G04-30

VIT
Vellore Institute of Technology

Slot :A2

School of Information Technology and Engineering

Winter Semester 2022-2023 (Freshers)

Continuous Assessment Test – II

Programme Name & Branch M.C.A & Computer Application

Course Name & code:ITA5004 & Object Oriented Programming using JAVA

Class Number (s): VL2022230500239, VL2022230500268, VL2022230500294

Faculty Name (s): Bimal Kumar Ray(10134), Shynu P G(12340), Thanga Mariappan L(19709)

Exam Duration: 90 Min. **Maximum Marks: 50**

General instruction(s):

Q.No.	Question	Max Marks
1.	Illustrate how to address the sorts of problems that can arise when you try to synchronize threads, let's consider a simple application in which several threads use a shared resource. You're familiar with those take-a-number devices that are used in bakeries to manage a waiting line. Customers take a number when they arrive, and the clerk announces who's next by looking at the device. As customers are called, the clerk increments the "next customer" counter by one. There are some obvious potential coordination problems here. The device must keep proper count and can't skip customers. Nor can it give the same number to two different customers. Nor can it allow the clerk to serve nonexistent customers. Our task is to build a multithreaded simulation that uses a model of a take-a-number device to coordinate the behavior of customers and a (single) clerk in a bakery waiting line. To help illustrate the various issues involved in trying to coordinate threads, develop the program based on the problem statement.	10
2.	Write a Software Phone App using Java Swing . The user enters the phone number it need to display in the number box and pushes the "CALL" button to start a phone call. Once the call is started, the label of the "CALL" button it will display a message box that call initiated and also changes to "HANG UP". When the user hangs up, the display is cleared. The user clicking the end button a message box to show the call termination message. When the user press clear button it need to deletes the last entered number.	10
3.	Create an ATM program for representing ATM transaction. In the ATM program, First the user needs to do login by setting the password with a condition that the following must be eight characters minimum out of which atleast one Capital letter, special character, numbers while typing the password it need to display as "*" instead of characters in the password field, then after verifying login the user has to select an option from the menu displayed on the screen. The options are related to withdraw the money, deposit the money, check the balance, and exit. Initially set a balance amount as Rs1,25,000/- To withdraw the money, we simply get the withdrawal amount from the user and remove that amount from the total balance and print the successful message.	10

	To deposit the money, we simply get the deposit amount from the user, add it to the total balance and print the successful message. To check balance, we simply print the total balance of the user. Display messagebox for each menu.	
4.	Write a program in Java named Copy to copy one file into another. The program should prompt the user for two file names, filename1 and filename2. Both filename1 and filename2 must exist or the program should throw a FileNotFoundException. Although filename1 must be the name of a file (not a directory), filename2 may be either a file or a directory. If filename2 is a file, then the program should copy filename1 to filename2. If filename2 is a directory, then the program should simply copy filename1 into filename2. That is, it should create a new file with the name filename1 inside the filename2 directory, copy the old file to the new file, and then delete the old file.	10
5.	Define a data-manipulation application for the books database. The user should be able to edit existing data and add new data to the database. Allow the user to edit the database in the following ways with statements supporting dynamic parameter: a) Add a new author. b) Edit the existing information for an author. c) Add a new title for an author. (Remember that the book must have an entry in the AuthorISBN table.). d) Add a new entry in the AuthorISBN table to link authors with titles. e) Return the resultset f) count number of updates performed in the Database	10

VIT
Vellore Institute of Technology

School of Information Technology and Engineering

Continuous Assessment Test - 1

Course Name & code: ITA5004- Object Oriented Programming using JAVA (Slot: A2)
Programme Name & Branch: MCA Class Number: VL2022230500239/0294/0268

Faculty Name: Prof. B K Ray/Prof. Thanga Mariappan L/Prof. Shynu P.G. Duration: 90 min Max Marks:50

Answer All (5 x 10 marks)

---

1) Create a Java program that sorts arrays *using method overloading*. The program should have overloaded methods named `sortArray()` that can handle the following array types:
Integer arrays: Pass an integer array, sort it in ascending order, and return the sorted array.
Double arrays: Pass a double array, sort it in ascending order, and return the sorted array.
String arrays: Pass a String array, sort it alphabetically, and return the sorted array.

Prompt the users to select the type of array they want to sort (e.g., 1 for Integer, 2 for Double, 3 for String). Ask the user to enter the number of elements in the array. Display the sorted array to the user after processing. Check if the user's choice for the type of array is valid (1, 2, or 3). If not, display an error message and prompt the user to re-enter their choice. Validate that the number of elements the user enters is a positive integer. If not, display an error message and prompt the user to re-enter the number of elements.

2) Create a Java program that simulates an online store's inventory management system. The system should include the following classes: Product, Category, and Inventory.
   - Product class: This class should have a product ID, name, price, and a Category object. Create a constructor that takes these parameters and initializes the class variables. Define a toString() method to display the product's information.
   - Category class: This class should have a category ID and a category name. Create a constructor that takes these parameters and initializes the class variables. Define a toString() method to display the category's information.
   - Inventory class: This class should have a list of Product objects. Implement the following methods:
     - addProduct(Product product): Adds a product to the inventory.
     - removeProduct(int productID): Removes a product from the inventory by its product ID.
     - updateProductPrice(int productID, double newPrice): Updates the price of a product by its product ID.
     - searchProductByCategory(Category category): Searches for products by their category and returns a list of matching product objects.
     - displayInventory(): Displays the entire inventory.

Demonstrate *passing and returning of objects*, focusing on the interaction between the Product, Category, and Inventory classes. For example, when adding a product to the inventory, pass a Product object to the addProduct() method. When searching for products by category, pass a Category object to the searchProductByCategory() method, which returns a list of Product objects.

3) Create a Java program that simulates a vehicle service management system. The program should demonstrate *method overriding and polymorphism using an inheritance hierarchy* of different vehicle types. Implement the following classes:

Vehicle: This class should have attributes such as vehicle ID, make, model, and manufacture year. Include methods to get and set the attributes and a toString() method to display the vehicle's information. Define an

abstract method `service()` that will be overridden in the subclasses. Car: This class should inherit from Vehicle. It should have additional attributes specific to cars, such as body type and number of doors. Override the `service()` method to display the service details, including a message like "Car service includes engine check, tire rotation, and brake inspection." **Motorcycle:** This class should inherit from Vehicle. It should have additional attributes specific to motorcycles, such as engine displacement and whether it has ABS. Override the `service()` method to display the service details, including a message, "Motorcycle service includes engine check, chain lubrication, and brake inspection." **Truck:** This class should inherit from Vehicle. It should have additional attributes specific to trucks, such as payload capacity and the number of axles. Override the `service()` method to display the service details, including a message like "Truck service includes engine check, tyre rotation, and suspension inspection."

Create a ServiceCentre class to manage the vehicles and their services. Implement the following methods:

- addVehicle(Vehicle vehicle): Adds a vehicle to the service centre.
- removeVehicle(int vehicleID): Removes a vehicle from the service centre by its vehicle ID.
- displayVehicles(): Displays all vehicles in the service centre.
- performService(int vehicleID): Performs the service for a vehicle by calling the service() method, which should display the appropriate service message based on the vehicle type.

4) Demonstrate *how to use abstract classes and interfaces* to model the scenario given in Q. No-3. Write down suitable assumptions required for the design and write the program with the explanation.
5) Implementing suitable *exception-handling* requirements ensures that the Vehicle Service Management System in Q. No-3 runs smoothly and provides a user-friendly experience. Implement the following three exception-handling requirements to ensure the program runs smoothly and handles potential errors: (1) **Invalid user input**: - Check for invalid user input when adding a new vehicle or performing other operations. If the input does not match the expected format or value range, throw a custom exception `InvalidInputException` with an appropriate error message. Catch the exception and prompt the user to re-enter the input. (2) **Vehicle not found**: - When attempting to remove a vehicle, perform a service, or display details for a specific vehicle, check if the vehicle with the given vehicle ID exists in the service centre. If not, throw a custom exception `VehicleNotFoundException` with an appropriate error message. Catch the exception and inform the user that the vehicle ID was not found. (3) **Duplicate vehicle ID**: - When adding a new vehicle to the service centre, check if a vehicle with the same vehicle ID already exists. If so, throw a custom exception `DuplicateVehicleIDException` with an appropriate error message. Catch the exception and ask the user to provide a unique vehicle ID.

Best Wishes!!!