```
* CSE222 HW01 101044044
* @author Samet Sait Talayhan
package bankprogram;
import java.util.InputMismatchException;
import java.util.Scanner;
public class BPConsoleUI implements BPUserInterface {
     * A reference to the BankProgram object to be processed. Globally
     * available to the command-processing methods.
   private BankProgram theProgram = null;
     * Scanner to read from input console.
   private Scanner scIn = null;
   // Constructor
   /**
    * Default constructor.
   public BPConsoleUI() {
        scIn = new Scanner(System.in);
    // Methods
    * Method to display the command choices and process user commands. pre: The
      program exists and has been loaded with data. post: The program is
      updated based on user commands.
      @param theBankProgram A reference to the BankProgram to be
      processed
   @Override
   public void processCommands(BankProgram theBankProgram) {
        String[] commands = {
            "Add/Change Account",
            "Add/Change Worker",
            "Look Up Account",
            "Remove Account",
            "Save Program",
            "Exit"};
        System.out.println("
                                     Bank Program ");
        theProgram = theBankProgram;
        int choice;
        do {
                                                             ");
            System.out.println("
            for (int i = 0; i < commands.length; ++i) {
                System.out.println("Select " + i + ":
                        + commands[i]);
            try {
                choice = scIn.nextInt(); // Read the next choice.
                scIn.nextLine(); // Skip trailing newline.
                switch (choice) {
                    case 0:
                                                                           ");
                        System.out.println("
                        doAddChangeAccount();
                        break:
                    case 1:
```

```
");
                    System.out.println("
                    doAddChangeWorker();
                    break;
                case 2:
                    System.out.println("_
                                                                      ");
                    doLookupAccount();
                    break;
                case 3:
                    System.out.println("
                                                                     ");
                    doRemoveAccount();
                    break:
                case 4:
                case 5:
                    doSave();
                    break;
                default:
                    System.out.println("*** Invalid choice "
                            + choice
                            + " - try again!");
        } catch (InputMismatchException ex) {
            System.out.println("*** Incorrect data entry - "
                    + "enter an integer between 0 and "
                    + (commands.length - 1));
            scIn.nextLine(); // Discard bad input.
            choice = -1;
    } while (choice != commands.length - 1);
    System.exit(0);
}
 * Method to add or change an account. pre: The program exists and has been
  loaded with data. post: A new name is added, or the value for the name is
 * changed, modified is set to true.
private void doAddChangeAccount() {
    // Request the name.
    System.out.println("Enter name");
    String newName = scIn.nextLine();
    if (newName.equals("")) {
        return:
    }
    // Request the account number.
    System.out.println("Enter Account number");
    String newAccountNumber = scIn.nextLine();
    if (newAccountNumber.equals("")) {
        return;
    // Request the account balance.
    System.out.println("Enter Account balance");
    String newAccountBalance = scIn.nextLine();
    if (newAccountBalance.equals("")) {
        return;
    // Insert/change name-account numbe-account balance.
    String oldAccountNumber =
            theProgram.addOrChangeAccount(newName, newAccountNumber,
                                          newAccountBalance);
    String message;
    if (oldAccountNumber == null) { // New entry.
        message = newName + " was added to the program"
                + "\nNew Account number: " + newAccountNumber
                + "\nNew Balance: " + newAccountBalance +"$";
    } else { // Changed entry.
       message = "AccountNumber for " + newName + " was changed"
                + "\nOld Account number: " + oldAccountNumber
```

```
+ "\nNew number: " + newAccountNumber;
    // Display confirmation message.
    System.out.println(message);
}
/**
 * Method to add or change a worker. pre: The program exists and has been
 st loaded with data. post: A new worker is added, or the value for the worker
 * name is changed, modified is set to true.
 */
private void doAddChangeWorker() {
    // Request the name.
    System.out.println("Enter Bank Worker name");
    String newName = scIn.nextLine();
    if (newName.equals("")) {
        return;
    // Insert/change name.
    String oldWorkerName =
            theProgram.addOrChangeWorker(newName);
    String message;
    if (oldWorkerName == null) { // New entry.
        message = newName + " was added to the program\n"
                  +"Welcome to new office!";
    } else { // Changed entry.
        message = "" + newName + " was changed"
                  + oldWorkerName;
    }
    // Display confirmation message.
    System.out.println(message);
}
 * Method to look up a account. pre: The program has been loaded with data.
 * post: No changes made to the program data base.
private void doLookupAccount() {
    // Request the account number.
    System.out.println("Enter Account number:");
    String theAccountNumber = scIn.nextLine();
    if (theAccountNumber.equals("")) {
        return; // Dialog was cancelled.
    // Look up the account.
    String theAccountBalance = theProgram.lookupAccount(theAccountNumber);
    String message;
    if (theAccountBalance != null) { // Account was found.
        message = "The account for " + theAccountNumber + " has money $"
                  + theAccountBalance;
    } else { // Account was not found.
        message = theAccountNumber + " is not listed in the program";
    // Display the result.
    System.out.println(message);
}
* Method to look up a worker. pre: The program has been loaded with data.
 * post: No changes made to the program data base.
private void doLookupWorker() {
    // Request the bank worker name.
    System.out.println("Enter Bank Worker name:");
    String theBankWorkerName = scIn.nextLine();
    if (theBankWorkerName.equals("")) {
        return; // Dialog was cancelled.
```

```
}
    // Look up the bank worker.
    String control = theProgram.lookupBankWorker(theBankWorkerName);
    String message;
    if (control != null) { // Account was found.
        message = "" + theBankWorkerName + " still works here!";
    } else { // Account was not found.
        message = theBankWorkerName + " does not work here!";
    }
    // Display the result.
    System.out.println(message);
}
 * Method to remove a account Pre: The program has been loaded with data.
 * Post: The requested account number is removed, modifed is set true
*/
private void doRemoveAccount() {
           // Request the account number.
    System.out.println("Enter account number:");
    String theAccountNumber = scIn.nextLine();
    if (theAccountNumber.equals("")) {
        return;
    // Remove the account
    if(theProgram.removeAccount(theAccountNumber) != null)
        // Display confirmaion message
        System.out.println(theAccountNumber + " has been removed from"
                + " the program!");
    }
    else
        System.out.println(theAccountNumber + " not found!");
}
 * Method to save the program to the data file. pre: The program has
 * been loaded with data. post: The current contents of the program have
 * been saved to the data file.
private void doSave() {
    theProgram.save();
```

}