# Java Programming (COMP217009) Spring, 2020

Instructor: Sangtae Ahn

# **Final Exam**

- The first line in all your codes should type your student ID and name as a comment.
   ex) // 20201234 Sangtae Ahn
- Your code should satisfy all the given requirements.

### Q1. (10 points)

Fill out the missing parts (marked as // FILL OUT //) to design a class named *Location* for finding maximal, minimal values and their locations in a two-dimensional array.

- 1) The class Location contains public data fields row, column, maxValue, and minValue
- 2) Public data fields **maxValue** and **minValues** are *double* data types and they store the maximal and minimal values in a two-dimensional array, respectively.
- 3) Public data fields **row** and **column** are *int* data types and they store indices for **maxValue** and **minValue** in a two-dimensional array.
- 4) Print maxValue, minValue, row and column.
- 5) Do not edit the code or rename the variables.
- 6) Submission File Name: Q1 FindLocation.java

# Q2. (10 points)

According to the given UML, design a class named *Rectangle* to represent the width, height, area, and perimeter. Satisfy the following requirements.

1) Class name: Rectangle

2) Write a test program that creates 3 *Rectangle* objects: *Rectangle()*, *Rectangle(4,40)* and *Rectangle(3.5,35.9)* 

3) Print the width, height, area, and perimeter for each object.

4) Submission File Name: Q2\_TestRectangle.java

Rectangle	
width: double	The width of this rectangle (default 1).
height: double	The height of this rectangle (default 1).
Rectangle()	Constructs a default rectangle.
Rectangle(width: double, height: double)	Constructs a rectangle with the specified width and height.
getArea(): double	Returns the area of this rectangle.
getPerimeter(): double	Returns the perimeter of this rectangle.

### Q3. (15 points)

According to the given UML, design a class named *Time* to represent the hour, minute, and second. Satisfy the following requirements.

1) Class name: Time

2) Write a test program that creates 3 *Time* objects: *Time()*, *Time(5555500000)*, and *Time(5, 23, 55)* 

3) Print the hour, minute, and second for each object.

4) Submission File Name: Q3\_TestTime.java

Time
-hour: int
-minute: int
-second: int
+Time()
+Time(elapseTime: long)
+Time(hour: int, minute: int, second: int)
+getHour(): int
+getMinute(): int
+getSecond(): int
+setTime(elapsedTime: long): void

The hour for the time.

The minute for the time.

The second for the time.

Constructs Time for the current time.

Constructs Time with a specified elapse time in milliseconds.

Constructs Time with the specified hour, minute, and second.

Returns the clock hour for the time.

Returns the minute for the time.

Returns the second for the time.

Sets a time for the specified elapsed time.

### Q4. (15 points)

According to the given UML, design a class named *Triangle* that extends *GeometricObject*. Satisfy the following requirements.

- 1) An abstract class is given: GeometricObject.java
- 2) Write a test program that prompts the user to enter three sides of the triangle, a color, and a Boolean value to indicate whether the triangle is filled.
- 3) Print the area, perimeter, color, and true or false to indicate whether the triangle is filled or not.
- 4) Override the *getArea* method from **GeometricObject.java**Override the *getPerimeter* method from **GeometricObject.java**Override the *toString* method from **GeometricObject.java**(toString method returns each side of the triangle)
- 5) Submission File Name: Q4\_TestTriangle.java

	¬
GeometricObject	
<u> </u>	
Triangle	
-side1: double	Three sides of the triangle.
-side2: double	
-side3: double	
+Triangle()	Creates a triangle with default sides (1).
+Triangle(side1: double, side2: double, side3: double)	Creates a triangle with the specified sides.
+getSide1(): double	Returns side1 of this triangle.
+getSide2(): double	Returns side2 of this triangle.
+getSide3(): double	Returns side3 of this triangle.
+getArea(): double	Returns the area of this triangle.
+getPerimeter(): double	Returns the perimeter of this triangle.
+toString(): String	Returns a string representation of this triangle.

## Q5. (15 points)

Write a Java program that counts the number of characters, words, and lines in a file (file reading and writing). Words are separated by a space.

- 1) A text file is given: Lincoln.txt
- 2) Check the file exists. If not, print this message "Source file does not exist" and terminate the program.
- 3) Use *Exception*.
- 4) Use the **Scanner** class to read the file.
- 5) Write a text file (Q5 CountLincoln.txt) that includes the number of characters, words, and lines.
- 6) Check the file (Q5\_CountLincoln.txt) already exists. If so, print this message "New file already exists" and terminate the program.
- 7) Submission File Names: Q5\_CountLincoln.java and Q5\_CountLincoln.txt

#### Q6. (15 points)

According to the given UML, design a class named *Account* to represent the balance, monthly interest, date when an account was created.

- 1) Class name: Account
- 2) 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%.
- 3) Use the *withdraw* method to withdraw \$2,500 Use the *deposit* method to deposit \$3,000
- 4) Print the balance, monthly interest, and date when the account was created.
- 5) Submission File Name: Q6 TestAccount.java

						ı
А	C	C	O	п	n	T

-id: int

-balance: double

-annualInterestRate: double

-dateCreated: java.util.Date

+Account()

+Account(id: int, balance: double)

+getId(): int

+getBalance(): double

+getAnnualInterestRate(): double

+getMonthlyInterestRate(): double

+getDateCreated(): java.util.Date

+setId(id: int): void

+setBalance(balance: double): void

+setAnnualInterestRate(

annualInterestRate: double): void

+getMonthlyInterest(): double

+withdraw(amount: double): void

+deposit(amount: double): void

The ID of this account (default 0).

The balance of this account (default 0).

The annual interest rate of this account (default 0).

The date when this account was created.

Constructs a default account.

Constructs an account with the

specified ID and balance.

Returns the ID of this account.

Returns the balance of this account.

Returns the interest rate of this account

Returns the monthly interest rate of this account

Returns the date when this account was created.

Sets a new ID of this account.

Sets a new balance for this account.

Sets a new interest rate for this account.

Returns the monthly interest of this account.

Withdraws the specified amount from this account.

Deposits the specified amount to this account.

#### Q7. (20 points)

Write a Java program that simulate an ATM machine by using the Account class created in Q6.

Create 10 accounts in an array with ID 1,2,3,... 10 and an initial balance of \$100.
 The system prompts the user to enter an ID.
 If the ID entered incorrectly, ask the user to enter a correct ID.

2) Once ID is accepted, the main menu is displayed as follows.

Choice 1 : check the current balance

Choice 2: withdraw money

Choice 3: deposit money

Choice 4: exit the menu (return to enter an ID)

- 3) Once you exit, the system will prompt for an ID again.
- 4) The system will not stop once the system starts.
- 5) Submission File Name: Q7 TestATM.java

```
Here is a sample run.
Enter an ID: 4
Main menu
                                                 Main menu
1: check balance
                                                 1: check balance
2: withdraw
                                                 2: withdraw
3: deposit
                                                 3: deposit
4: exit
                                                 4: exit
Enter a choice: 1
                                                 Enter a choice: 3
The balance is 100.0
                                                 Enter an amount to deposit: 10
Main menu
                                                 Main menu
1: check balance
                                                 1: check balance
2: withdraw
                                                 2: withdraw
3: deposit
                                                3: deposit
4: exit
                                                4: exit
Enter a choice: 2
                                                Enter a choice: 1
Enter an amount to withdraw: 30
                                                The balance is 80.0
Main menu
                                                 Main menu
1: check balance
                                                 1: check balance
2: withdraw
                                                 2: withdraw
3: deposit
                                                3: deposit
4: exit
                                                4: exit
Enter a choice: 1
                                                Enter a choice: 4
The balance is 70.0
                                                 Enter an ID:
```