# Part B

Handson Problem (1 question)

A DTH service provider is using an application for managing its users. This application has multiple modules, one of the module of this application is to add users. You have to implement the module and write test cases using JUnit and Mockito to test the service class of this module.

The description of different classes is given below:

**Class Descriptions**

1. DTO classes (com.tcs.dto)

    UserDTO
    - userId: Integer
    - userName: String
    - password: String
    - email: String
    - mobileNumber: String
    - address: AddressDTO

    AddressDTO
    - addressId: Integer
    - doorNumber: String
    - street: String
    - city: String
    - state: String
    - zipcode: Integer

2. **Exception class** (com.tcs.exception)

UserException
- serialVersionUID: long

3. **Validator class** (com.tcs.validator)

This class is used for validating the input values. Implement this class according to the class diagram and description given below:

Note : The methods of this class are static.

Method description

- validate(UserDTO userDTO)
  - This method will receive UserDTO object. In case of failed validation, exceptions are thrown with the corresponding messages as shown in the table below:

| Violation For | Message |
|---|---|
| userName | Validator.INVALID_USERNAME |
| password | Validator.INVALID_PASSWORD |
| email | Validator.INVALID_EMAIL |
| mobileNumber | Validator.INVALID_MOBILENUMBER |
| address | Validator.INVALID_ADDRESS |

- isValidUserName(String userName)
  - This method validates userName.
  - It should contain only alphabets and spaces, but should not start or end with a space.
  - If the above conditions are satisfied, return true, otherwise return false.

- isValidPassword(String password)
  - This method validates password.
  - It should have minimum 8 characters and at least one special character.

    - If the above conditions are satisfied, return true, otherwise return false.

- isValidEmail(String email)
    - This method validates email.
    - It should contain only alphabets, numbers and underscores, should have a dot (.) after, but not immediately following, an at-sign (@), should have at least one character before and after the ‘@’ and ‘.’ each.
    - If the above conditions are satisfied, return true, otherwise return false.

.

- isMobileNumber(String mobileNumber)
    - This method validates mobileNo.
    - It should have 10 digits, with all the digits not same.
    - If the above conditions are satisfied, return true, otherwise return false.

.

- isValidAddress(AddressDTO address)
    - This method validates address.
    - It should have the fields doorNumber, street, city, state and zipCode.
    - zipCode should have 5 digits, with all the digits not same.
    - If the above conditions are satisfied, return true, otherwise return false.

.

4. **UserServiceImpl** (com.tcs.service)

This is the service class of application. Implement this class according to class diagram and description given below:

UserServiceImpl
- userRepository: UserRepository

Instance variable description

- Injected userRepository using appropriate annotation.

Method description

addUser(User user)

- This method receives user details, validates it and send it to Repository to be added to database.
- It invokes validate(UserDTO userDTO) method of Validator for each user.
- If all the details are valid, it should invoke addUser method of UserRepository, which in turn returns an integer.
- It returns the value received in previous step.

5. **UserRepositoryImpl** (com.tcs.repository)

This is the Repository class of application.

Method description

addUser (User user)

- This method is used to save the user details in database.

6. **LoggingAspect** (com.tcs.utility)

This is an AOP aspect class. It contains advices which log exceptions thrown from service and repository classes.

```
LoggingAspect
        - LOGGER: Log
```

Method description

logRepositoryException(Exception exception)

- Annotate this method with appropriate advice which gets executed when exceptions are thrown only from RepositoryImpl class.
- It should also log the thrown exception at ERROR level.

logServiceException(Exception exception)

- Annotate this method with appropriate advice which gets executed when exceptions are thrown only from ServiceImpl class.
- It should also log the thrown exception at ERROR level.

7. **<u>UserInterface class</u>** (com.tcs)

This is the class which run your Spring Boot application.

+ userService: UserService
+ Environment: Environment
- LOGGER: Log

8. **<u>UserToTraineeApplicationTests class</u>**

This class is used for testing the addUser() method of UserServiceImpl class. Implement the following test cases in this class using JUnit and Mockito. Add appropriate annotations wherever requires.

Methods:

addUserValidTest
addUserInvalidUserNameTest
addUserInvalidPasswordTest
addUserInvalidEmailTest
addUserInvalidMobileNumberTest
addUserInvalidAddressTest