Project 11-1: Work with customer and employee data

Console

Welcome to the Person Tester application

Create customer or employee? (c/e): c

Enter first name: Frank
Enter last name: Jones
Enter email address: frank44@hotmail.com
Customer number: M10293

You entered:
Name: Frank Jones
Email: frank44@hotmail.com
Customer number: M10293

Continue? (y/n): y

Create customer or employee? (c/e): e

Enter first name: Anne
Enter last name: Prince
Enter email address: anne@murach.com
Social security number: 111-11-1111

You entered:
Name: Anne Prince
Email: anne@murach.com
Social security number: 111-11-1111

Continue? (y/n): n

Operation

• The application prompts the user to enter a customer or an employee.

• If the user selects customer, the application asks for name, email, and customer number.

• If the user selects employee, the application asks for name, email, and social security number.

• When the user finishes entering data for a customer or employee, the application displays the data that the user entered.
Project 11-1: Work with customer and employee data (cont.)

Specifications

- Create an abstract Person class that stores first name, last name, and email address. This class should provide a no-argument constructor, get and set methods for each instance variable, and it should override the toString method so it returns the first name, last name, and email fields in this format:

  Name: Frank Jones
  Email: frank44@hotmail.com

  In addition, it should contain an abstract method named getDisplayText that returns a string.

- Create a class named Customer that inherits the Person class. This class should store a customer number, it should provide get and set methods for the customer number, it should provide a no-argument constructor, and it should provide an implementation of the getDisplayText method. The getDisplayText method should return a string that consists of the string returned by the toString method of the Person class appended with the Customer number like this:

  Name: Frank Jones
  Email: frank44@hotmail.com
  Customer number: M10293

- Create a class named Employee that inherits the Person class. This class should store a social security number, it should provide get and set methods for the social security number, it should provide a no-argument constructor, and it should provide an implementation of the getDisplayText method. The getDisplayText method should return a string that consists of the string returned by the toString method of the Person class appended with the Employees social security number like this:

  Name: Anne Prince
  Email: anne@murach.com
  Social security number: 111-11-1111

- Create a class named PersonApp that prompts the user as shown in the console output. This class should create the necessary Customer and Employee objects from the data entered by the user, and it should use these objects to display the data to the user. To print the data for an object to the console, this application should use a static method named print that accepts a Person object.

- Use the Console class from chapter 8 or a variation of it to get entries from the user.
Project 12-1: Calculate a monthly balance

Console

Welcome to the Account Calculator

Starting Balance
Checking: $1,000.00

Enter the transactions for the month

Withdrawal or deposit? (w/d): w
Amount: 500

Continue? (y/n): y

Withdrawal or deposit? (w/d): d
Amount: 200

Continue? (y/n): n

Monthly Fees
Checking fee: $1.00

Final Balance
Checking: $699.00

Operation

• The application begins by displaying the starting balance for a checking account.
• The application prompts the user to enter the amount for a withdrawal or deposit.
• When the user finishes entering deposits and withdrawals, the application displays the fees for the month followed by the final balances for the month.
Project 12-1: Calculate a monthly balance (cont.)

Specifications

• Create interfaces named Depositable, Withdrawable, and Balanceable that specify the methods that can be used to work with accounts. The Depositable interface should include this method:
  ```java
  void deposit(double amount)
  ```
The Withdrawable interface should include this method:
  ```java
  void withdraw(double amount)
  ```
And the Balanceable interface should include these methods:
  ```java
  double getBalance()
  void setBalance(double amount)
  ```

• Create a class named Account that implements all three of these interfaces. In addition, it should supply a method like the following method that returns a balance that has been formatted as currency:
  ```java
  String getBalanceFormatted()
  ```

• Create a class named CheckingAccount that inherits the Account class. This class should include an instance variable for the monthly fee and these methods:
  ```java
  void subtractMonthlyFee()
  void setMonthlyFee(double monthlyFee)
  double getMonthlyFee()
  String getMonthlyFeeFormatted()
  ```
  By default, the monthly fee for a checking account should be $1.

• Create a class named Transactions that contains the following static methods for depositing and withdrawing funds from either type of account:
  ```java
  public static void deposit(Depositable account, double amount) {
      account.deposit(amount);
  }
  ```
  ```java
  public static void withdraw(Withdrawable account, double amount) {
      account.withdraw(amount);
  }
  ```

• Create a class named AccountApp that prompts the user for a transaction, posts the transaction, and displays the information shown in the console output. Create the necessary objects for each transaction, and post the transaction using the appropriate method of the Transactions class.

• Use the Console class presented in chapter 8 or a variation of it to get entries from the user.

• This application should not allow the user to withdraw more than the current account balance.

• This application should not allow the user to deposit more than $10,000 per transaction.