BACS2023 Object-Oriented Programming

Practical 4: Objects and Classes

Q1. (a) Given the **Employee** class as below:

```
public class Employee{
   String name;
   double salary;

   Employee(String employeeName, double currentSalary)   {
      name = employeeName;
      salary = currentSalary;
   }

   void raiseSalary(double percent){
      salary = salary + (salary * percent / 100.0);
   }
}
```

Write a driver program that creates an **Employee** object and displays his/her salary before and after a salary raise of 8%.

(b) *(Access modifiers)* Modify the **Employee** class such that data field encapsulation is applied and the correct visibility modifiers are used. Specifically,

- Make the data fields **private**.
- Implement setters and getters for each data field.
- All constructors and methods should be declared as **public**.

Modify your driver program from Q1(a) accordingly.

(c) *(Constructors and constructor overloading)* Add the following constructors to the **Employee** class.

(i) A no-argument constructor that sets the employee name to an empty string (i.e., "") and the salary field to 0.0.

(ii) A constructor with 1 parameter. This constructor receives a String argument storing a name. The salary field should be set to 0.0.

Implement a driver program that creates 2 **Employee** objects each using a different constructor above. Include statements to:

- Display the employee with a higher salary
- Compute and display the total salary of the 2 employees

Q2. (a) Create a class named **Student** that includes data fields for the student ID, name, number of quizzes taken and a total quiz score. In your class, include the following :

- A no-arg constructor
- A constructor with 2 parameters for student ID and name
- accessors for all data fields
- appropriate mutators
- addQuiz(int score) [each quiz is worth 10 marks]
- getTotalScore()
- getAverageScore()

IMPORTANT NOTE: Remember to adhere to the ***standard Java naming convention*** for your class names, data field names and method names.

(b) Create a driver program that tests all constructors and methods.

Q3. (a) (*Using the* ***this*** *keyword*) Modify the **`Student`** class from Q2 based on the following:

- In the parameterized constructor and set methods, the parameters have the same name as the corresponding data field.
- The no-arg constructor will invoke the parameterized constructor

(b) (*Using static variables and methods*) Suppose your tutor wants the contribution of quizzes to the coursework mark be stored in the **`Student`** class and this value may be revised in other semesters. Modify the **`Student`** class accordingly.

(c) Modify the driver program in Q2(b) to include the following:

- Set a value of 20% for the contribution of quizzes to the coursework mark
- Calculate and display the quiz marks (out of 20 marks) obtained by each student

Q4. Consider the following UML diagram representing a class to be used for keeping track of short courses offered by Alpha College.

Course
− courseTitle: String − feesPerStudent : double − noOfStudents: int − studentNames : String [] − courseCount: int
+ Course(courseTitle: String, feesPerStudent: double) + Set and Get methods + calcFeesCollected(): double + addStudent(studName : String) : void + toString(): String

Note:

- The data member **`studentNames`** is an array that can store a maximum of 100 student names. (Assume that maximum enrollment for a course is 100.)
- **`courseCount`** is used to keep track of the total number of courses offered in the college.

(a) Implement the **`Course`** class. Note the following requirements:

- In the method **`calcFeesCollected`**`()`, use the formula given below to compute the total amount of fees collected from a course.
  `feesPerStudent * noOfStudents`
- The **`addStudent`**`()` method will allow a new student who has enrolled for a course, to be added to the **`studentNames`** array.
- The **`toString`**`()` method must return a string containing all the information for a course except the student names.

(b) Alpha College has recently offered a new course with details as given below:

Course Title : Introduction to Computers
Fees per student : RM250

Currently, 3 students namely Ali Said, Wong Ken and Peter Lim have registered for this course.

Create a program that display the details of the new course, including the total fees collected for the course. The output should also contain the names of students who have enrolled for the course.

Q5. The Road Transport Department needs to keep track of car registrations. For every car registered, the following information will be stored: *registration number*, *owner's name*, *owner's IC number*, the car's *plate number*, *make*, *model*, *engine capacity* (e.g. 1.8L, 2.0L), *color* and *year manufactured*.

[Note: the required information to be stored has been greatly simplified to make it manageable as a practical exercise].

Identify the required classes and appropriate relationships (aggregation or composition) between the classes. Implement all the classes and then write a driver program that will obtain input details from the user on the cars registered. Then, produce a listing containing the details of the car registrations. A sample listing is shown below:

Car Registration Listing

Reg No.	Name	IC No.	Plate No.	Color	Year	Make	Model	Capacity
1001	John Wayne	111111111	ABC123	Blue	2010	Toyota	Vios	1.5
1002	Bea Arthur	222222222	WEA888	Red	2010	Nissan	Teana	2.0
1003	Meg Ryan	333333333	PBL168	Black	2011	Honda	City	1.6
1004	Jane Doe	444444444	BBB777	White	2011	Nissan	Teana	2.0
1005	Al Johnson	555555555	CAT118	Green	2012	Toyota	Vios	1.5
1006	Ned Beatty	666666666	TV798	Blue	2012	Toyota	Vios	1.5

Note: As the focus of this question is the implementation of composition and aggregation, you may simplify your solution as follows:

- Use the `String` type for the owner's name.
- For the entity classes, you only need to implement the following:
  - A parameterized constructor to initialize all the instance variables.
  - The `toString` method to return all the data fields' values as a string.
  - Automate the registration number.
- In your driver program,
  - you may create an array of car type objects where each object stores the information of a car type. This array may be created using the *array initializer*. Then, during user input for a car registration, you may display the various car types for the user to select from.
  - use a fixed for-loop to input several registration data.
  - assume that other than the owner's name, the other string-typed data will only consist of a single word .
  - input validations are not needed.

Q6. Mickey Cake House needs a cake ordering application. A customer may order more than one cake at a time. For each cake ordered, details to be stored include the flavour, weight and the quantity ordered. There are 5 available cake flavours: Chocolate Moist, Strawberry, Blueberry, American Chocolate and Tiramisu. The bakery sells cakes in 3 different weights: 1kg, 2kg and 3kg. The prices for cakes differ according to the cake flavour as well as the weight of the cake. The formulas used to calculate the price of the 2kg and 3kg cakes are as follows:

- Price of 2kg cake = (Price of 1kg cake X 2) – RM5.00
- Price of 3kg cake = (Price of 1kg cake X 3) – RM15.00

A sample dialog of the application is shown in Figure 1:

```
Mickey Cake House


How many types of cake you would like to order: 2

  No Flavours                1kg       2kg       3kg
  1. Chocolate Moist      RM 40.00  RM 75.00  RM105.00
  2. Strawberry           RM 50.00  RM 95.00  RM135.00
  3. Blueberry            RM 45.00  RM 85.00  RM120.00
  4. American Chocolate   RM 55.00  RM105.00  RM150.00
  5. Tiramisu             RM 30.00  RM 55.00  RM 75.00


Cake item 1
--------------
Enter your choice of cake flavour (1 - 5): 1
Enter the weight of the cake (1 - 1kg, 2 - 2kg and 3 - 3kg): 3
Enter quantity ordered: 2

Cake item 2
--------------
Enter your choice of cake flavour (1 - 5): 5
Enter the weight of the cake (1 - 1kg, 2 - 2kg and 3 - 3kg): 1
Enter quantity ordered: 3


Order Details:
----------------
No  Cake Flavour     Weight    Unit Price(RM)   Quantity  Total Price(RM)
--  ------------     ------    --------------   --------  --------------
1   Chocolate Moist    3kg          105.00             2          210.00
2   Tiramisu           1kg           30.00             3           90.00
-------------------------------------------------------------------------
                                                Grand Total:       300.00
```

Figure 1 Sample dialog for the cake order application program

(a) In groups of 2-3 students, identify the required classes and the relationships between these classes by applying relevant class design guidelines. Then, draw a UML class diagram to show your class design.

(b) Create the required classes.

(c) Write a Java program that, using the classes from part (b), performs operations to handle cake orders as depicted in Figure 1.