Objectives

The objective of today’s lab session is to build on the previous Account class to use some of the programming concepts that you have learned during last week’s lectures. By the end of this lab you should develop better understanding of how to:

- define and use static members (data fields and methods)
- use this keyword:
  - to reference hidden data fields
  - to invoke a constructor within a class
- to define and recognize immutable classes

Please note that you should bring your Account class with you to the lab as to be able to easily incorporate the modifications described below.

Exercise 1

Incorporate the modifications written in green below to the Account class that you have created in the lab last week.

Design a class named Account that contains:

a. A private int data field named id for the account (default 0)
b. A private double data field named balance for the account (default 0)
c. A private Date data field named dateCreated that stores the date when the account was created
d. A static private int data field named numberOfCreatedAccounts that tracks the number of created accounts.
e. A no-arg constructor that creates a default account. (Use this keyword to invoke the other constructor explicitly).
f. A constructor that creates an account with the specified id and initial balance. (Use this keyword to reference the data fields id and balance)
g. The accessor and mutator methods for id and balance. (in the setter methods, use this keyword to reference the data fields id and balance)
h. The accessor method for dateCreated
i. A static method named getNumberOfCreatedAccounts() that returns the number of created accounts.
j. A method named withdraw() that withdraws a specified amount from the account
k. A method named deposit() that deposits a specified amount to the account

1. Update the UML diagram for the Account class to include the new static members added to the class.
2. Implement the Account class
3. Is the Account class immutable? Explain your answer.
4. Use a test program to do the following:
   - Create account1 object using a default constructor and display its initial balance.
   - Create account2 object with specified account ID of 1122, and a balance of SR20,000 and display its initial balance. Use the withdraw() method to withdraw SR2,500, use the deposit() method to deposit SR3,000 to account2, and print the final balance.
   - Display the date when these two accounts were created.
   - Display the number of created accounts.
**Sample Run:**

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Balance of account1 is</td>
<td>0.0</td>
</tr>
<tr>
<td>Initial Balance of account2 is</td>
<td>20000.0</td>
</tr>
<tr>
<td>Now the balance of account2 after withdrawing is</td>
<td>17500.0</td>
</tr>
<tr>
<td>Final Balance of account2 after depositing is</td>
<td>20500.0</td>
</tr>
<tr>
<td>account1 was created at</td>
<td>Fri Feb 09 13:06:42 AST 2018</td>
</tr>
<tr>
<td>account2 was created at</td>
<td>Fri Feb 09 13:06:42 AST 2018</td>
</tr>
<tr>
<td>Number of created Accounts:</td>
<td>2</td>
</tr>
</tbody>
</table>

**Homework:**

Modify your testing main to define 10 Account objects in an array using the default constructor. Deposit a salary of SR10000 to the created Account objects and display their balance. Hint: use loops to iterate between objects.