Project Details

This project is designed for you to practice inheritance and abstract classes. 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).

There are two types of express card accounts: a student express account or a faculty express account. All express account have fields `accountNumber`, `accountBalance`, `numberOfMeals`, `pricePerMeal`, `baseAmtForBonus` and `accountTypeName`.

- 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 example, if the `baseAmtForBonus` for a student is $500 and deposit $499 for a student account will not receive any bonus. The criteria of receiving bonus for these two types of express accounts are different:
  - The `baseAmtForBonus` is set to $500.0 for a student express account.
  - The `baseAmtForBonus` is set to $0.0 for a faculty express account in the corresponding constructor.
- A student express account has extra two variables for bonus: `rewardLevel` and `rewardAmt`. 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.
- A faculty express account has only one extra variable for bonus: `rewardPct`, the percentage of deposit that will become the bonus. In this project, it is set to 0.01. For example, since the `baseAmtForBonus` is $0.0 for a faculty, a $50 deposit to a $0 balance faculty express account will result $0.5 bonus and a new balance of $50.5.
- The `pricePerMeal` is $8.0 for a faculty account and $10.0 for a student.

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, you can use `java.util.Scanner` to directly accept input from `System.in` or use `java.io.BufferedReader` to read text from a character input stream, buffering characters so as to provide for the efficient reading. Check Java Doc according to your JDK version for the usage of these classes.

- Notice that you have two levels of menu: main menu and sub-menus. 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 a array list
  - retrieve an account information from the array 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.

  You should use `ArrayList` to store the list of express accounts. A tutorial for using `ArrayList` can be found here: Java ArrayList.

- To gradually build up the whole menu, you may first use control flow statements to make a user menu with only one general express account type. This means that the menu is pretty much the same as the final version, except that there is only one type of express accounts.

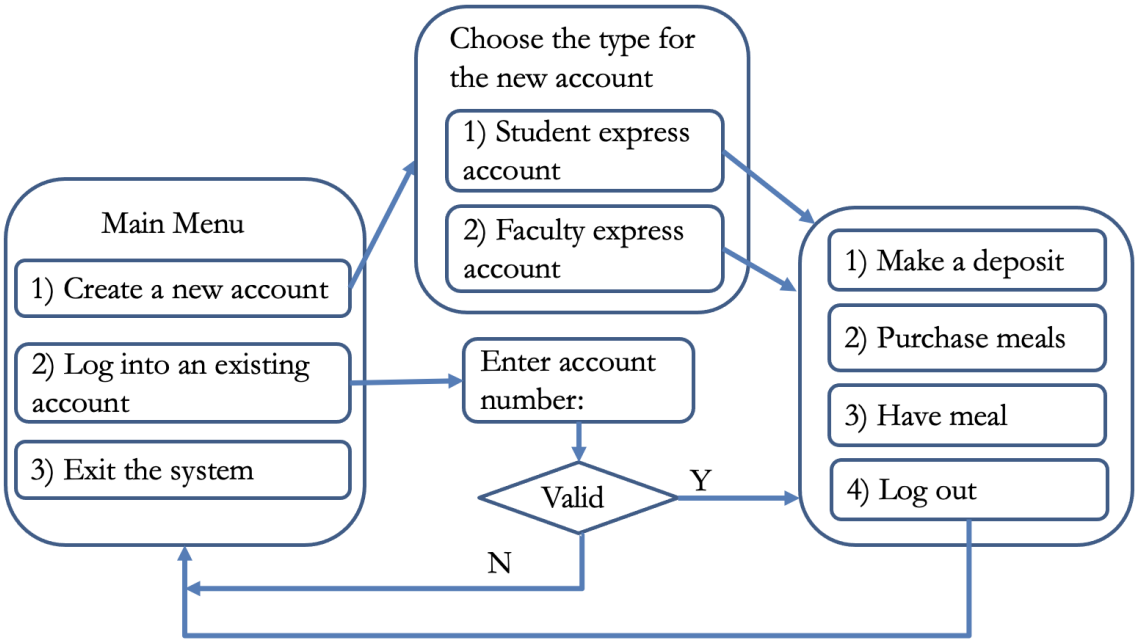
- Error handling: A user may enter invalid input. For example, in the main menu, a user may enter 4 which is invalid. Your implementation should handle such cases. For example,

  - If an invalid menu selection is entered, the system should print out the message
    "Invalid selection. Please try again."
    and then print out the same menu.
  - If a wrong account number is entered, the system should print out
    "Invalid account number (must be between ## and ##)"
    where ## denotes the existing first and the last account numbers, respectively.
    Then the system should ask the user to enter account number again.
  - When a user wants to purchase meals without enough balance or have a meal without any meals left in the account, the system should prompt proper messages and then print out the same menu for the user to continue making selections.

Your error handling should be implemented such that your program will not create any run-time exception for any input and at any point of execution of your program.

- Since we're working with money, all decimals need to be rounded off to two decimal places.
- Implement classes for the student and faculty express accounts that extend the `ExpressAccount` and implement the appropriate behavior for each type of account. These subclasses should reuse as much as possible from the base `ExpressAccount` class. If functionality is shared between both account types, then it should be included in the `ExpressAccount` class (i.e., don't implement the same thing three times in the subclasses, implement it once in the `ExpressAccount` class). You may add constructors and protected setters to your `ExpressAccount` class; however, you should be able to reuse the methods `purchaseMeal` and `haveMeal` implemented in your `ExpressAccount` class.
- 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.

Menu relation



Code Skeleton

Your final java source code should include the following classes:

- `ExpAcctDriver.java` : This is where your `main` method will be. It contains the whole logic of the application.
- `ExpressAccount.java` : This is the super class that contains the common properties and behaviors of the classes `FacultyExpressAccount` and `StudentExpressAccount`.
- `FacultyExpressAccount.java`
- `StudentExpressAccount.java`

Part of the code skeleton is shown as follows:

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

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

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

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

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

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

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

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

  public abstract void deposit(double amount);
  ......
  //Other methods such as purchaseMeal(),  haveMeal(), or helper methods.
}
```

```
public class FacultyExpressAccount extends ExpressAccount {
  ......
}

public class StudentExpressAccount extends ExpressAccount {
  ......
}
```

What to turn in:

Turn in a zip file named `Firstname_Lastname_P1.zip` containing all your source code for the project. Include the Javadoc tag @author in each class source file.

Grading:

- Documentation and style: 2 points
  Make sure that your projects are commented properly. This includes file headers, method headers, and appropriate inline comments.
- Correctness: 8 points
  - All methods and variables in a class are correctly defined with appropriate modifiers.
  - The application logic is clean and clear.
  - Use `Scanner` class correctly.
  - Inheritance is correctly implemented.
  - The output of the application is exactly as it shows in the example run (see below).
  - The implementation should satisfy all the requirements described in the project.

Example Running Trace

Here's an example trace of the program running once the assignment is complete.

```
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

CHOOSE THE TYPE FOR THE NEW ACCOUNT
1.) Student express account
2.) Faculty express account
```

```
Please enter your selection: 1
Created new Student Express account #0, balance: $0.0, number of meals: 0

STUDENT 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
Received bonus of $6.0
Deposit $600.0 New balance $606.0

STUDENT 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

STUDENT 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

STUDENT 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.

STUDENT 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.
```

```
STUDENT 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

STUDENT 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: 1
Invalid account number (must be between 0 and 0)

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

STUDENT 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: 1

CHOOSE THE TYPE FOR THE NEW ACCOUNT
1.) Student express account
```

```
2.) Faculty express account
Please enter your selection: 2
Created new Faculty Express account #1, balance: $0.0, number of meals: 0

FACULTY EXPRESS ACCOUNT #1, 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: 80
Received bonus of $0.8
Deposit $80.0 New balance $80.8

FACULTY EXPRESS ACCOUNT #1, BALANCE: $80.8, 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: 3
Purchased 3 meals with $8.0 per meal New balance $56.8

FACULTY EXPRESS ACCOUNT #1, BALANCE: $56.8, NUMBER OF MEALS: 3
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
```