Review Lab

All Tasks given below must be submitted via Blackboard submission system by the deadline. Create a zip folder which contains all the **.java** files of the tasks and screen shots for the output.

1. Write a program to sum the following series:

$$\frac{1}{3}+\frac{3}{5}+\frac{5}{7}+\frac{7}{9}+\frac{9}{11}+\frac{11}{13}+\cdots+\frac{95}{97}+\frac{97}{99}$$

2. Design a class named **Account** that contains:

   - A private **int** data field named **id** for the account (default **0**).
   - A private **double** data field named **balance** for the account (default **0**).
   - A private **double** data field named **annualInterestRate** that stores the current interest rate (default **0**). Assume all accounts have the same interest rate.
   - A private **Date** data field named **dateCreated** that stores the date when the account was created.
   - A no-arg constructor that creates a default account.
   - A constructor that creates an account with the specified id and initial balance.
   - The accessor and mutator methods for **id**, **balance**, and **annualInterestRate**.
   - The accessor method for **dateCreated**.
   - A method named **getMonthlyInterestRate()** that returns the monthly interest rate.
   - A method named **getMonthlyInterest()** that returns the monthly interest.
   - A method named **withdraw** that withdraws a specified amount from the account.
   - A method named **deposit** that deposits a specified amount to the account.

   (*Hint*: Monthly interest is **balance * monthlyInterestRate**. **monthlyInterestRate** is **annualInterestRate / 12**.)

   Write a test program that creates an **Account** object with an account ID of 1122, a balance of $20,000, and an annual interest rate of 4.5%. Use the **withdraw** method to withdraw $2,500, use the **deposit** method to deposit $3,000, and print the balance, the monthly interest, and the date when this account was created.

3. Use the **Account** class created in Exercise 2 (above) to simulate an ATM machine. Create ten accounts in an array with id **0**, **1**, . . . , **9**, and initial balance $100. The system prompts the user to enter an id. If the id is entered incorrectly, ask the user to enter a correct id. Once an id is accepted, the main menu is displayed as shown in the sample run. You can enter a choice **1** for viewing the current balance, **2** for withdrawing money,

3 for depositing money, and **4** for exiting the main menu. Once you exit, the system will prompt for an id again. Thus, once the system starts, it will not stop.

Sample Run:

```
Enter an id: 4
Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 1
The balance is 100.0

Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 2
Enter an amount to withdraw: 3

Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 1
The balance is 97.0
```

4. In Programming Exercise 2, the **Account** class was defined to model a bank account. An account has the properties account number, balance, annual interest rate, and date created, and methods to deposit and withdraw funds. Create two subclasses for **checking** and **saving** accounts. A checking account has an overdraft limit, but a savings account cannot be overdrawn.
   Write a test program that creates objects of **Account**, **SavingsAccount**, and **CheckingAccount** and invokes their **toString()** methods.

5. Design a new **Account** class as follows:

- Add a new data field **name** of the **String** type to store the name of the customer.
- Add a new constructor that constructs an account with the specified name, id, and balance.

- Add a new data field named **transactions** whose type is **ArrayList** that stores the transaction for the accounts. Each transaction is an instance of the **Transaction** class. The **Transaction** class is defined as shown in Figure below.
- Modify the **withdraw** and **deposit** methods to add a transaction to the **transactions** array list.
- All other properties and methods are the same as in Programming Exercise 2.

Write a test program that creates an **Account** with annual interest rate **1.5%**, balance **1000**, id **1122**, and name **George**. Deposit $30, $40, and $50 to the account and withdraw $5, $4, and $2 from the account. Print an account summary that shows account holder name, interest rate, balance, and all transactions.

Transaction
-date: java.util.Date **-type: char** **-amount: double** **-balance: double** **-description: String**
+Transaction(type: char, amount: double, balance: double, description: String)

The date of this transaction.
The type of transaction, such as 'W' for withdrawal, 'D' for deposit.
The amount of the transaction.
The new balance after transaction.
The description of this transaction.

6. Write a method that removes the duplicate elements from an array list of integers using the following header:

```
public static void removeDuplicate(ArrayList<Integer> list)
```

Write a test program that prompts the user to enter 10 integers to a list and displays the distinct integers separated by exactly one space. Here is a sample run:

```
Enter 10 Integers: 34 5 3 5 6 4 33 2 2 4
The district integers are: 34 5 3 6 4 33 2
```

7. Write the following method that shuffles an **ArrayList** of numbers:

   public static void shuffle(ArrayList<Number> list)

8. Write the following method that shuffles an **ArrayList** of numbers:

   public static <E> **void** shuffle(ArrayList<E> list)