CS203 Java Programming and Application
Fall, 2016

Assignment 2
Assigned Date: Sunday, October 9, 2016
Due Date: Midnight Sunday, October 23, 2016

QUESTION 1 (25 MARKS)

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.

Implement the class in Java as required by the design above. (*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.)

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.

QUESTION 2 (30 MARKS)

Use the **Account** class created in Question 1 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. Below is a sample run of the program:

```
Enter an id: 4 ↵Enter

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

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

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

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

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

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

Enter an id:
```

QUESTION 3 (20 MARKS)

This question demonstrates the use of inheritance and polymorphism.

Design a Ship class that the following members:
- A field for the name of the ship (a string).
- A field for the year that the ship was built (a string).
- A constructor and appropriate accessors and mutators.
- A toString method that displays the ship's name and the year it was built.

Design a CruiseShip class that extends the Ship class. The CruiseShip class should have the following members:
- A field for the maximum number of passengers (an int).
- A constructor and appropriate accessors and mutators.
- A toString method that overrides the toString method in the base class. The CruiseShip class's toString method should display only the ship's name and the maximum number of passengers.

Design a CargoShip class that extends the Ship class. The CargoShip class should have the following members:
- A field for the cargo capacity in tonnage (an int).
- A constructor and appropriate accessors and mutators.
- A toString method that overrides the toString method in the base class. The CargoShip class's toString method should display only the ship's name and the ship's cargo capacity.

Demonstrate the classes in a program that has a Ship array. Assign various Ship, CruiseShip, and CargoShip objects to the array elements. The program should then step through the array, calling each object's toString method.

QUESTION 4 (15 MARKS)

The **split** method in the **String** class returns an array of strings consisting of the substrings split by the delimiters. However, the delimiters are not returned. Implement the following new method that returns an array of strings consisting of the substrings split by the matching delimiters, including the matching delimiters.

public static String[] split(String s, String regex)

For example, **split("ab#12#453", "#")** returns **ab**, **#**, **12**, **#**, **453** in an array of **String**, and **split("a?b?gf#e", "[?#]")** returns **a**, **?**, **b**, **?**, **gf**, **#**, and **e** in an array of **String**.

Write a test program to show that your split method works properly.

QUESTION 5 (10 MARKS)

The **String** class is provided in the Java library. Provide your own implementation for the following methods (name the newclass **MyString2**):

public MyString2(String s);
public int compare(String s);
public MyString2 substring(**int** begin);
public MyString2 toUpperCase();
public char[] toChars();
public static MyString2 valueOf(**boolean** b);

Write a test program to show that all your methods work properly.