# 16TIN2054-Teknik Pemrograman Praktek

Week 8 – PLOO


Dikerjakan oleh:

Muhammad Azhar Alauddin – 201524013

1A – D4 Jurusan Teknik Komputer dan Informatika


Tugas ini dikumpulkan untuk memenuhi sebagian persyaratan kelulusan matakuliah Teknik Pemrograman Praktek


**Program Studi D4 Teknik Informatika**

**Politeknik Negeri Bandung**


**2020/2021**

# Week 8 : PLOO

## Assignment Detail:

- Anda diberikan sebuah program tentang transaksi di ATM. Berberapa fitur pada Program tersebut telah berfungsi, seperti Login dan menampilkan menu. Tugas anda adalah, pelajari source program tersebut, lalu lengkapi agar output yang dihasilkan seperti pada OutputATM.txt.
- Berikut ini poin-poin yang harus anda lengkapi :
  1. Berhasil Login menggunakan account number dan pin yang diminta
  2. menampilkan Balance Information
  3. memilih penarikan (withdrawal)
  4. menampilkan Balance Information setelah dilakukan penarikan
  5. Deposit funds
  6. menampilkan Balance Information setelah dilakukan deposit
  7. exit
- Untuk menyelesaikan kasus ini pelajari ulang materi tentang inheritance, encapsulation dan abstract class

## Account.java

```java
// Account.java
// Represents a bank account

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

  // Account constructor initializes attributes
  public Account(int theAccountNumber, int thePIN,
    double theAvailableBalance, double theTotalBalance) {
    accountNumber = theAccountNumber;
    pin = thePIN;
    availableBalance = theAvailableBalance;
    totalBalance = theTotalBalance;
  }

  // determines whether a user-specified PIN matches PIN in Account
  public boolean validatePIN(int userPIN) {
    if (userPIN == pin) {
      return true;
    }
    else {
      return false;
    }
  }

  // returns available balance
  public double getAvailableBalance() {
```

```
    return availableBalance;
  }

  // returns the total balance
  public double getTotalBalance() {
    return totalBalance;
  }

  // credits an amount to the account
  public void credit(double amount) {
    totalBalance += amount; // add to total balance
  }

  // debits an amount from the account
  public void debit(double amount) {
    availableBalance -= amount; // subtract from available balance
    totalBalance -= amount; // subtract from total balance
  }

  // returns account number
  public int getAccountNumber() {
    return accountNumber;
  }
}



/**************************************************************************
 * (C) Copyright 1992-2018 by Deitel & Associates, Inc. and               *
 * Pearson Education, Inc. All Rights Reserved.                           *
 *                                                                        *
 * DISCLAIMER: The authors and publisher of this book have used their     *
 * best efforts in preparing the book. These efforts include the          *
 * development, research, and testing of the theories and programs        *
 * to determine their effectiveness. The authors and publisher make       *
 * no warranty of any kind, expressed or implied, with regard to these    *
 * programs or to the documentation contained in these books. The authors *
 * and publisher shall not be liable in any event for incidental or       *
 * consequential damages in connection with, or arising out of, the       *
 * furnishing, performance, or use of these programs.                     *
 *************************************************************************/
```

## BankDataBase.java

```
package iniPackage;

// BankDatabase.java
// Represents the bank account information database

public class BankDatabase {
  private Account[] accounts; // array of Accounts

  // no-argument BankDatabase constructor initializes accounts
  public BankDatabase() {
    accounts = new Account[2]; // just 2 accounts for testing
    accounts[0] = new Account(12345, 54321, 1000.0, 1200.0);
    accounts[1] = new Account(98765, 56789, 200.0, 200.0);
```

```
  }

  // retrieve Account object containing specified account number
  private Account getAccount(int accountNumber) {
    // loop through accounts searching for matching account number
    for (Account currentAccount : accounts) {
      // return current account if match found
      if (currentAccount.getAccountNumber() == accountNumber) {
        return currentAccount;
      }
    }

    return null; // if no matching account was found, return null
  }

  // determine whether user-specified account number and PIN match
  // those of an account in the database
  public boolean authenticateUser(int userAccountNumber, int userPIN) {
    // attempt to retrieve the account with the account number
    Account userAccount = getAccount(userAccountNumber);

    // if account exists, return result of Account method validatePIN
    if (userAccount != null) {
      return userAccount.validatePIN(userPIN);
    }
    else {
      return false; // account number not found, so return false
    }
  }

  // return available balance of Account with specified account number
  public double getAvailableBalance(int userAccountNumber) {
    return getAccount(userAccountNumber).getAvailableBalance();
  }

  // return total balance of Account with specified account number
  public double getTotalBalance(int userAccountNumber) {
    return getAccount(userAccountNumber).getTotalBalance();
  }

  // credit an amount to Account with specified account number
  public void credit(int userAccountNumber, double amount) {
    getAccount(userAccountNumber).credit(amount);
  }

  // debit an amount from Account with specified account number
  public void debit(int userAccountNumber, double amount) {
    getAccount(userAccountNumber).debit(amount);
  }
}
```

```
 * DISCLAIMER: The authors and publisher of this book have used their     *
 * best efforts in preparing the book. These efforts include the          *
 * development, research, and testing of the theories and programs        *
 * to determine their effectiveness. The authors and publisher make       *
 * no warranty of any kind, expressed or implied, with regard to these    *
 * programs or to the documentation contained in these books. The authors *
 * and publisher shall not be liable in any event for incidental or       *
 * consequential damages in connection with, or arising out of, the       *
 * furnishing, performance, or use of these programs.                     *
 *************************************************************************/
```

## Withdrawal.java

```
package iniPackage;

public class Withdrawal extends Transaction {
  private int amount; // amount to withdraw
  private Keypad keypad; // reference to keypad
  private CashDispenser cashDispenser; // reference to cash dispenser

  // constant corresponding to menu option to cancel
  private final static int CANCELED = 6;

  // Withdrawal constructor
  public Withdrawal(int userAccountNumber, Screen atmScreen,
    BankDatabase atmBankDatabase, Keypad atmKeypad,
    CashDispenser atmCashDispenser) {

    // initialize superclass variables
    super(userAccountNumber, atmScreen, atmBankDatabase);

    // initialize references to keypad and cash dispenser
    keypad = atmKeypad;
    cashDispenser = atmCashDispenser;
  }

  // perform transaction
  @Override
  public void execute() {
    boolean cashDispensed = false; // cash was not dispensed yet
    double availableBalance; // amount available for withdrawal

    // get references to bank database and screen
    BankDatabase bankDatabase = getBankDatabase();
    Screen screen = getScreen();

    // loop until cash is dispensed or the user cancels
    do {
      // obtain a chosen withdrawal amount from the user
      amount = displayMenuOfAmounts();

      // check whether user chose a withdrawal amount or canceled
      if (amount != CANCELED) {
        // get available balance of account involved
        availableBalance =
          bankDatabase.getAvailableBalance(getAccountNumber());
```

```
        // check whether the user has enough money in the account
        if (amount <= availableBalance) {
          // check whether the cash dispenser has enough money
          if (cashDispenser.isSufficientCashAvailable(amount)) {
            // update the account involved to reflect the withdrawal
            bankDatabase.debit(getAccountNumber(), amount);

            cashDispenser.dispenseCash(amount); // dispense cash
            cashDispensed = true; // cash was dispensed

            // instruct user to take cash
            screen.displayMessageLine("\nYour cash has been" +
              " dispensed. Please take your cash now.");
          }
          else { // cash dispenser does not have enough cash
            screen.displayMessageLine(
              "\nInsufficient cash available in the ATM." +
              "\n\nPlease choose a smaller amount.");
          }
        }
        else { // not enough money available in user's account
          screen.displayMessageLine(
            "\nInsufficient funds in your account." +
            "\n\nPlease choose a smaller amount.");
        }
      }
      else { // user chose cancel menu option
        screen.displayMessageLine("\nCanceling transaction...");
        return; // return to main menu because user canceled
      }
    } while (!cashDispensed);
  }

  // display a menu of withdrawal amounts and the option to cancel;
  // return the chosen amount or 0 if the user chooses to cancel
  private int displayMenuOfAmounts() {
    int userChoice = 0; // local variable to store return value

    Screen screen = getScreen(); // get screen reference

    // array of amounts to correspond to menu numbers
    int[] amounts = {0, 20, 40, 60, 100, 200};

    // loop while no valid choice has been made
    while (userChoice == 0) {
      // display the withdrawal menu
      screen.displayMessageLine("\nWithdrawal Menu:");
      screen.displayMessageLine("1 - $20");
      screen.displayMessageLine("2 - $40");
      screen.displayMessageLine("3 - $60");
      screen.displayMessageLine("4 - $100");
      screen.displayMessageLine("5 - $200");
      screen.displayMessageLine("6 - Cancel transaction");
      screen.displayMessage("\nChoose a withdrawal amount: ");

      int input = keypad.getInput(); // get user input through keypad

      // determine how to proceed based on the input value
```

```
        switch (input) {
          case 1: // if the user chose a withdrawal amount
          case 2: // (i.e., chose option 1, 2, 3, 4 or 5), return the
          case 3: // corresponding amount from amounts array
          case 4:
          case 5:
            userChoice = amounts[input]; // save user's choice
            break;
          case CANCELED: // the user chose to cancel
            userChoice = CANCELED; // save user's choice
            break;
          default: // the user did not enter a value from 1-6
            screen.displayMessageLine(
              "\nInvalid selection. Try again.");
        }
      }

      return userChoice; // return withdrawal amount or CANCELED
    }
}



/**************************************************************************
 * (C) Copyright 1992-2018 by Deitel & Associates, Inc. and               *
 * Pearson Education, Inc. All Rights Reserved.                           *
 *                                                                        *
 * DISCLAIMER: The authors and publisher of this book have used their     *
 * best efforts in preparing the book. These efforts include the          *
 * development, research, and testing of the theories and programs        *
 * to determine their effectiveness. The authors and publisher make       *
 * no warranty of any kind, expressed or implied, with regard to these    *
 * programs or to the documentation contained in these books. The authors *
 * and publisher shall not be liable in any event for incidental or       *
 * consequential damages in connection with, or arising out of, the       *
 * furnishing, performance, or use of these programs.                     *
 *************************************************************************/
```

## Transaction.java

```
package iniPackage;

// Transaction.java
// Abstract superclass Transaction represents an ATM transaction

public abstract class Transaction {
  private int accountNumber; // indicates account involved
  private Screen screen; // ATM's screen
  private BankDatabase bankDatabase; // account info database

  // Transaction constructor invoked by subclasses using super()
  public Transaction(int userAccountNumber, Screen atmScreen,
    BankDatabase atmBankDatabase) {

    accountNumber = userAccountNumber;
    screen = atmScreen;
    bankDatabase = atmBankDatabase;
```

```
  }

  // return account number
  public int getAccountNumber() {
    return accountNumber;
  }

  // return reference to screen
  public Screen getScreen() {
    return screen;
  }

  // return reference to bank database
  public BankDatabase getBankDatabase() {
    return bankDatabase;
  }

  // perform the transaction (overridden by each subclass)
  abstract public void execute();
}
```

## Screen.java

```
package iniPackage;

// Screen.java
// Represents the screen of the ATM

public class Screen {
  // display a message without a carriage return
  public void displayMessage(String message) {
    System.out.print(message);
  }

  // display a message with a carriage return
  public void displayMessageLine(String message) {
    System.out.println(message);
  }

  // displays a dollar amount
  public void displayDollarAmount(double amount) {
    System.out.printf("$%,.2f", amount);
  }
}



/**************************************************************************
 * (C) Copyright 1992-2018 by Deitel & Associates, Inc. and               *
 * Pearson Education, Inc. All Rights Reserved.                           *
 *                                                                        *
 * DISCLAIMER: The authors and publisher of this book have used their     *
 * best efforts in preparing the book. These efforts include the          *
 * development, research, and testing of the theories and programs        *
 * to determine their effectiveness. The authors and publisher make       *
 * no warranty of any kind, expressed or implied, with regard to these    *
```

```
 * programs or to the documentation contained in these books. The authors *
 * and publisher shall not be liable in any event for incidental or       *
 * consequential damages in connection with, or arising out of, the       *
 * furnishing, performance, or use of these programs.                     *
 *************************************************************************/
```

## Keypad.java

```
package iniPackage;

// Keypad.java
// Represents the keypad of the ATM
import java.util.Scanner; // program uses Scanner to obtain user input

public class Keypad {
  private Scanner input; // reads data from the command line

  // no-argument constructor initializes the Scanner
  public Keypad() {
    input = new Scanner(System.in);
  }

  // return an integer value entered by user
  public int getInput() {
    return input.nextInt(); // we assume that user enters an integer
  }

}



/**************************************************************************
 * (C) Copyright 1992-2018 by Deitel & Associates, Inc. and               *
 * Pearson Education, Inc. All Rights Reserved.                           *
 *                                                                        *
 * DISCLAIMER: The authors and publisher of this book have used their     *
 * best efforts in preparing the book. These efforts include the          *
 * development, research, and testing of the theories and programs        *
 * to determine their effectiveness. The authors and publisher make       *
 * no warranty of any kind, expressed or implied, with regard to these    *
 * programs or to the documentation contained in these books. The authors *
 * and publisher shall not be liable in any event for incidental or       *
 * consequential damages in connection with, or arising out of, the       *
 * furnishing, performance, or use of these programs.                     *
 *************************************************************************/
```

## DepositSlot.java

```
package iniPackage;

// DepositSlot.java
// Represents the deposit slot of the ATM

public class DepositSlot {
  // indicates whether envelope was received (always returns true,
  // because this is only a software simulation of a real deposit slot)
  public boolean isEnvelopeReceived() {
```

```
    return true; // deposit envelope was received
  }
}

/**************************************************************************
 * (C) Copyright 1992-2018 by Deitel & Associates, Inc. and               *
 * Pearson Education, Inc. All Rights Reserved.                           *
 *                                                                        *
 * DISCLAIMER: The authors and publisher of this book have used their     *
 * best efforts in preparing the book. These efforts include the          *
 * development, research, and testing of the theories and programs        *
 * to determine their effectiveness. The authors and publisher make       *
 * no warranty of any kind, expressed or implied, with regard to these    *
 * programs or to the documentation contained in these books. The authors *
 * and publisher shall not be liable in any event for incidental or       *
 * consequential damages in connection with, or arising out of, the       *
 * furnishing, performance, or use of these programs.                     *
 *************************************************************************/
```

## Deposit.java

```
package iniPackage;

public class Deposit extends Transaction {
  private double amount; // amount to deposit
  private Keypad keypad; // reference to keypad
  private DepositSlot depositSlot; // reference to deposit slot
  private final static int CANCELED = 0; // constant for cancel option

  // Deposit constructor
  public Deposit(int userAccountNumber, Screen atmScreen,
    BankDatabase atmBankDatabase, Keypad atmKeypad,
    DepositSlot atmDepositSlot) {

    // initialize superclass variables
    super(userAccountNumber, atmScreen, atmBankDatabase);

    // initialize references to keypad and deposit slot
    keypad = atmKeypad;
    depositSlot = atmDepositSlot;
  }

  // perform transaction
  @Override
  public void execute() {
    BankDatabase bankDatabase = getBankDatabase(); // get reference
    Screen screen = getScreen(); // get reference

    amount = promptForDepositAmount(); // get deposit amount from user

    // check whether user entered a deposit amount or canceled
    if (amount != CANCELED) {
      // request deposit envelope containing specified amount
      screen.displayMessage(
        "\nPlease insert a deposit envelope containing ");
      screen.displayDollarAmount(amount);
      screen.displayMessageLine(".");
```

```
      // receive deposit envelope
      boolean envelopeReceived = depositSlot.isEnvelopeReceived();

      // check whether deposit envelope was received
      if (envelopeReceived) {
        screen.displayMessageLine("\nYour envelope has been " +
          "received.\nNOTE: The money just deposited will not " +
          "be available until we verify the amount of any " +
          "enclosed cash and your checks clear.");

        // credit account to reflect the deposit

        bankDatabase.credit(getAccountNumber(), amount);

      }
      else { // deposit envelope not received
        screen.displayMessageLine("\nYou did not insert an " +
          "envelope, so the ATM has canceled your transaction.");
      }
    }
    else { // user canceled instead of entering amount
      screen.displayMessageLine("\nCanceling transaction...");
    }
  }

  // prompt user to enter a deposit amount in cents
  private double promptForDepositAmount() {
    Screen screen = getScreen(); // get reference to screen

    // display the prompt
    screen.displayMessage("\nPlease enter a deposit amount in " +
      "CENTS (or 0 to cancel): ");
    int input = keypad.getInput(); // receive input of deposit amount

    // check whether the user canceled or entered a valid amount
    if (input == CANCELED) {
      return CANCELED;
    }
    else {
      return (double) input / 100; // return dollar amount
    }
  }
}



/**************************************************************************
 * (C) Copyright 1992-2018 by Deitel & Associates, Inc. and               *
 * Pearson Education, Inc. All Rights Reserved.                           *
 *                                                                        *
 * DISCLAIMER: The authors and publisher of this book have used their     *
 * best efforts in preparing the book. These efforts include the          *
 * development, research, and testing of the theories and programs        *
 * to determine their effectiveness. The authors and publisher make       *
 * no warranty of any kind, expressed or implied, with regard to these    *
 * programs or to the documentation contained in these books. The authors *
 * and publisher shall not be liable in any event for incidental or       *
```

```
 * consequential damages in connection with, or arising out of, the      *
 * furnishing, performance, or use of these programs.                    *
 *************************************************************************/
```

## CashDispenser.java

```
package iniPackage;

public class CashDispenser {
  // the default initial number of bills in the cash dispenser
  private final static int INITIAL_COUNT = 500;
  private int count; // number of $20 bills remaining

  // no-argument CashDispenser constructor initializes count to default
  public CashDispenser() {
    count = INITIAL_COUNT; // set count attribute to default
  }

  // simulates dispensing of specified amount of cash
  public void dispenseCash(int amount) {
    int billsRequired = amount / 20; // number of $20 bills required
    count -= billsRequired; // update the count of bills
  }

  // indicates whether cash dispenser can dispense desired amount
  public boolean isSufficientCashAvailable(int amount) {
    int billsRequired = amount / 20; // number of $20 bills required

    if (count >= billsRequired) {
      return true; // enough bills available
    }
    else {
      return false; // not enough bills available
    }
  }
}



/*************************************************************************
 * (C) Copyright 1992-2018 by Deitel & Associates, Inc. and              *
 * Pearson Education, Inc. All Rights Reserved.                          *
 *                                                                       *
 * DISCLAIMER: The authors and publisher of this book have used their    *
 * best efforts in preparing the book. These efforts include the         *
 * development, research, and testing of the theories and programs       *
 * to determine their effectiveness. The authors and publisher make      *
 * no warranty of any kind, expressed or implied, with regard to these   *
 * programs or to the documentation contained in these books. The authors *
 * and publisher shall not be liable in any event for incidental or      *
 * consequential damages in connection with, or arising out of, the      *
 * furnishing, performance, or use of these programs.                    *
 *************************************************************************/
```

## BalanceInquiry.java

```
package iniPackage;
```

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

public class BalanceInquiry extends Transaction {
  // BalanceInquiry constructor
  public BalanceInquiry(int userAccountNumber, Screen atmScreen,
    BankDatabase atmBankDatabase) {

    super(userAccountNumber, atmScreen, atmBankDatabase);
  }

  // performs the transaction
  @Override
  public void execute() {
    // get references to bank database and screen
    BankDatabase bankDatabase = getBankDatabase();
    Screen screen = getScreen();

    // get the available balance for the account involved
    double availableBalance =
      bankDatabase.getAvailableBalance(getAccountNumber());

    // get the total balance for the account involved
    double totalBalance =
      bankDatabase.getTotalBalance(getAccountNumber());

    // display the balance information on the screen
    screen.displayMessageLine("\nBalance Information:");
    screen.displayMessage(" - Available balance: ");
    screen.displayDollarAmount(availableBalance);
    screen.displayMessage("\n - Total balance:    ");
    screen.displayDollarAmount(totalBalance);
    screen.displayMessageLine("");
  }
}
```

## ATM.java

```java
package iniPackage;

public class ATM {
  private boolean userAuthenticated; // whether user is authenticated
  private int currentAccountNumber; // current user's account number
  private Screen screen; // ATM's screen
  private Keypad keypad; // ATM's keypad
  private CashDispenser cashDispenser; // ATM's cash dispenser
  private DepositSlot depositSlot; // ATM's deposit slot
  private BankDatabase bankDatabase; // account information database

  // constants corresponding to main menu options
  private static final int BALANCE_INQUIRY = 1;
  private static final int WITHDRAWAL = 2;
  private static final int DEPOSIT  = 3;
  private static final int EXIT = 4;

  // no-argument ATM constructor initializes instance variables
```

```
public ATM() {
  userAuthenticated = false; // user is not authenticated to start
  currentAccountNumber = 0; // no current account number to start
  screen = new Screen(); // create screen
  keypad = new Keypad(); // create keypad
  cashDispenser = new CashDispenser(); // create cash dispenser
  depositSlot = new DepositSlot(); // create deposit slot
  bankDatabase = new BankDatabase(); // create acct info database
}

// start ATM
public void run() {
  // deleted the outer loop so program will stop if user choose exit menu
  // loop while user is not yet authenticated
    while (!userAuthenticated) {
      screen.displayMessageLine("\nWelcome!");
      authenticateUser(); // authenticate user
    }

    performTransactions(); // user is now authenticated
    userAuthenticated = false; // reset before next ATM session
    currentAccountNumber = 0; // reset before next ATM session
    screen.displayMessageLine("\nThank you! Goodbye!");

}

// attempts to authenticate user against database
private void authenticateUser() {
  screen.displayMessage("\nPlease enter your account number: ");
  int accountNumber = keypad.getInput(); // input account number
  screen.displayMessage("\nEnter your PIN: "); // prompt for PIN
  int pin = keypad.getInput(); // input PIN

  // set userAuthenticated to boolean value returned by database
  userAuthenticated =
    bankDatabase.authenticateUser(accountNumber, pin);

  // check whether authentication succeeded
  if (userAuthenticated) {
    currentAccountNumber = accountNumber; // save user's account #
  }
  else {
    screen.displayMessageLine(
      "Invalid account number or PIN. Please try again.");
  }
}

// display the main menu and perform transactions
private void performTransactions() {
  // local variable to store transaction currently being processed
  Transaction currentTransaction = null;

  boolean userExited = false; // user has not chosen to exit

  // loop while user has not chosen option to exit system
  while (!userExited) {
    // show main menu and get user selection
    int mainMenuSelection = displayMainMenu();
```

```
      currentTransaction = createTransaction(mainMenuSelection);
      // decide how to proceed based on user's menu selection
      switch (mainMenuSelection) {
        // user chose to perform one of three transaction types
        case BALANCE_INQUIRY:
          currentTransaction.execute();
          break;
        case WITHDRAWAL:
          currentTransaction.execute();
          break;
        case DEPOSIT:
          currentTransaction.execute();
          break;
        case EXIT: // user chose to terminate session
          screen.displayMessageLine("\nExiting the system...");
          userExited = true; // this ATM session should end
          break;
        default: // user did not enter an integer from 1-4
          screen.displayMessageLine(
            "\nYou did not enter a valid selection. Try again.");
          break;
      }
    }
  }

  // display the main menu and return an input selection
  private int displayMainMenu() {
    screen.displayMessageLine("\nMain menu:");
    screen.displayMessageLine("1 - View my balance");
    screen.displayMessageLine("2 - Withdraw cash");
    screen.displayMessageLine("3 - Deposit funds");
    screen.displayMessageLine("4 - Exit\n");
    screen.displayMessage("Enter a choice: ");
    return keypad.getInput(); // return user's selection
  }

  // return object of specified Transaction subclass
  private Transaction createTransaction(int type) {
    Transaction temp = null; // temporary Transaction variable

    // determine which type of Transaction to create
    switch (type) {
      case BALANCE_INQUIRY: // create new BalanceInquiry transaction
        temp = new BalanceInquiry(
          currentAccountNumber, screen, bankDatabase);
        break;
      case WITHDRAWAL: // create new Withdrawal transaction
        temp = new Withdrawal(currentAccountNumber,
          screen, bankDatabase, keypad, cashDispenser);
        break;
      case DEPOSIT: // create new Deposit transaction
        temp = new Deposit(currentAccountNumber, screen, bankDatabase, keypad, depositSlot);
        break;
    }

    return temp; // return the newly created object
  }
}
```

## ATMMain.java

```java
package iniPackage;

public class ATMMain {
  // main method creates and runs the ATM
  public static void main(String[] args) {
    ATM theATM = new ATM();
    theATM.run();
  }
}
```

## Output Program :

```
Welcome!

Please enter your account number: 12345

Enter your PIN: 54321

Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

Enter a choice: 1

Balance Information:
 - Available balance: $1,000.00
 - Total balance:     $1,200.00

Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

Enter a choice: 2

Withdrawal Menu:
1 - $20
2 - $40
3 - $60
4 - $100
5 - $200
6 - Cancel transaction

Choose a withdrawal amount: 5

Your cash has been dispensed. Please take your cash now.

Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit
```

```
Enter a choice: 1

Balance Information:
 - Available balance: $800.00
 - Total balance:     $1,000.00

Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

Enter a choice: 2

Withdrawal Menu:
1 - $20
2 - $40
3 - $60
4 - $100
5 - $200
6 - Cancel transaction

Choose a withdrawal amount: 6

Canceling transaction...

Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

Enter a choice: 3

Please enter a deposit amount in CENTS (or 0 to cancel): 2000

Please insert a deposit envelope containing $20.00.

Your envelope has been received.
NOTE: The money just deposited will not be available until we verify the amount
of any enclosed cash and your checks clear.

Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

Enter a choice: 1

Balance Information:
 - Available balance: $800.00
 - Total balance:     $1,020.00

Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit
```

```
Enter a choice: 4

Exiting the system...

Thank you! Goodbye!
```

## Screenshoot Hasil Program :