Object Oriented Programming (SWE-103) | LAB EXAM

Section A

INSTRUCTIONS:

Attempt 4 questions in all.

The formula for Question is next question is

Question 1= (Rollno mod 10)+1 for e.gRollno is 125 mod 10=5+1 Result 6.

Question 2=(Rollno mod 10)+2 =5+2 Result 7

Question 3=(Rollno mod 10)+3 =5+3 Result 8

Question 4=(Rollno mod 10)+4 =5+4 Result 9

You need to attempt Part 6, 7, 8 & 9.

(NOTE: If Result greater than 10, then take Result mod 10)

1. Write an application for a college's admissions office. Prompt the user for a student's numeric high school grade point average (for example, 3.2) and an admission test score from 0 to 100. Display the message "Accept" if the student has any of the following: A grade point average of 3.0 or above and an admission test score of at least 60. A grade point average below 3.0 and an admission test score of at least 80. Throw an exception if the student does not meet either of the qualification criteria. Save the file as Admission.java.

2. Write and execute a Java program that defines an ABSTRACT class for a VEHICLE. The common information for a vehicle includes model, maximum speed, and engine type (piston or valve). The class for vehicle must have a constructor that takes values for the common information as parameters and assigns those values. The class for vehicle must have an ABSTRACT method showData() that is responsible for showing all information in any subclass of vehicle. Define a class for a CAR that is a subclass of vehicle. The subclass must have 2 information items of its own, manufacturing year and number of doors. Define another class for a TRUCK that is a subclass of vehicle. The subclass must have 2 information items of its own, load capacity and number of wheels. Each subclass must have the capability to show all of its information by realizing its parent class (super class). Create sample objects (3 each) of each subclass (filled with sample values) and use the showData() method to show all information for each object.

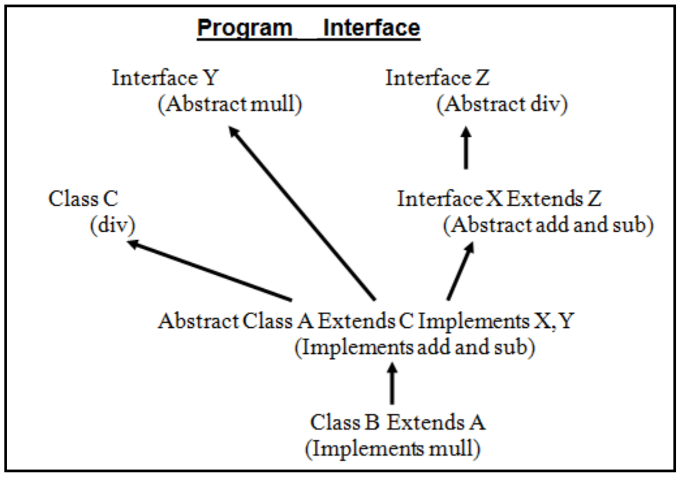
3. Create a class named Box that includes integer data fields for length, width, and height. Create three constructors that require one, two, and three parameters, respectively.

When one argument is passed to the constructor, assign it to length, assign zeros to height and width, and display “Line created”. When two arguments are used, assign them to length and width, assign zero to height, and display “Rectangle created”. When three arguments are used, assign them to the three variables and display “Box created”. Save this file as Box.java. Create an application named TestBox that demonstrates each method works correctly. Save the application as TestBox.java.

4. Create an interface named Turner, with a single method named turn(). Create a class named Leaf that implements turn() to display “Changing colors”. Create a class named Page that implements turn() to display “Going to the next page”. Create a class named Pancake that implements turn() to display “Flipping”. Write an application named DemoTurners that creates one object of each of these class types and demonstrates the turn() method for each class. Save the files as Turner.java, Leaf.java, Page.java, Pancake.java, and DemoTurners.java.

5. Define a class for an inventory item that includes the information Id, name, weight, color, number of units, and price. Suppose that, the information for Id and name MUST be PROVIDED WHEN an OBJECT for an inventory item is CREATED and this information CAN NEVER be CHANGED. Define a 2 parameter constructor in the class that takes the Id and name as parameters and assigns the values. Also, define a 3 parameter constructor in the class that takes the Id, name, and number of units as parameters and assigns the values. Both constructors MUST RESET the REMAINING FIELDS. Define get and set methods for fields other than Id and name. The program must create 4 objects for inventory item and apply some of the get and set methods to each object. The program must also show the contents of at least 2 objects after applying the get and set methods. A methodsuch as showInfo() may be defined for this purpose.

6. Implement this diagram into equivalent java code



7. Create a class named Order that performs order processing of a single item. The class has five fields: customer name, customer number, quantity ordered, unit price, and total price. Include set and get methods for each field except the total price field. The set methods prompt the user for values for each field. This class also needs a method to compute the total price (quantity times unit price) and a method to display the field values. Create a subclass named ShippedOrder that overrides computePrice() by adding a shipping and handling charge of $4.00. Write an application named UseOrder that instantiates an object of each of these classes. Prompt the user for data for the Order object, and display the results; then prompt the user for data for the ShippedOrder object, and display the results. Save the files as Order.java, ShippedOrder.java, and UseOrder.java.

8. Develop a registration system for a University. It should consist of three classes namely Student, Teacher, and Course. For example, a student needs to have a name, roll number, address and GPA to be eligible for registration. Therefore choose appropriate data types for encapsulating these properties in a Student objects. Similarly a teacher needs to have name, address, telephone number, and a degree (or a list of degrees) he/she has received. Finally courses must have a name, students (5 students) registered for the course, and a teacher assigned to conduct the course. Create an object of Course with 5 Students and a Teacher. A call to a method, say printDetails(), of the selected course reference should print name of the course, details of the teacher assigned to that course, and names and roll numbers of the students enrolled with the course.

9. Create an abstract class called Student. The Student class includes a name and a Boolean value representing full-time status. Include an abstract method to determine the tuition, with full-time students paying a flat fee of $2,000 and part-time students paying $200 per credit hour. Create two subclasses called FullTime and PartTime. Create an application that demonstrates how to create objects of both subclasses. Save the files as Student.java, FullTime.java, PartTime.java, and UseStudent.java.

10. Write an application for a furniture company; the program determines the price of a table. Ask the user to choose 1 for pine, 2 for oak, or 3 for mahogany. The output is the name of the wood chosen as well as the price of the table. Pine tables cost $100, oak tables cost $225, and mahogany tables cost $310. If the user enters an invalid wood code, set the price to 0. Save the file as Furniture.java. Throw an exception if the input is in the wrong format or if it is out of range, print an error message, and halt gracefully.