CSE 215L: Programming Language II Lab
Faculty: Silvia Ahmed, Sec – 2, 16
Final Assignment – Spring 2020
Total Marks: 90, Submission Date: 04.06.2020

Task – 1 **10 points**

A positive integer is called a perfect number if it is equal to the sum of all of its positive divisors, excluding itself. For example, 6 is the first perfect number because 6 = 3 + 2 + 1. The next is 28 = 14 + 7 + 4 + 2 + 1. Write a java program to find perfect numbers less than 10,00,000.

Task – 2 **10 +10 + 20 = 40 points**

(*The* **Account** *class*) 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**).
■ A private **Date** data field named **dateCreated** that stores the date when the account was created.
■ A data field **name** of the **String** type to store the name of the customer.
■ A no-arg constructor that creates a default account.
■ A constructor that creates an account with the specified id and initial balance.
■ A constructor that constructs an account with the specified name, id, and balance.
■ A 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.

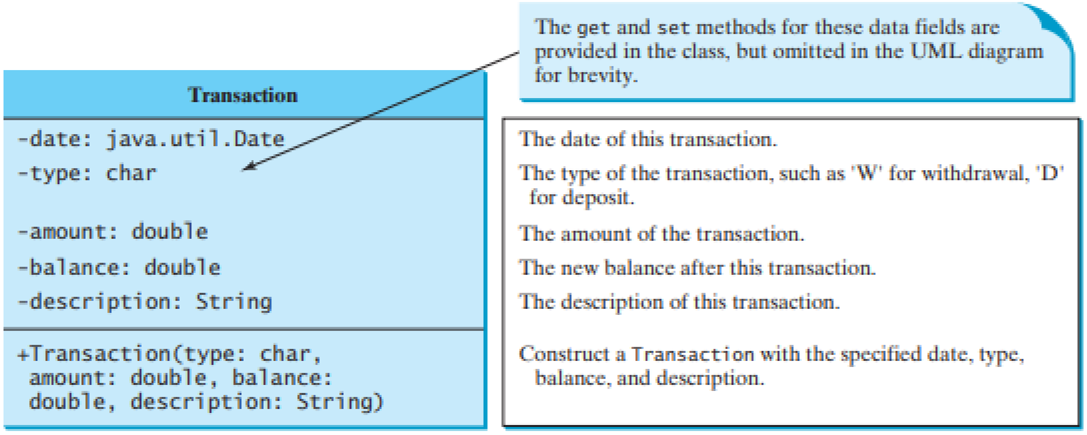
The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.
Transaction
-date: java.util.Date
-type: char
-amount: double
-balance: double
-description: String
+Transaction(type: char, amount: double, balance: double, description: String)
The date of this transaction.
The type of the transaction, such as 'W' for withdrawal, 'D' for deposit.
The amount of the transaction.
The new balance after this transaction.
The description of this transaction.
Construct a Transaction with the specified date, type, balance, and description.

Figure: The **Transaction** class describes a transaction for a bank account.

■ 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. (*Hint*: The method **getMonthlyInterest()** is to return monthly interest, not the interest rate. Monthly interest is **balance** *

monthlyInterestRate. **monthlyInterestRate** is **annualInterestRate / 12**. Note that **annualInterestRate** is a percentage, e.g.,like 4.5%. You need to divide it by 100.).

■ A method named **withdraw** that withdraws a specified amount from the account and adds a transaction to the **transactions** array list.
■ A method named **deposit** that deposits a specified amount to the account and adds a transaction to the **transactions** array list.

Write a test program that creates an **Account** array of size 10 with annual interest rate **1.5%**, balance **1000**. Assume all accounts have the same interest rate. Take user input for the other information of the array and perform withdraw and deposit for each account. Print an account summary for each of the accounts in a file that shows account holder name, interest rate, balance, and all transactions.

Task – 3 **10 +10 + 20 = 40 points**

Design the **Employee** class that extends **Person** which implements an interface called **Comparable**. Implement the **compareTo** method in the **Person** class to compare persons in alphabetical order of their last name. Implement the **compareTo** method to compare employees in alphabetical order of their last name. Sort the employees according to their salary. Add this methods:

- public static void **sort(Comparable[] list)**
- public static void **printList(Object[] object)**
- public static Comparable **max(Comparable[] list)**

Write a test class to test the **sort(Comparable[] list)**, **printList(Object[] object)**, and **max(Comparable[] list)** methods using an array of 10 employees. Throw **IllegalAccessException** if the user enters any wrong information for any data field.