**JAVA PROBLEM SOLVING USING OBJECT ORIENTED ANALYSIS AND DESIGN**

While working on the following exercises, the knowledge acquired during OOAD session should be utilized and Java design principles and best practices should be used. The objective of this session is to inculcate the following qualities in a Java developer:

1. Asking the right set of questions to understand the business requirements in hand.
2. Designing the solution using object oriented principles and practices, like encapsulation, inheritance, designing to interfaces, separation of concerns, cohesion and coupling, etc.

**P1** Consider the following problem description:

Employees receive their biweekly paychecks. They are paid their hourly rates for each hour worked; however, if they worked more than 40 hours per week, they are paid at 150 percent of their regular wage for those overtime hours.

Design the classes that you would use to implement the above.

**P2** Consider the following problem description:

You have to maintain the assessment scores of trainees in a batch, and rate them based on the scores. There are different types of assessments – online, coding and class performance. Different weightage will be given to the different assessments and final score will be calculated. Rating will be given based on the final score. Trainee information and scores will be saved in database.

Design the classes that you would use to implement the above.

**P3** Consider the following problem description:

You have to design an application that presents a quiz and grades user’s responses. A quiz consists of questions. There can be different kinds of questions:

* Fill-in-the-blank
* Choice (single or multiple)
* Numeric (where an approximate answer is ok, e.g., 1.33 or 4/3)

Design the classes for the above. You need to design the classes only and not the user interface for taking the test.

**P4** Consider the following problem description:

Your application needs a number sequence and there can be different kinds of sequences, like a sequence of random numbers, prime numbers, squares, odd numbers, etc. Different scenarios may use different sequences. A sequence generator should accept the kind of sequence and number of values in sequence and generate and display the sequence.

Design and implement the classes that you will be need to achieve the above. Use a test class to test the generation of the different number sequences.

(Hint: Have an interface Sequence with method next(), SquareSequence, RandomSequence, PrimeSequence, etc will implement this interface and override next() to return the next number in sequence. SequenceGenerator class will have a method generate(Sequence seq, intvaluesInSequence). This method will call seq.next() and return and print the next number in sequence.)

**P5** There is a new requirement to support transfer of funds from one bank account to another. How will you achieve this considering there can be different kinds of accounts.

**P6** Consider the following problem description:

You have to design an Order execution system with the following requirements:

1. The system accepts orders and executes the orders in sequence of order date.
2. Input to the system: Order attributes – Quantity, Product, Order date, Order Type (Sales or Purchase)
3. An Order can beeither a Sales order or Purchase Order.
4. Product can be two types: Durable, Consummable.
5. Durable product types are: *TOOTHBRUSH*,*BUCKET*,*SHOES*,*LUNCHBOX*
6. Consummable product types are: *BREAD*,*BUTTER*,*CAKE*,*PIZZA*,*BURGER*
7. The system will execute the order. Execution steps will be as follows:
   * The ordered quantity should be validated from the database. If quantity is not available, exception should be thrown, with proper error message.
   * Check the expiry date of the ordered product from the database. If product is expired, throw exception with proper error message.
   * Fetch the price of the product from database table.
   * If it is a Sales order, apply sales tax on the order price and calculate the final price.
   * If it is a Purchase order, apply vat on the order price and calculate the final price.
   * Save the order to database with all order details.
   * Also, the product inventory should be updated with quantity. In case of sales order, quantity will be deducted. In case of purchase order, quantity will be increased.
8. Final Output should be List of Orders Stored in the database table with quantities updated according to the sale and purchase of the Products.

Design the classes to achieve the above requirements.