JP Assignment 1

Date of Submission (30 / 09 / 2024)

<table>
<tr><td>Q1</td><td>Explain JVM in detail with a diagram. Also explain how is Java platform independent.</td></tr>
<tr><td>Q2</td><td>WAP to accept a number from command line and print the sum of cubes of individual digits.</td></tr>
<tr><td>Q3</td><td>WAP to display following patterns
1.
1
2 3
4 5 6
7 8 9 10
.
.
.
N lines
2.
* * * * *
* * * *
* * *
* *
*
3.
A
ABA
ABCBA
ABCDCBA
ABCDEDCBA
.
.
N lines</td></tr>
<tr><td>Q4</td><td>WAP to find and display reverse of an array</td></tr>
<tr><td>Q5</td><td>Create a class Employee with data members empid, empname, designation and salary. Write methods get_employee() to take user input and show_employee() to display employee details. Make an array of objects of this class and accept and display information of ‘n’ employees.</td></tr>
<tr><td>Q6</td><td>What are Constructors? Explain different types of constructor with example codes.</td></tr>
<tr><td>Q7</td><td>You are designing a simple Banking System where you need to enforce certain rules regarding the immutability of certain classes and methods. Your task is to implement the following classes while utilizing the final keyword appropriately.

Requirements:

1. Class: BankAccount
o Attributes:
▪ accountNumber (String, should be final)
▪ accountHolder (String)
▪ balance (double)
o Methods:
▪ Constructor to initialize accountNumber, accountHolder, and balance.
▪ Getter methods for accountNumber, accountHolder, and balance.
▪ A method deposit(double amount) that increases the balance.
▪ A method withdraw(double amount) that decreases the balance, ensuring that the balance does not go below zero.
2. Class: SavingsAccount
o Inherits from BankAccount.
o Additional Attribute:
▪ interestRate (double)
o Methods:
▪ Constructor to initialize all attributes (including inherited ones).
▪ A method applyInterest() that calculates and adds interest to the balance based on interestRate.
▪ Override the withdraw method to ensure that a minimum balance of $100 is maintained in the account.
3. Class: CurrentAccount</td></tr>
</table>

	○ Inherits from BankAccount. ○ Additional Attribute: ▪ overdraftLimit (double, should be final) ○ Methods: ▪ Constructor to initialize all attributes (including inherited ones). ▪ Override the withdraw method to allow withdrawal up to the overdraft limit. 4. **Class: AccountUtils (final)** ○ This class should contain utility methods for account operations. ○ Methods: ▪ A static method transfer(BankAccount fromAccount, BankAccount toAccount, double amount) that transfers money from one account to another, ensuring both accounts have sufficient balance (considering overdraft for CurrentAccount).
Q8	You are tasked with developing a User Registration System that validates user input. To handle various validation errors, you will create custom exceptions. Your system should validate user information and throw appropriate exceptions when invalid data is provided. **Requirements:** 1. **Custom Exception Classes:** ○ **`InvalidUsernameException`**: ▪ This exception should be thrown if the username does not meet specific criteria: ▪ Must be at least 6 characters long. ▪ Cannot contain special characters (only letters and numbers). ○ **`InvalidEmailException`**: ▪ This exception should be thrown if the email format is invalid. The email should contain an "@" symbol and a domain (e.g., "example.com"). ○ **`WeakPasswordException`**: ▪ This exception should be thrown if the password does not meet the criteria: ▪ Must be at least 8 characters long. ▪ Must contain at least one uppercase letter, one lowercase letter, one digit, and one special character. 2. **Class: `User`** ○ Attributes: ▪ `username` (String) ▪ `email` (String) ▪ `password` (String) ○ Methods: ▪ Constructor to initialize the attributes. ▪ Validation method `validateUser()` that checks the username, email, and password using the defined criteria. ▪ If any validation fails, the corresponding custom exception should be thrown. 3. **Class: `UserRegistration`** ○ Contains a method `registerUser(User user)` which attempts to register a user by calling `validateUser()`. It should handle any exceptions thrown and print a user-friendly message.