SIT311 ADVANCED OBJECT-ORIENTED PROGRAMMING

COMPUTER LAB PRACTICE EXERCISES

(All questions are to be solved by writing a Java/C++ program)

WEEK 1: Class Design and Core API

1. Write a Java class Rectangle with fields for length and width. Include constructors, getters, setters, and a method to calculate area and perimeter.
2. Demonstrate method overloading by writing multiple printInfo() methods for different parameter lists.
3. Create a String utility program that counts the number of vowels in a given string.
4. Write a program that models a BankAccount class with methods for deposit, withdraw, and check balance.

WEEK 2: Inheritance and Polymorphism

1. Define a superclass Vehicle and subclasses Car and Truck. Override a method move() in each subclass.
2. Show how dynamic method dispatch (runtime polymorphism) works using a superclass reference pointing to subclass objects.
3. Write a program that demonstrates the use of super to access parent class members.
4. Implement a Shape class hierarchy with Circle and Rectangle subclasses, each overriding an area() method.

WEEK 3: Abstract Classes, Interfaces, and Nested Classes

1. Create an abstract class Employee with abstract method calculateSalary(). Extend it into Manager and Clerk with different implementations.
2. Define an interface Playable with method play(). Implement it in classes Guitar and Piano.
3. Write a program demonstrating the use of a static nested class.
4. Implement a local inner class inside a method that calculates the factorial of a number.

WEEK 4: Immutable Classes and Nested Classes

1. Write a Java class Student as an immutable class with fields id and name.
2. Create a program that uses a HashSet to check membership of elements.
3. Demonstrate the use of an anonymous inner class to handle a button click event.

4. Implement an immutable Book class with fields title and author. Show how it can be safely shared between threads.

WEEK 5: Enumerations and Lambda Expressions

1. Write an enum Day with constants for the days of the week. Demonstrate the use of values() and ordinal().
2. Create a lambda expression that takes two integers and returns their sum.
3. Demonstrate the use of Comparator with a lambda expression to sort a list of strings by length.
4. Implement an enum TrafficLight with constants RED, YELLOW, and GREEN. Write a program to display the current signal and its meaning.

WEEK 6 & 7: Exception Handling and Internationalization

1. Write a program that throws a custom exception when a negative number is entered.
2. Demonstrate the use of try-catch-finally in dividing two integers.
3. Write a program that uses ResourceBundle to display messages in both English and French.
4. Create a program that reads a number from the user, throws an exception if negative, and prints the number in US and France currency formats along with today's date in French.

WEEK 8: Collections and Generics

1. Write a program that stores student names in an ArrayList and displays them.
2. Demonstrate the use of HashSet to eliminate duplicate elements.
3. Implement a generic class Pair<K, V> to store key-value pairs.
4. Write a program that simulates a shopping cart using a HashMap<String, Double> where keys are item names and values are prices.

WEEK 9: Concurrency in Java

1. Write a program that creates two threads: one prints even numbers, the other prints odd numbers.
2. Demonstrate creating threads using both Thread class and Runnable interface.
3. Write a program that uses synchronization to prevent multiple threads from accessing a shared resource at the same time.

4. Implement a multithreaded program that simulates multiple users depositing and withdrawing money from a shared bank account.

WEEK 10: Input and Output in Java

1. Write a program that reads a text file and prints its contents line by line.
2. Demonstrate the use of FileWriter to write a list of strings into a file.
3. Implement object serialization and deserialization for a Student class.
4. Write a program that manages student records stored in a text file. It should allow adding new records, displaying all records, and searching by student ID.

WEEK 11: JavaFX and SceneBuilder

1. Create a JavaFX program that displays “Hello JavaFX” in a window.
2. Build a simple JavaFX form with fields for name and email, and a button to display the entered data.
3. Demonstrate event handling in JavaFX using both lambda expressions and anonymous inner classes.
4. Implement a JavaFX calculator application with basic arithmetic operations.

WEEK 12: Java Database Connectivity (JDBC)

1. Write a program to connect to a MySQL database using JDBC.
2. Demonstrate how to insert and retrieve records from a table using PreparedStatement.
3. Write a program that updates and deletes records in a student database.
4. Implement a full student management system using JDBC with options to insert, update, delete, and display student records.