```
This program allows a bank customer to make a payment on their loan.
It asks for the user to input their IRD number, select a loan, select
from which account to pay, and how much to pay.

Instructions:
Simply run the program using ‘java LoanPayer’ and follow the instructions.
An example of input that will create a successful transaction:

Please enter your IRD number:

123456781

Select which loan you wish to make a payment on:

Press '1' for home loan 354 (Amount owing: $10000.00)

1

You have selected loan '354'
Select which account to pay from:
Press '1' for savings account 23346 (Current balance: $3098.12)
Press '2' for investment account 23347 (Current balance: $10200.00)

2

You have selected account '58588'
Select payment amount (Do no include commas)

4000.00

Transaction complete
Final balance for account 58588: $6200.00
Final amount owing for loan 003: $6000.00
```

SOURCE CODE:

```
/**
 * File LoanPayer.java
 * @author Jeremy Haakma
 **/
import java.io.*;
import java.util.*;
import java.sql.*;
import oracle.jdbc.*;
import oracle.jdbc.pool.*;
import java.lang.*;

/**
 * This program allows a bank customer to make a payment on their loan.
 * It asks for the user to input their IRD number, select a loan, select
 * from which account to pay, and how much to pay.
 *
 * An example of input that will create a successful transaction:
 *
```

```
 * "Please enter your IRD number"
 *
 * 123456781
 *
 * "Select which loan you wish to make a payment on:"
 *     "Press '1' for home loan 354 (Amount owing: $10000.00)"
 *
 * 1
 *
 * "You have selected loan '354'"
 * "Select which account to pay from:"
 *     "Press '1' for savings account 23346 (Current balance: $3098.12)"
 *     "Press '2' for investment account 23347 (Current balance: $10200.00)"
 *
 * 2
 *
 * "You have selected account '58588'"
 * "Select payment amount (Do no include commas)
 *
 * 4000.00
 *
 * "Transaction complete"
 * "Final balance for account 58588: $6200.00"
 * "Final amount owing for loan 003: $6000.00"
 **/
public class LoanPayer{
    /** 9 digit IRD number entered by user **/
    static int IRDNumber;
    /** 2D ArrayList of loan information **/

    static ArrayList<String> loanNumList;
    static ArrayList<String> loanTypeList;
    static ArrayList<String> loanAmountList;
    /** 2S ArrayList of account information **/

    static ArrayList<String> accountNumList;
    static ArrayList<String> accountTypeList;;
    static ArrayList<String> accountAmountList;
    /** an int between 1 and n, where n is the number of loans available **/
    static int loanSelected;
    /** an int between 1 and n, where n is the number of accounts available **/
    static int accountSelected;
    /** A double representing the size of the payment to be made **/
    static double paymentAmount;



    /**
     * Main method: welcomes user, creates a new instance of the LoanPayer class
     * and calls the go() method.
     **/
    public static void main(String[] args){
        System.out.println("Welcome to the loan payment system.\n"+
                           "Press CTRL+z at any time to exit\n");
        new LoanPayer().go();
    }//end main()
```

```
/**
 * This method prompts the user for a valid IRD number. When one is
 * entered, the rest of the program is run.
 **/
private void go (){
    while(true){//Loops until valid input (int) is entered
        try{
            while(true){
                System.out.println( "Please enter your IRD number: ");
                IRDNumber = Integer.parseInt(new Scanner(System.in).nextLine());
                if(String.valueOf(IRDNumber).length() == 9){
                    break;
                } else{
                    System.out.println("Number must be 9 digits long");
                }
            }
            //Valid input entered: Prompt user for transactin info
            setTables();
            selectLoan();
            selectAccount();
            selectPayment();
            //Make the transaction
            transaction();
            System.out.println("Transaction complete");
            //Access new data values after transaction
            setTables();
            //Print final balance
            System.out.printf("Final balance for account %s: $%.2f\n",
                        accountNumList.get(accountSelected),
                        Double.parseDouble(accountAmountList.get(accountSelected)));
            System.out.printf("Final amount owing for loan %s: $%.2f\n",
                        loanNumList.get(loanSelected),
                        Double.parseDouble(loanAmountList.get(loanSelected)));
        break;
        }catch(NumberFormatException e){
            System.out.println("Invalid input.");
        }
    }
}//end go()


/**
 * Sends two queries to the Oracle server using the getValues() method.
 * Loads the results from each query into the ArrayLists for loans and accounts
 **/
private void setTables(){
    ArrayList<ArrayList<String>> loansInfo =
        getValues("SELECT * FROM loan "+
                  "WHERE loanno IN "+
                  "(SELECT loanno FROM loan_of "
                  +"WHERE ird = "+IRDNumber+")");

    ArrayList<ArrayList<String>> accountsInfo =
        getValues("SELECT * FROM account "+
                  "WHERE acctno IN "+
                  "(SELECT acctno FROM account_of "
                  +"WHERE ird = "+IRDNumber+")");
```

```
        loanNumList = loansInfo.get(0);
        loanTypeList = loansInfo.get(1);
        loanAmountList = loansInfo.get(2);
        //Access account info

        accountNumList = accountsInfo.get(0);
        accountTypeList = accountsInfo.get(1);
        accountAmountList = accountsInfo.get(2);
    }


    /**
     * This method pulls table information out of an Oracle database
     * and puts it in a 2D ArrayList.
     * @param query SQL query to be executed
     * @return output 2D ArrayList of query results.
     **/
    private ArrayList<ArrayList<String>> getValues(String query){
        ArrayList<ArrayList<String>> output = new ArrayList<ArrayList<String>>();
        //ArrayLists store each column of data from the table
        ArrayList<String> column1 = new ArrayList<String>();
        ArrayList<String> column2 = new ArrayList<String>();
        ArrayList<String> column3 = new ArrayList<String>();

        //Read pass.dat
        UserPass login = new UserPass();
        String user = login.getUserName();
        String pass = login.getPassWord();
        String host = "silver";
        Connection con = null;

        try{
            // Register the driver and connect to Oracle
          DriverManager.registerDriver
            (new oracle.jdbc.driver.OracleDriver());
          String url = "jdbc:oracle:thin:@" + host + ":1527:cosc344";
          con = DriverManager.getConnection(url, user, pass);
            Statement stmt = con.createStatement();
            ResultSet result = stmt.executeQuery(query);
            int count = 1;
            while(result.next()){
                //We can assume that column 1 and 2 are Strings, and Column 3 is double
                //As we only call this method on loan and account tables
                column1.add(result.getString(1));
                column2.add(result.getString(2));
                column3.add(String.valueOf(result.getDouble(3)));
                count++;
            }
        } catch (SQLException e) {
          System.out.println(e.getMessage());
          System.exit(1);
      } finally {
          if (con != null) {
            try {
                con.close();
            } catch (SQLException e) {
                quit(e.getMessage());
            }
```

```
      }
    }
    output.add(column1);
    output.add(column2);
    output.add(column3);
    return output;
}//end getValues()



/**
 * Allows user to select a loan from an available list.
 * Retrieves loan information from the Oracle SQL server, then
 * displays this information to the user, prompting them to
 * select one to make a payment on.
 **/
private void selectLoan(){

    while(true){//Loop until valid response is made
        Scanner sc = new Scanner(System.in);
        int numPress = 0;
        //Select loans from customer based on IRDNumber
        //checks whether there are any loans associated with selected IRD number
        if (!loanNumList.isEmpty()){
            System.out.println("Select which loan you wish to make a payment on: ");
            do { //Iterate through each row of loan info
                System.out.printf("\tPress '%d' for %s loan %s (Amount owing: $%.2f)\n",
                                (numPress+1),
                                loanTypeList.get(numPress).trim(),
                                loanNumList.get(numPress).trim(),
                                Double.parseDouble(loanAmountList.get(numPress).trim()));
                numPress++;
            } while(numPress < loanAmountList.size());
        } else {//If there are no loans connected to this IRD number, exit
            System.out.println("Sorry, there are no loans associated "+
                               "with this IRD number");
            System.exit(0);
        }
        try{//Check for valid response
            loanSelected = sc.nextInt()-1;//Minus one to line up with arraylist
            if(loanSelected >= 0 && loanSelected < numPress){
                //CORRECT RESPONSE
                System.out.printf("\nYou have selected loan '%s'\n",
                                  loanNumList.get(loanSelected));
                break;
            }else {
                invalidResponse();
            }
        } catch (InputMismatchException e){
            invalidResponse();
        }
    }
}//end selectLoan()


 /**
  * Allows user to select an account from an available list.
  * Retrieves account information from the Oracle SQL server, then
```

```
      * displays this information to the user, prompting them to
      * select one to make a payment from.
      **/
    private void selectAccount(){

        Boolean validAccountSelected = false;
        while(!validAccountSelected){//Loop until valid response is made
            Scanner sc = new Scanner(System.in);
            int numPress = 0;

            //Select accounts from customer based on IRDNumber
            //checks whether there are any accounts associated with selected IRD number
            if (!accountNumList.isEmpty()){
                System.out.println("Select which account to pay from: ");
                do {//Iterate over and display rows of account info
                    System.out.printf("\tPress '%d' for %s account %s "+
                                      "(Current balance: $%.2f)\n",
                                      (numPress+1),
                                      accountTypeList.get(numPress).trim(),
                                      accountNumList.get(numPress).trim(),
                                      Double.parseDouble
                                      (accountAmountList.get(numPress).trim()));
                    numPress++;
                } while(numPress < accountAmountList.size());
            } else {
                System.out.println("Sorry, there are no accounts associated "+
                                   "with this IRD number");
                System.exit(0);
            }

            //Make sure accountSelected response is an int
            try{
                accountSelected = sc.nextInt()-1;//Minus one to line up with arraylist
                if(accountSelected >= 0 && accountSelected < numPress){
                    //CORRECT RESPONSE
                    validAccountSelected = true;
                    System.out.printf("You have selected account '%s'\n",
                                      accountNumList.get(accountSelected));
                }else {
                    invalidResponse();;
                }
            } catch (InputMismatchException e){
                invalidResponse();
            }

        }
    }//end selectAccount()



    /**
     * Prompts user to indicate how large a payment they want to make.
     * When a valid amount is entered (less than loan amount, less than account
     * balance, greater than zero), the transaction() method is called to make
     * the payment.
     **/
    private void selectPayment(){
```

```
        while(true){//Loop until valid amount is entered
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter payment amount (Do not include commas)");
            try{//Check if input is a double
                paymentAmount = sc.nextDouble();
                if(paymentAmount > Double.parseDouble(loanAmountList.get(loanSelected)) ){
                    System.out.println("Invalid number. Your total loan amount is "+
                                       loanAmountList.get(loanSelected));

                } else if (paymentAmount >
                           Double.parseDouble(accountAmountList.get(accountSelected))){
                    System.out.println("Insufficient funds. Please enter another amount");

                } else if (paymentAmount <= 0){
                    System.out.println("Please enter a positive value");

                }else if (String.valueOf(paymentAmount).substring
                          (String.valueOf(paymentAmount).indexOf(".")).length() > 3){
                    System.out.println("Too many decimal places");

                } else {
                    //Valid amount entered: Break out of while loop
                    break;
                }
            } catch (InputMismatchException e){
                System.out.println("Incorrect response. "+
                                   "Please try again(or CTRL+z to exit)");
            }
        }
    }//end selectPayment()



    /**
     * Connects to Oracle server and performs a transaction.
     * Updates loan and account balances by subtracting the
     * payment value set by the user.
     * This method does not implement concurrency control.
     */
    private void transaction(){
        double amount = paymentAmount;
        String acctno = accountNumList.get(accountSelected);
        String loanno = loanNumList.get(loanSelected);
        //Read pass.dat
        UserPass login = new UserPass();
        String user = login.getUserName();
        String pass = login.getPassWord();
        String host = "silver";

        Connection con = null;
        try{
            // Register the driver and connect to Oracle
          DriverManager.registerDriver
            (new oracle.jdbc.driver.OracleDriver());
          String url = "jdbc:oracle:thin:@" + host + ":1527:cosc344";
          con = DriverManager.getConnection(url, user, pass);
            //Subtract payment amount from loan amount and account balance
```

```
            PreparedStatement pstmt1 = con.prepareStatement("UPDATE loan "+
                                                            "SET amount = amount - "+amount+
                                                            " WHERE loanno = "+loanno);
            PreparedStatement pstmt2 = con.prepareStatement("UPDATE account "+
                                                            "SET balance = balance - "+amount+
                                                            " WHERE acctno = "+acctno);
            pstmt1.executeUpdate();
            pstmt1.close();
            pstmt2.executeUpdate();
            pstmt2.close();

        } catch (SQLException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        } finally {
            if (con != null) {
                try {
                    con.close();
                } catch (SQLException e) {
                    quit(e.getMessage());
                }
            }
        }
    }//end transaction()


    /** Prints error when invalid response is typed **/
    private void invalidResponse(){
        System.out.println("\nInvalid response. "+
                           "Please try again(or CTRL+z to exit)");
    }

    /** Used to output an error message and exit
     * @param message error message to send
     **/
    private void quit(String message) {
        System.err.println(message);
        System.exit(1);
    }
}
```

```
/*
  File:  UserPass.java
  July 2002
*/

import java.io.*;
import java.util.*;
import java.lang.*;

/**
 * Reads a username and password from a file called pass.dat.
 *
 * @author Paul Werstein
 */

public class UserPass {

    private String password;
    private String username;

    // Constructor - Also reads the username and password
    //               from the file.
    public UserPass () {
      String line = null;
      String passwordFile = "pass.dat";
      try {
          BufferedReader inFile =
            new BufferedReader(new FileReader(passwordFile));
          // Read the username from the file and store it.
          if ((line = inFile.readLine()) == null) {
            quit(passwordFile + " is empty");
          }
          StringTokenizer tok = new StringTokenizer(line);
          if (tok.countTokens() != 1) {
            quit("Username line has an error");
          }
          username = tok.nextToken();
          // Read the password from the file and store it.
          if ((line = inFile.readLine()) == null) {
            quit(passwordFile + " has a bad format");
          }
          tok = new StringTokenizer(line);
          if (tok.countTokens() != 1) {
            quit("Password line has an error");
          }
          password = tok.nextToken();
      } catch (FileNotFoundException e) {
          quit("The file, " + passwordFile + ", was not found.");
      } catch (IOException e) {
          quit("An error occured trying to read " + passwordFile);
      }
    }

    // Returns the password
```

```
    public String getPassWord() {
      return password;
    }

    // Returns the username

    public String getUserName() {
      return username;
    }

    // Used for printing reasons for exceptions or errors.

    private void quit(String message) {
      System.err.println(message);
      System.exit(1);
    }

} // end class UserPass
```