CS 1400-03 Introduction to Programming and Problem Solving
Coding Practice #6
(Due: 11:59 PM, Friday, 3/12/2021)

Except Coding Practice #1, I will not grade your coding practice submissions. Instead, they will be treated as participation points. On blackboard, you will receive full points as long as you work on the exercises, which don't necessary mean they are all correct. Please check your own programs carefully and make sure they do generate the desired output.

Objectives:

- Be able to write
  - Classes, instance variables, instance methods, constructors, setters, and getters
  - Driver program to demonstrate the capabilities and features of a class
- Be able to test and debug a program

Change your working directory to `cs1400/codingPractice` for this assignment.

Task #1 Car

Write a `Car` class that has the following private fields:

`yearModel`: an `int` that holds the car's year model
`make`: a `String` that holds the make of the car
`speed`: an `int` that holds the car's current speed

The class should have the following public methods:

`constructor`: Accepts the car's year model and make as arguments. The constructor should assign 0 to the speed field.
`setters`: One for each object's yearModel, make, and speed fields.
`getters`: One for each object's yearModel, make, and speed fields.
`accelerate`: This method should add 5 to the speed field each time it is called.
`brake`: This method should subtract 5 from the speed field each time it is called.

Write a program, called `CarTest.java`, that demonstrates the `Car` class by doing the following:

a. creating a `Car` object (e.g. 2018 Porsche)
b. displaying the current status of the car
c. calling the `accelerate` method five times
d. displaying the current speed of the car
e. calling the `brake` method five times
f. displaying the current speed of the car again

The following are sample output of the program.

```
Current status of the car:
Year model: 2018
Make: Porsche
Speed: 0

Accelerating...
Now the speed is 25

Braking...
Now the speed is 0
```

Task #2 Savings Account

Design a `SavingsAccount` class that stores a savings account's annual interest rate and balance. The class constructor should accept the amount of the savings account's starting balance and annual interest rate. The class should also have methods for subtracting the amount of a withdrawal, adding the amount of a deposit, and adding the amount of monthly interest to the balance. The monthly interest rate is the annual interest rate divided by twelve. To add the monthly interest to the balance, multiply the monthly interest rate by the balance, and add the result to the balance. The method `addInterest()` will also return the amount of monthly interest after adding it to the balance. Here is the UML diagram for class `SavingsAccount`.

SavingsAccount
- balance: double - annualInterestRate: double
+ SavingsAccount(bal: double, rate: double) + getBalance(): double + getInterestRate(): double + deposit(amount: double): void + withdraw(amount: double): void + addInterest(): double

Test the class in a program `SavingsAccountTest.java` that calculates the balance of a savings account at the end of a period of time. It should ask the user for the annual interest rate, the starting balance, and the number of months that have passed since the account was established. A loop should then iterate once for every month, performing the following:

a. Ask the user for the amount deposited into the account during the month. Use the class method to add this amount to the account balance.
b. Ask the user for the amount withdrawn from the account during the month. Use the class method to subtract this amount from the account balance.
c. Use the class method to calculate the monthly interest and add the result to the balance.

After the last iteration, the program should display the total amount of deposits, the total amount of withdrawals, the total interest earned, and the ending balance. ***Note that in any month, if the withdraw amount is greater than the balance, set balance to 0, and the iteration continues.***

The following are examples of program executions, where user's input is shown in bold. Make sure your program generates the same output as below, including spacing.

```
fcsang@garrison ~/cs1400/codingPractice $ java SavingsAccountTest
Enter the savings account's starting balance: 100
Enter the savings account's annual interest rate: .12
How many months have passed since the account was opened? 1
Enter the amount deposited during month 1: 100
Enter the amount withdrawn during month 1: 0
Total deposited: $100.00
Total withdrawn: $0.00
Interest earned: $2.00
Ending balance: $202.00
```

```
fcsang@garrison ~/cs1400/codingPractice $ java SavingsAccountTest
Enter the savings account's starting balance: 100
Enter the savings account's annual interest rate: .12
How many months have passed since the account was opened? 3
Enter the amount deposited during month 1: 110
Enter the amount withdrawn during month 1: 10
Enter the amount deposited during month 2: 120
Enter the amount withdrawn during month 2: 20
Enter the amount deposited during month 3: 130
Enter the amount withdrawn during month 3: 30
Total deposited: $360.00
Total withdrawn: $60.00
Interest earned: $9.07
Ending balance: $409.07

fcsang@garrison ~/cs1400/codingPractice $ java SavingsAccountTest
Enter the savings account's starting balance: 100
Enter the savings account's annual interest rate: .12
How many months have passed since the account was opened? 3
Enter the amount deposited during month 1: 100
Enter the amount withdrawn during month 1: 0
Enter the amount deposited during month 2: 0
Enter the amount withdrawn during month 2: 300
Enter the amount deposited during month 3: 100
Enter the amount withdrawn during month 3: 0
Total deposited: $200.00
Total withdrawn: $300.00
Interest earned: $3.00
Ending balance: $101.00
```

Submission:
Generate a `script` file `practice6.txt` with appropriate time stamps and the following steps visible:

1) a `pwd` to show the current working directory
2) a `ls -l` to show in long format the files in your `cs1400/codingPractice` directory
3) display both `Car.java` and `CarTest.java`
4) compile `CarTest.java`
5) run `CarTest`
6) display both `SavingsAccount.java` and `SavingsAccountTest.java`
7) compile `SavingsAccountTest.java`
8) run `SavingsAccountTest` three times with all the given test cases

Submit the script file `practice6.txt` on Bb, under the Coding Practice Folder, Practice #6 link.