Java Coding Assessment

Total Duration : 1 Hr **Marks : 100**

1.Design a class **Customer** in the **com.telecom.model** package containing the following private members:

- customerID (text)
- name (text)
- plan (text)
- balance (float)
- phoneNumber (text)
- email (text)

Define a public parameterized constructor to initialize the Customer object.

2.Design a class called **CustomerService** in the **com.telecom.service** package which contains an appropriate collection object to store Customer instances.

Implement the following operations:

1. addCustomer(Customer c)
   - To add a new Customer object into the collection.
2. searchByPlan(String plan)
   - Search customers based on their plan and if found, display the details.
3. displayAllCustomers()
   - Print the details of all the customers.
4. getCustomerWithHighestBalance()
   - Find and display the customer with the highest balance.

3.Create a class **CustomerUtil** in the package **com.telecom.util** which has the **main** method.

- Instantiate the CustomerService class.
- Read data from the user for 3 Customer objects.
- Perform the following validations.
  - plan must be “Prepaid”, “Postpaid”, or “Family”.
  - balance cannot be negative.

- customerID must start with 'C' and must be of length 5 characters.
- phoneNumber must be a 10-digit number.

If any of the validations fail, throw a user-defined exception **InvalidCustomerException** with an appropriate message (as specified in the validations above). Declare this exception in the **com.telecom.exception** package.

Implement the following operations:

- ✓ For a valid Customer, call the addCustomer method to add the customer objects into the collection.
- ✓ Search for customers by plan.
- ✓ Display all customer details.
- ✓ Find and display the customer with the highest balance.

Rubrics									
Class Design and Implementation (25 points)			Collection and Methods Implementation (35 points)		Validation and User Input Handling (20 points)		Code Quality and Readability (10 points)		Output (10 points)
Class Structure (10)	Parameterized Constructor (5)	Exception Handling (10)	Collection Usage (15)	Method Implementation (20)	Validation Logic (10)	User Input Handling (10)	Code Organization and Modularity (5)	Code Readability and Comments (5)	Output /Display (10)