Project Details

In this project, you will write a program that allows a user to repeatedly create and manage express card accounts.

Each express card is like a debit card, except that it can only be used to purchase meals. A new express card account starts at a balance of $0. The user can

1. deposit money to the express card, increasing the balance (by depositing more money),
2. purchase meals at a fixed rate, thus decreasing the balance, and
3. have meals (each swipe of the card reduces the remaining number of meals by one).

In this project, we will only implement one type of express accounts: student express accounts. A student express account have fields `accountNumber`, `accountBalance`, `numberOfMeals`, `pricePerMeal`, `baseAmtForBonus`, `rewardLevel` and `rewardAmt`.

- The field `baseAmtForBonus` is used to record the bonus cashback for an account when making a deposit.
- There is a base amount one-time deposit (`baseAmtForBonus`) to receive bonus which is the minimum amount of money for each deposit. For this project, it is set to $500.0. For example, if the `baseAmtForBonus` is $500 and deposit $499 will not receive any bonus.
- For every deposit that is greater than the `baseAmtForBonus`, for each `rewardLevel`, a student account receives `rewardAmt` amount of money. For example, if `rewardLevel` is set to 200.0 and `rewardAmt` is set to 2.0, then a deposit of $500 to a $0 balance student express account will result $4.0 bonus and a new balance of $504.0. However, a deposit of $499 to a $100 balance student express account will not obtain any bonus and final balance is $599.0.

  The variable `rewardLevel` should be set to $200.0 and `rewardAmt` to $2.0. While we won't change the value of them in this project, they must be variables.
- The `pricePerMeal` is $10.0.

Implementation Hints

- Start by writing the `ExpressAccount` class with only the `accountBalance` field, a `toString()` method that outputs the balance, and a `main()` method that instantiates a new instance of the `ExpressAccount` class and outputs the result of calling the `toString()` method. Compile and test to make sure that this simple version works.
- Gradually build up the functionality of the `ExpressAccount` class, adding each field and method in turn, testing thoroughly that your class works at each step. Do this very gradually, breaking the development of the `ExpressAccount` class into small steps. It is often a handy trick to write a `main()` method for every class that tests the class thoroughly. This is very useful for diagnostic purposes, and will be essential when we start developing larger programs.
- To accept input from the console, use `java.util.Scanner` to directly accept input from `System.in`.

- Notice that you have two levels of menu: main menu and sub-menus. The main menu looks as follows:

```
MAIN MENU
1.) Create a new account
2.) Log into an existing account
3.) Exit the banking system
```

The sub-menu looks as follows:

```
EXPRESS ACCOUNT #0, BALANCE: $0.0, NUMBER OF MEALS: 0
1.) Make a deposit
2.) Purchase meals
3.) Have meal
4.) Log out
```

For each option in the main menu, there is a sub-menu for this option. However, some sub-menus share the same functionalities. For example, after you create an account, the display is the same as after you choose to log into an existing account. Instead of writing same code several times, you should try to make your program as modularly as possible by writing small methods. One good practice would be separating the display the menu from the logic of making options on the menu by passing parameters and getting return values wisely. With this in mind, you will notice that we have the following functionalities in the menu part:

  - display the main menu
  - create a new account of type `ExpressAccount`
  - store an account into the list
  - retrieve an account information from the list
  - display a user menu (for making a deposit, purchasing meals, etc)

  For each of these functionalities, it is recommended to have a method implementing it.

- Be sure to do error checking on the inputs. Negative deposit and number of meals to purchase are not allowed.
- Since we're working with money, all decimals need to be rounded off to two decimal places.
- After you are done with the implementation, make sure you clean up your code. Remove unnecessary testing code and comments.
- All source files should include a description header, be properly indented and commented.

Code Skeleton

Your final java source code should include the following classes:

- `ExpAcct.java` : This is where your `main` method will be. It contains the whole logic of the application. That is, the main menu and the sub-menu are implemented in this class using methods. Moreover, a list of accounts, implemented in ArrayList, is maintained in this class.
- `ExpressAccount.java` : This is the class that contains the account properties and behaviors of the classes.

Part of the code skeleton is shown as follows:

```
public class ExpAcct {
  ......
  public static void main(String[] args) {
    ......
  }
}

public class ExpressAccount {
  ......
  public ExpressAccount(int accNumber) {
    ......
  }

  public int getAccountNumber() {
    ......
  }

  public double getAccountBalance() {
    ......
  }

  public double getBaseAmtForBonus() {
    ......
  }

  public double getPricePerMeal() {
    ......
  }

  public int getNumOfMeals() {
    ......
  }

  public String toString() {
    ......
  }

  ......
}
```

Example running trace:

Your program **must** have the same interface as shown below.

```
Welcome to the Express Account Company

MAIN MENU
1.) Create a new account
2.) Log into an existing account
3.) Exit the banking system
Please enter your selection: 1

EXPRESS ACCOUNT #0, BALANCE: $0.0, NUMBER OF MEALS: 0
1.) Make a deposit
2.) Purchase meals
3.) Have meal
4.) Log out
Please enter your selection: 1
Enter deposit amount: 600
Deposit $600.0 New balance $606.0

EXPRESS ACCOUNT #0, BALANCE: $606.0, NUMBER OF MEALS: 0
1.) Make a deposit
2.) Purchase meals
3.) Have meal
4.) Log out
Please enter your selection: 2
Enter the number of meals you want to purchase: 1
Purchased 1 meals with $10.0 per meal New balance $596.0

EXPRESS ACCOUNT #0, BALANCE: $596.0, NUMBER OF MEALS: 1
1.) Make a deposit
2.) Purchase meals
3.) Have meal
4.) Log out
Please enter your selection: 3

EXPRESS ACCOUNT #0, BALANCE: $596.0, NUMBER OF MEALS: 0
1.) Make a deposit
2.) Purchase meals
3.) Have meal
4.) Log out
Please enter your selection: 3
No meals left on your account. Please purchase meals first.

EXPRESS ACCOUNT #0, BALANCE: $596.0, NUMBER OF MEALS: 0
1.) Make a deposit
2.) Purchase meals
3.) Have meal
4.) Log out
```

```
Please enter your selection: 1
Enter deposit amount: 0
The deposit must be a positive amount.

EXPRESS ACCOUNT #0, BALANCE: $596.0, NUMBER OF MEALS: 0
1.) Make a deposit
2.) Purchase meals
3.) Have meal
4.) Log out
Please enter your selection: 2
Enter the number of meals you want to purchase: 60
Not enough balance for 60 meals
Purchased 59 meals, New balance $6.0

EXPRESS ACCOUNT #0, BALANCE: $6.0, NUMBER OF MEALS: 59
1.) Make a deposit
2.) Purchase meals
3.) Have meal
4.) Log out
Please enter your selection: 4
Goodbye!

MAIN MENU
1.) Create a new account
2.) Log into an existing account
3.) Exit the system
Please enter your selection: 2

Enter account number: 0
Welcome back Student Express account #0, balance: $6.0, number of meals: 59

EXPRESS ACCOUNT #0, BALANCE: $6.0, NUMBER OF MEALS: 59
1.) Make a deposit
2.) Purchase meals
3.) Have meal
4.) Log out
Please enter your selection: 4
Goodbye!

MAIN MENU
1.) Create a new account
2.) Log into an existing account
3.) Exit the system
Please enter your selection: 3
Exiting the system
```

What to turn in:

JAR your *.java files into a file called Project4.jar. Upload the jar file to Canvas under category Project4.