Assigment 03 – Polymorphism & Abstraction

Duration: 70'

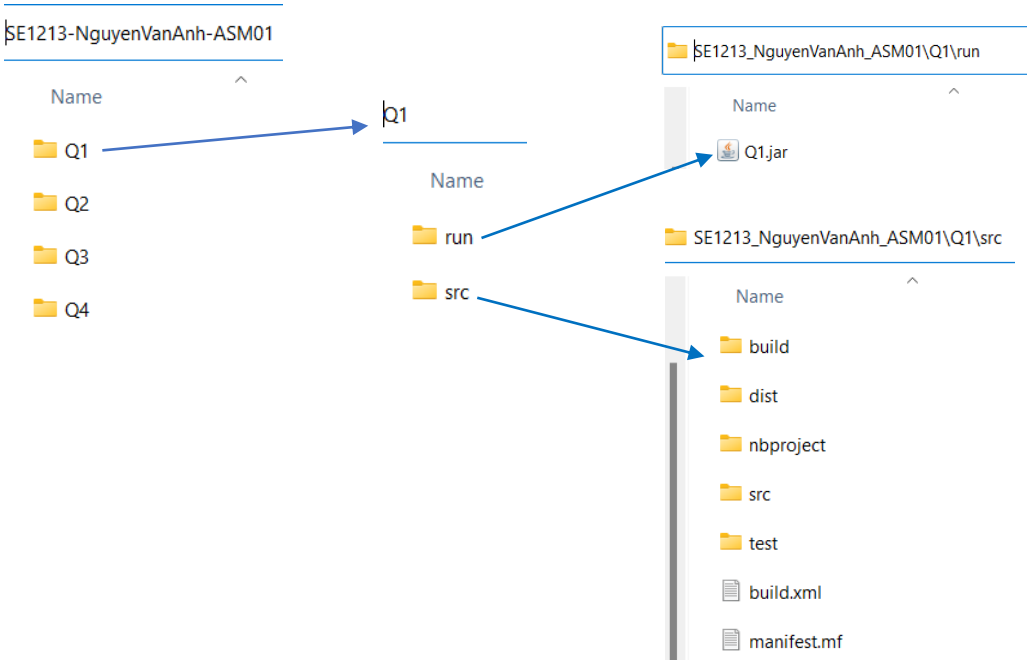
==========

Software Requirements

- Netbean 8.2 or later, JDK 8 or later, Notepad, Command Prompt, WinRAR / WinZip with Windows Explorer (File Explorer) on Windows 7 and above.

Instructions

- Step 1: Students download the given materials from LMS.
- Step 2: Students read questions and prepare answers in the given template.
- Step 3: Prepare to submit the answer:
  - For each question (e.g., question **Q1, Q2, Q3,…**), please create two sub-folders: **run** and **src**.
  - Copy the *.**jar** file into the **run** folder, and the entire project source code file into the **src** folder.
- Step 4: Submit a solution for each question:
  - Create a folder named : RollNumber_FullName_ASM0x (**x : 1,2,..,6**) that contains folders (created Step 03) as the below figure:



  - Use WinRAR / WinZip tool to compress the **RollNumber_FullName_ASM0x** folder and submit it to LMS

❖ **Importance:**

- *Do not change the names of the folders specified (or required) in the exam and the name of submit folder in Step 04 must be correct. If you change it (incorrectly), the grading software can not find the execute file (.jar) to score, thus the exam result will be 0.*

==========

❖ **Question 2: (10 marks)**

Write an abstract class **Beverage** and a class **Coffee** extending from **Beverage** (i.e. Beverage is a superclass and Coffee is a subclass) with the following information :

Note: String comparison in the assginment no case-sensitive

Beverage

Where:

-id: String -name: String -price: double -quantity: int
+Beverage() +Beverage(id:String, name:String, price:double, quantity:int) +getId():String +setId(value:String):void +getName (): String +setName(value:String):void +setPrice(value:double):void +getPrice (): double +setQuantity(value:int):void +getQuantity (): int +abstract subTotal():double +override toString():String

- Check validation (apply to constructors and setters):
  - check the **id** is not empty and formatted : XXxxx (X is letters, x is digits). If value is invalid then set id to TN000
  - check the **name** is not empty and length from 5 to 50 characters. If value is invalid then set name to "new beverage".
  - check the **price** from 1 to 5000. If value invalid then set price to 1
  - check the **quantity** from 1 to 100. If value invalid then set quantity to 1
- Beverage() - default constructor (numeric value is 0, string value is empty)
- Beverage(id:String, name:String, price:double, quantity:int): constructor, which sets values to id, name, price, quantity)
- getName ():String – return name with title case and each words is seperated by a space
- toString():String – return the string of format :**id, name, price, quantity, subTotal = price * quantity** (the **price** and **subTotal** are formatted with three decimal places).

Coffee
-expire: int -type: String
+Coffee() +Coffee(id: String, name: String, price: double, quantity: int, type: String, expire:int) +setExpire(value:int):void +getExpire (): int +getType (): String +setType(value:String):void +override toString():String +override subTotal (): double

Where:

- Check validation (apply to constructors and setters):
  - check the **expire** from 1 to 180. If the value is invalid then set expire to 180
  - check the **type** in [special, high, medium, low]. If the value is invalid then set the type to "medium"
- Coffee() - default constructor (numeric value is 0, string value is empty)
- Coffee(id: String, name: String, price: double, quantity: int, type: String, expire:int): constructor, which sets values to id, name, price, quantity, type, expire
- toString():String – return the string of format : **id, name, type , expire , price, quantity, subTotal,** (the **price** and **subTotal** are formatted with three decimal places).
- getType ():String – return type with title case
- override subTotal (): double – sub total = price * quantity * rate (default rate = 1)
  -If type is "special" or id started with "DB" then rate = 1.2
  -If type is "high" or id started with "HC" then rate = 1.1

-If type is “medium” and expire <= 30 then rate = 0.5

Do not format the result.

The program output might look something like this:

```
#Case 1:  [2.0 marks]
Test Coffee class
1.Test validation
2.Test toString()
Enter Test Case No.(1 | 2):1
Enter id:888
Enter name:abc
Enter price:-1
Enter quantity:-1
Enter type:abc
Enter expire:0
OUTPUT:
TN000,New Beverage,Medium,180,1.000,1,1.000
--------------------------------------------------

#Case 2:  [2.0 marks]
Enter Test Case No.(1 | 2):1
Enter id:H1111
Enter name:caphe g7
Enter price:90000
Enter quantity:80000
Enter type:dac biet
Enter expire:9000
OUTPUT:
TN000,Caphe G7,Medium,180,1.000,1,1.000
--------------------------------------------------

#Case 3:  [2.0 marks]
Test Coffee class
1.Test validation
2.Test toString()
Enter Test Case No.(1 | 2):2
Enter id:DB001
Enter name:ca phe g7
Enter price:100
Enter quantity:200
Enter type:special
Enter expire:190
OUTPUT:
DB001,Ca Phe G7,Special,180,100.000,1,120.000
--------------------------------------------------

#Case 4:  [2.0 marks]
Test Coffee class
1.Test validation
2.Test toString()
Enter Test Case No.(1 | 2):2
Enter id:HC099
Enter name:caphe g9
Enter price:100
```

```
Enter quantity:100
Enter type:low
Enter expire:200
OUTPUT:
HC099,Caphe G9,Low,180,100.000,100,11000.000

--------------------------------------------------

#Case 5:  [2.0 marks]
Test Coffee class
1.Test validation
2.Test toString()
Enter Test Case No.(1 | 2):2
Enter id:GC991
Enter name:coffee
Enter price:2000
Enter quantity:50
Enter type:medium
Enter expire:30
OUTPUT:
GC991,Coffee,Medium,30,2000.000,50,50000.000
```