Seneca College

February 02, 2018

Applied Arts & Technology
SCHOOL OF COMPUTER STUDIES

JAC444 **Due dates: February 6 and February 13, 2018**

Workshop 1

1. Design a class named **Location** for locating a maximal value and its location in a two-dimensional array. The class contains public data fields **row**, **column**, and **maxValue** that store the maximal value and its indices in a two-dimensional array with **row** and **column** as **int** types and **maxValue** as a **double** type.
   Write the following method that returns the location of the largest element in a two-dimensional array:

   public static Location locateLargest(**double**[][] a)

   The return value is an instance of **Location**. Write a test program that prompts the user to enter a two-dimensional array and displays the location of the largest element in the array. Here is a sample run:

```
Enter the number of rows and columns in the array:  3 4 Enter
Enter the array:
23.5 35 2 10 Enter
4.5 3 45 3.5 Enter
35 44 5.5 9.6 Enter
The location of the largest element is 45 at (1, 2)
```

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

Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 3
Enter an amount to deposit: 10


Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 1
The balance is 107.0

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

Enter an id:
```

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.