VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of the UGC Act, 1956)

Reg. No. : 23MCA1030

Final Assessment Test (FAT) - May 2024

Programme	M.C.A.	Semester	WINTER SEMESTER 2023 - 24
Course Title	JAVA PROGRAMMING	Course Code	PMCA502L
Faculty Name	Prof. MADHESWARI	Slot	D2+TD2
		Class Nbr	CH2023240501379
Time	3 Hours	Max. Marks	100

General Instructions:

- Write only Register Number in the Question Paper where space is provided (right-side at the top) & do not write any other details.

Answer all questions (10 X 10 Marks = 100 Marks)

01. Design a Java program to manage employee records using an **Employee class** within a **package** named **company.records**. The Employee class should have attributes such as **name, employeeID,** and an **array salaries** to store monthly salaries for different months. Implement multiple constructors in the Employee class using the **'this'** keyword to initialize these attributes with different combinations of parameters. [10]

Include methods to:

- Display employee details.
- Calculate the average salary.
- Update the salary for a specific month.

Utilize the Scanner class for user input and ensure error handling for invalid input, such as non-numeric or negative salaries. Create an **EmployeeManager** class within the same package to demonstrate the functionalities of the Employee class.

02. Design a Java program for a library management system that utilizes inheritance, method overloading, and method overriding. Implement a superclass **'Item'** with attributes like **'title'**, **'itemID'**, and **'availableCopies'**. Include methods for **'checkout'**, **'return'**, and displaying **'item information'**. Extend this class to create subclasses **'Book'** and **'DVD'**, representing different types of library items. In the **'Book'** subclass, overload the **'checkout'** method to include an additional parameter for the due date. Override the **'checkout'** method in the **'DVD'** subclass to implement specific restriction checks, such as age restrictions for certain DVDs. Implement input validation to handle errors such as trying to check out an unavailable item, incorrect due date format (e.g., not in the format **'YYYY-MM-DD'**), and invalid item details (e.g., **empty title** or **invalid item ID format** - must be alphanumeric). [10]

Ensure that the format for item details is as follows:

- Title: A string containing at least one character.
- Item ID: Alphanumeric characters only, such as "B1234" or "X789YZ".
- Available Copies: An integer greater than or equal to zero.

Additionally, add the following requirements:

- Implement a final method in the **'Item'** class named **'displayItemInfo'** to display the information of the item.
- Implement an abstract method in the **'Item'** class named **'calculateLateFee'** to calculate the late fee for returning the item late.

03. Develop a Java program that demonstrates various scenarios of exception handling. Create a custom exception class named **'InvalidInputException'** to handle situations where the user provides incorrect input for a specific operation. Construct a class responsible for performing mathematical operations, such as addition and division, on user-provided numeric values. Implement handling for **'ArithmeticException'** if division by zero occurs during calculations. Define a class **'Customer'** with attributes **'name'** and **'balance'**, then attempt to access the **'balance'** attribute without initializing the object. Include handling for **'NullPointerException'** in this scenario. Write a class that interacts with an external database to retrieve user data, and handle **'SQLException'** if there are issues with the database connection or query execution. Implement a method to manipulate a collection, such as an ArrayList, and handle **'IndexOutOfBoundsException'** if attempting to access an element at an invalid index. **[10]**

04. Design a JavaFX program for a basic 'temperature converter' application. The application should feature a graphical user interface (GUI) with input fields for entering temperature in Celsius or Fahrenheit, and buttons to convert between the two units. Write code to perform temperature conversion calculations when the corresponding conversion button is clicked. **[10]**

Error Handling:

- Handle scenarios where users input non-numeric values in the temperature fields.
- Manage out-of-range temperatures, ensuring that temperatures entered are within reasonable bounds (e.g., -273.15°C to 1000°C or -459.67°F to 1832°F).

The GUI (Grid)layout should arrange the components in the following order:

Label: "Temperature Converter"

Text Field: for entering Celsius temperature

Text Field: for entering Fahrenheit temperature

Button: "Convert Celsius to Fahrenheit"

Button: "Convert Fahrenheit to Celsius"

Button: "Reset"

Feedback area to display conversion results or error messages

05. Write a Java program for a Shape Calculator's application. The application should have a graphical user interface (GUI) with input fields for parameters of geometric objects like **rectangles**, **circles**, and **triangles**. Provide buttons to determine the **area** and **perimeter** of each shape. Implement the abstract class **'Shape'** and its methods for computing area and perimeter. Extend this abstract class to include concrete classes for the geometric shapes **'Rectangle'**, **'Circle'**, and **'Triangle'**. Use the **final** keyword to prevent future sub classing of these concrete classes, guaranteeing that their implementations are preserved. Define an interface called **'Calculator'** with methods for computing area and perimeter, then implement it in each concrete form class. **[10]**

06. Develop a **JavaServer Pages (JSP)** application for an online movie database. Create a homepage **(index.jsp)** that welcomes users to the movie database and presents a list of available movies with details such as title, director, and release year. Implement a search form allowing users to search for movies by title, director, or genre, with dynamic handling to display matching results. Ensure that the homepage provides clear navigation options and an intuitive user interface. Use JSP to dynamically generate **HTML** content based on the movies available in the database. Implement a backend **servlet or controller** to handle user requests and interact with the database to fetch movie information. **[10]**

Consider the following aspects in your code:

- Design a suitable **database schema** to store movie details such as title, director, release year, and genre.
- Implement methods to retrieve movie data from the database and populate the homepage dynamically.
- Develop a **search functionality** that allows users to enter keywords for title, director, or genre, and display matching results dynamically.
- Handle errors and provide appropriate feedback to users in case of invalid input or database connectivity issues.

07. Design a web application for a simple **online bookstore** using the **Model-View-Controller (MVC)** architectural pattern. Describe how you would organize the application's components and outline the functionalities of each: **[10]**

- **Model:** Implement a model to represent books in the bookstore. Define the attributes each book should have, such as title, author, ISBN, and price.
- **View:** Create a view to display a list of available books in the bookstore. Users should be able to view details of each book, including title, author, and price.
- **Controller:** Develop a servlet to handle user requests and manage interactions between the model and view components. The servlet should retrieve book data from the model and render it in the view.
- **Servlet Functionality:** Implement servlet functionality to manage user interactions, such as displaying the list of books, adding books to a shopping cart, and processing checkout requests.

08. Design a menu-driven program that allows a user to perform various operations on a ArrayList of integers. The program should display a menu with the following options: **[10]**

- Add an integer to the list
- Remove an integer from the list
- Display the list of integers
- Calculate the sum of integers in the list
- Find the maximum integer in the list
- Exit the program

Implement the program using a do-while loop to repeatedly display the menu until the user chooses to exit. Use a switch-case statement to handle the user's menu choices.

09. Imagine you are developing a Java-based application for managing **student records** using **Hibernate**. Your task is to implement **CRUD operations** (Create, Read, Update, Delete) for the student records. [10]

Create: Implement functionality to add new student records to the database. Users should be able to input student details such as name, age, and grade, and the application should persist this information in the database.

Read: Develop functionality to retrieve student details from the database based on various criteria. Users should be able to search for students by name, age, or grade, and the application should display matching student records.

Update: Implement functionality to update existing student records in the database. Users should be able to modify student details such as name, age, or grade, and the application should reflect these changes in the database.

Delete: Develop functionality to delete student records from the database. Users should be able to select student records to delete based on their unique identifiers (e.g., student ID), and the application should remove these records from the database.

10. Design a Java program aimed at printing numbers alternately using two threads, each specialized for even and odd numbers. Create a superclass named **NumberPrinter**, which implements the **Runnable interface** and contains abstract methods for printing numbers. Derive subclasses **EvenNumberPrinter** and **OddNumberPrinter**, which prints even and odd numbers, respectively. Each subclass overrides the **printNumbers(int max)** method to fulfill its specific functionality. Write a **ThreadManager** class responsible for thread management, including the creation and synchronization of threads. Employ synchronization techniques like **locks or semaphores** to ensure threads print numbers alternately and prevent race conditions. [10]

⇔⇔⇔