new JComponent[16]; // array keys contains 16 JButtons
4410            // initialize all digit key buttons
4411            for (int i = 0; i <= 9; i++) {
4412                JButton button = new JButton(String.valueOf(i));
4413                keys[i] = button;
4414                button.addActionListener(getNumActionListener(String.valueOf(i)));
4415            }
4416            // initialize all function key buttons
4417            {
4418                JButton button = new JButton(KeyPadButtonIcons.IMAGEICON_CANCEL);
4419                keys[10] = button;
4420                button.addActionListener(getCancelActionListener());
4421            }
4422            {
4423                JButton button = new JButton(KeyPadButtonIcons.IMAGEICON_CLEAR);
4424                keys[11] = button;
4425                button.addActionListener(getClearActionListener());
4426            }
4427            {
4428                JButton button = new JButton(KeyPadButtonIcons.IMAGEICON_ENTER);
4429                keys[12] = button;
4430                button.addActionListener(getEnterActionListener());
4431            }
4432            keys[13] = new JPanel();
4433            {
4434                JButton button = new JButton("00");
4435                keys[14] = button;
4436                button.addActionListener(getNumActionListener("00"));
4437            }
4438            {
4439                JButton button = new JButton(".");
4440                keys[15] = button;
4441                button.addActionListener(getDotActionListener());
4442            }
4443            // add buttons to keyPadJPanel panel
4444            // 7 8 9
4445            for (int i = 7; i <= 9; i++)
4446                numberKeysJPanel.add(keys[i]);
4447            // 4 5 6
4448            for (int i = 4; i <= 6; i++)
4449                numberKeysJPanel.add(keys[i]);
4450            // 1 2 3
4451            for (int i = 1; i <= 3; i++)
4452                numberKeysJPanel.add(keys[i]);
4453            // 0 . 00
4454            numberKeysJPanel.add(keys[0]);
4455            numberKeysJPanel.add(keys[15]);
4456            numberKeysJPanel.add(keys[14]);
4457            for (int i = 10; i <= 13; i++)
4458                functionKeysJPanel.add(keys[i]);
4459            switchMode(STRING_MODE_NULL);
4460            switchTarget(null);
4461        } // end CalculatorFrame constructor
4462        /** instance methods **/
4463        private void switchMode(String stringModePassword) {
4464            mode = stringModePassword;
4465            switch (mode) {
```

```
4466            case STRING_MODE_PASSWORD:
4467                dotEnable = false;
4468                maxLength = 5;
4469                break;
4470            case STRING_MODE_AMOUNT:
4471                dotEnable = true;
4472                maxLength = 8;
4473                break;
4474            case STRING_MODE_ACCOUNTNUMBER:
4475                dotEnable = false;
4476                maxLength = 13;
4477                break;
4478            case STRING_MODE_CASH_AMOUNT:
4479                dotEnable = false;
4480                maxLength = 5;
4481                break;
4482            case STRING_MODE_NULL:
4483                dotEnable = false;
4484                maxLength = 0;
4485                break;
4486            }
4487        }
4488        public void switchTarget(JTextComponent textComponent) {
4489            this.textComponent = textComponent;
4490        }
4491        private boolean hasDot() throws BadLocationException {
4492            for (char c : getText().toCharArray())
4493                if (c == '.')
4494                    return true;
4495            return false;
4496        }
4497        public String getText() throws BadLocationException {
4498            return textComponent.getText();
4499        }
4500        public void insertText(String str) throws BadLocationException {
4501            textComponent.getDocument().insertString(
4502                    textComponent.getCaretPosition(), str, null);
4503        }
4504        public int getDecimalPlace(String text) {
4505            int decimalPlace = 0;
4506            boolean meetDot = false;
4507            for (char c : text.toCharArray()) {
4508                if (c == '.')
4509                    meetDot = true;
4510                else if (meetDot)
4511                    decimalPlace++;
4512            }
4513            return decimalPlace;
4514        }
4515        private ActionListener getNumActionListener(final String content) {
4516            return new ActionListener() {
4517                @Override
4518                public void actionPerformed(ActionEvent e) {
4519                    try {
4520                        if (textComponent.getDocument().getLength() >= maxLength)
4521                            return;
4522                        if ((getDecimalPlace(textComponent.getText()) < 2)
4523                                && !((textComponent.getDocument().getLength() == 0) &&
                                    (content == "00")))
4524                            insertText(content);
```

```
                } catch (BadLocationException e1) {
                    insertTextAlternative(content);
                } catch (NullPointerException e2) {
                }
            }
        };
    }
    private ActionListener getDotActionListener() {
        return new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    if ((!hasDot()) && dotEnable) {
                        if (textComponent.getDocument().getLength() == 0)
                            insertText("0");
                        insertText(".");
                    }
                } catch (BadLocationException e1) {
                    insertTextAlternative(".");
                } catch (NullPointerException e2) {
                }
            }
        };
    }
    private ActionListener getCancelActionListener() {
        return new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                MonitorJFrame.returnButtonClick();
            }
        };
    }
    private ActionListener getClearActionListener() {
        return new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    textComponent.setText("");
                } catch (NullPointerException e2) {
                    // just ignore this
                }
            }
        };
    }
    private ActionListener getEnterActionListener() {
        return new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                switch (mode) {
                case STRING_MODE_PASSWORD:
                    System.out.println("[Enter] password mode");
                    ATM.getATM().authenticateUser(
                            String.valueOf(((JPasswordField) textComponent)
                                    .getPassword()));
                    break;
                case STRING_MODE_CASH_AMOUNT:
                    WithDrawalJPanel.enterButtonClickStatic();
                    break;
                case STRING_MODE_ACCOUNTNUMBER:
                case STRING_MODE_AMOUNT:
```

```
                    TransferJPanel.enterKeyPressedStatic();
                    break;
                }
            }
        };
    }
    private void insertTextAlternative(String content) {
        textComponent.setText(textComponent.getText() + content);
    }
    public void calcBounds() {
        setVisible(true);
        pack();
        Rectangle client = getBounds();
        Rectangle screen = GraphicsEnvironment.getLocalGraphicsEnvironment()
                .getMaximumWindowBounds().getBounds();
        int x = screen.width - client.width - 10;
        int y = screen.height - client.height - 10;
        setLocation(x, y);
    }
    /** static connector to instance methods **/
    public static void switchTargetStatic(JTextComponent textComponent,
            String mode) {
        for (KeypadJFrame keypadJFrame : contents) {
            keypadJFrame.switchTarget(textComponent);
            keypadJFrame.switchMode(mode);
        }
    }
    public static void switchModeStatic(String mode) {
        for (KeypadJFrame keypadJFrame : contents) {
            keypadJFrame.switchMode(mode);
        }
    }
    public static String getModeStatic() {
        for (KeypadJFrame keypadJFrame : contents) {
            return keypadJFrame.mode;
        }
        return null;
    }
} // end class CalculatorFrame
==> ./src/atm/gui/keypad/KeyPadButtonIcons.java <==
package atm.gui.keypad;
import java.awt.Image;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import javax.swing.ImageIcon;
import atm.gui.MyGUISettings;
import atm.gui.virtualslots.cashdispenser.notes.CashNote;
import atm.utils.MyURLs;
public class KeyPadButtonIcons implements CashNote {
    public static ImageIcon IMAGEICON_ENTER;
    public static ImageIcon IMAGEICON_CANCEL;
    public static ImageIcon IMAGEICON_CLEAR;
    public static void init() throws MalformedURLException {
        IMAGEICON_ENTER = new ImageIcon(new ImageIcon(new URL(
                MyURLs.IMAGE_ENTER)).getImage().getScaledInstance(
                MyGUISettings.FUNCTION_BUTTON_WIDTH,
                MyGUISettings.FUNCTION_BUTTON_HEIGHT, Image.SCALE_SMOOTH));
        IMAGEICON_CANCEL = new ImageIcon(new ImageIcon(new URL(
                MyURLs.IMAGE_CANCEL)).getImage().getScaledInstance(
```

```
                MyGUISettings.FUNCTION_BUTTON_WIDTH,
                MyGUISettings.FUNCTION_BUTTON_HEIGHT, Image.SCALE_SMOOTH));
        IMAGEICON_CLEAR = new ImageIcon(new ImageIcon(new URL(
                MyURLs.IMAGE_CLEAR)).getImage().getScaledInstance(
                MyGUISettings.FUNCTION_BUTTON_WIDTH,
                MyGUISettings.FUNCTION_BUTTON_HEIGHT, Image.SCALE_SMOOTH));
    }
    @Override
    public void fetchImage() throws IOException {
        System.out.println("fetching images of keypads");
        init();
    }
}
==> ./src/atm/ATMLauncher.java <==
package atm;
import java.io.IOException;
import javax.swing.JOptionPane;
import bank.BankDatabase;
import atm.core.ATM;
import atm.core.CashDispenser;
import atm.gui.ATMGUILauncher;
import atm.utils.MyStrings;
public class ATMLauncher {
    private ATMGUILauncher atmguiLauncher;
    public ATMLauncher() {
        BankDatabase.init();
        CashDispenser.init();
        ATM.initStatic();
        try {
            atmguiLauncher = new ATMGUILauncher();
        } catch (IOException e) {
            System.out.println(MyStrings.INTERNET_ERROR);
            JOptionPane.showMessageDialog(null, MyStrings.INTERNET_ERROR,
            "Internet Error",
                    JOptionPane.ERROR_MESSAGE);
            e.printStackTrace();
        }
    }
    public void start() {
        System.out.println("start GUI launcher");
        atmguiLauncher.start();
    }
}
==> ./src/launcher/ATMCaseStudy.java <==
package launcher;
import atm.ATMLauncher;
// ATMCaseStudy.java
// Driver program for the ATM case study
/**
 * Standard program entry point Launch the application.
 */
public class ATMCaseStudy {
    // main method creates and runs the ATM
    public static void main(String[] args) {
        System.out.println("ATMCaseStudy start");
        // ATM theATM = new ATM();
        // theATM.run();
        ATMLauncher atmLauncher = new ATMLauncher();
        System.out.println("start ATM Launcher");
        atmLauncher.start();
```

```
    } // end main
} // end class ATMCaseStudy
==> ./src/myutils/Utils.java <==
package myutils;
import java.util.Calendar;
import java.util.List;
import java.util.Random;
import java.util.Vector;
public class Utils {
    public static Random random = new Random(System.currentTimeMillis());
    public static Vector<Object> getShuffled(Vector<Object> ori) {
        Vector<Object> result = new Vector<Object>();
        for (Object o : ori)
            result.add(o);
        int t;
        Object tmp;
        int size = ori.size();
        for (int i = 0; i < size; i++) {
            t = random.nextInt(size);
            tmp = result.get(i);
            result.set(i, result.get(t));
            result.set(t, tmp);
        }
        return result;
    }
    public static String StringListToString(List<String> lines, String symbol) {
        String result = "";
        for (String line : lines)
            result += line + symbol;
        return result;
    }
    public static java.sql.Timestamp getCurrentTimestamp() {
        return new java.sql.Timestamp(Calendar.getInstance().getTime()
                .getTime());
    }
}
==> ./src/myutils/gui/cardlayout/AbstractCardJPanel.java <==
package myutils.gui.cardlayout;
import java.awt.CardLayout;
import java.awt.Component;
import javax.swing.JPanel;
public abstract class AbstractCardJPanel extends JPanel {
    /***/
    private static final long serialVersionUID = -6243990180583440256L;
    public CardLayout cardLayout;
    public AbstractCardJPanel() {
        cardLayout = new CardLayout();
        setLayout(cardLayout);
        myInit();
    }
    protected abstract void myInit();
    public void addToCards(Component component, String label) {
        add(component);
        cardLayout.addLayoutComponent(component, label);
    }
    public void switchToCard(String label) {
        cardLayout.show(this, label);
    }
}
==> ./src/myutils/gui/cardlayout/AbstractCardJFrame.java <==
```

```
4764 package myutils.gui.cardlayout;
4765 import java.awt.CardLayout;
4766 import java.awt.Component;
4767 import javax.swing.JFrame;
4768 public abstract class AbstractCardJFrame extends JFrame {
4769     /***/
4770     private static final long serialVersionUID = -1690280875210999260L;
4771     public CardLayout cardLayout;
4772     public AbstractCardJFrame(String title) {
4773         setVisible(false);
4774         setTitle(title);
4775         cardLayout = new CardLayout();
4776         getContentPane().setLayout(cardLayout);
4777         myInit();
4778         setVisible(true);
4779     }
4780     protected abstract void myInit();
4781     public void addToCards(Component component, String label) {
4782         add(component);
4783         cardLayout.addLayoutComponent(component, label);
4784     }
4785     public void switchToCard(String label) {
4786         cardLayout.show(getContentPane(), label);
4787     }
4788 }
4789 ==> ./src/myutils/gui/MyImageUtils.java <==
4790 package myutils.gui;
4791 import java.awt.Image;
4792 import java.awt.image.BufferedImage;
4793 import javax.swing.ImageIcon;
4794 public class MyImageUtils {
4795     public static ImageIcon scaleImageIconByHeight(ImageIcon image, int height) {
4796         int width = (int) Math.round(image.getIconWidth() * (height * 1.0 / image.
          getIconHeight()));
4797         return new ImageIcon(image.getImage().getScaledInstance(width, height,
          Image.SCALE_SMOOTH));
4798     }
4799     public static Image scaleImageByHeight(Image image, int height) {
4800         BufferedImage bufferedImage = (BufferedImage) image;
4801         int width = (int) Math.round(bufferedImage.getWidth() * (height * 1.0 /
          bufferedImage.getHeight()));
4802         return image.getScaledInstance(width, height, Image.SCALE_SMOOTH);
4803     }
4804 }
4805
```