**Assignment/Practical questions**

**Subject: Advance Java[b.sc csit/bit/bca]**

**Unit-I**

1. Array Declaration and Initialization: DONE

* Write a Java program that declares and initializes an array of integers. Print the elements of the array to the console.

1. Array Manipulation: DONE

* Create a Java program that takes an array of strings as input and prints the reverse of each string in the array.

1. Sum and Average: DONE

* Write a Java program that generates an array of random numbers (between 1 and 100), calculates the sum, and then computes the average of those numbers.

1. Finding Maximum and Minimum: DONE

* Develop a Java program that finds and prints the maximum and minimum values in an array of integers.

1. Sorting an Array:

* Write a Java program that sorts an array of integers in ascending order using the Bubble Sort algorithm. Print the sorted array.

1. Matrix Operations: DONE

* Design a Java program that performs matrix addition. Define two 2D arrays, add them, and print the result.

1. Two-Dimensional Array:

* Write a Java program that initializes and prints a two-dimensional array representing a multiplication table (up to a specified range).

1. Write a Java program that initializes an array of strings and uses a for-each loop to print each element of the array. DONE
2. Create a Java program that calculates and prints the sum of all elements in an array of integers using a for-each loop. Duplicate
3. Write a Java program that uses a for-each loop to copy elements from one array to another and prints the copied array. DONE
4. Develop a Java program that uses the enhanced for loop to iterate through a List of characters and prints each character.
5. Simple Class and Object: DONE

* Write a Java program that defines a class called Person with attributes such as name and age. Create an object of this class and print its details.

1. Multiple Objects: DONE

* Develop a Java program that defines a class named Car with attributes like model and year. Create multiple objects of the Car class and print their details.

1. Methods in Class: DONE

* Design a Java program that includes a class named Calculator with methods for addition, subtraction, multiplication, and division. Create an object of this class and perform arithmetic operations.

1. Constructor:3 DONE

* Write a Java program that defines a class Book with attributes like title and author. Include a parameterized constructor to initialize these attributes when creating an object.

1. Basic Overloading: DONE

* Write a Java program with a class that has two methods with the same name but different parameter types (e.g., one with an int parameter and another with a double parameter). Demonstrate calling both methods.

1. Overloading with Varargs:

* Create a Java program that defines a method using varargs and another method with a fixed number of parameters. Show how both methods can be called from the main method.

1. Overloading Constructors: DONE

* Design a Java class with multiple constructors. Demonstrate the concept of constructor overloading by creating objects using different constructors.

1. Overloading with Different Access Modifiers:

* Write a Java program that includes methods with different access modifiers (public, private, protected). Explain how method overloading works with varying access levels.

1. Overloading Static Methods:

* Develop a Java class with two static methods that have the same name but different parameters. Show how to call these methods without creating an instance of the class.

1. Overloading with Inheritance:

* Create a base class and a derived class. Implement overloaded methods in both classes. Demonstrate how method overloading works when dealing with inheritance.

1. Public Access Modifier:

* Write a Java program that includes a public class, a public method, and a public variable. Demonstrate how these public members can be accessed from another class in a different package.

1. Private Access Modifier:

* Create a Java program that uses a private variable and a private method within a class. Attempt to access these private members from outside the class and observe the results.

1. Protected Access Modifier and Inheritance:

* Design a Java program with a base class that has a protected method. Create a derived class in a different package and demonstrate how the protected method can be accessed within the subclass.

1. Default (Package-Private) Access Modifier:

* Write a Java program with two classes in the same package—one with a default access modifier and another with a public access modifier. Demonstrate how the default-access class can be accessed within the same package.

1. Access Modifiers in Constructors:

* Develop a Java program with a class that has a private constructor. Show how this class can be instantiated within the same class using a public method.

1. Basic Interface Implementation:

* Write a Java program that defines an interface named Drawable with a method draw(). Implement this interface in a class Circle that provides its own implementation of the draw() method.

1. Multiple Interface Implementation:

* Create a Java program with two interfaces: Shape and Color. Implement both interfaces in a class named ColoredCircle, providing appropriate implementations for the methods in each interface.

1. Interface Inheritance:

* Design a Java program that includes a base interface Vehicle with a method start(). Create a derived interface Car that extends Vehicle and adds a method accelerate(). Implement both interfaces in a class named SportsCar.

1. Static Inner Class:

* Write a Java program that includes a static inner class StaticInner within an outer class. Demonstrate how to create an instance of the static inner class from the main method.

1. The provided Java program, FinalExample, showcases various uses of the final keyword. The finalVariable is initialized with the value 42, serving as an immutable constant. The finalMethod is marked as final, emphasizing its unmodifiable nature, preventing overrides in subclasses. The FinalClass is declared as final, indicating its unextendable nature. The displayValues method utilizes final parameters x and message, illustrating the prohibition of modifications within the method. In the main method, the program accesses the final variable, calls the final method, creates an instance of the final class, and demonstrates the use of the displayValues method with final parameters, providing a comprehensive exploration of the final keyword in Java.
2. Static Variable in User Class:

Question: Create a User class with a static variable userCount to keep track of the total number of user instances. Implement methods to increment the count when a new user is created and display the total count. Demonstrate its usage in the main method.

1. Static Method in User Class:

Question: Extend the User class with a static method generateUsername that takes a user's first name and generates a unique username by appending the user count. Use this method to create usernames for several user instances in the main method.

1. Package Structure:

Question: Create a Java program with multiple classes organized into different packages. Demonstrate how to define a package, import classes from one package to another, and run the program.

1. Access Modifiers in Packages:

Question: Design a scenario with two packages, each containing a class. Use different access modifiers (public, private, default, protected) for the classes and demonstrate how they can be accessed or restricted across packages.

1. Simple Inheritance:

Question: Define a class Vehicle with properties like brand and year, and a method displayInfo that prints basic information about the vehicle. Create a subclass Car that inherits from Vehicle and adds specific properties like model and fuelType. Demonstrate inheritance by creating instances of both classes and invoking their methods.

1. Method Overriding:

Question: Extend the previous scenario by overriding the displayInfo method in the Car class to include information specific to cars. Show how method overriding allows the subclass to provide its own implementation of a method inherited from the superclass.

1. Basic Exception Handling:

Question: Write a Java program that attempts to perform a division operation inside a try block. Include a catch block to handle the ArithmeticException if the denominator is zero. Also, use a finally block to ensure that cleanup code (e.g., closing resources) is executed regardless of whether an exception occurred.

1. Multiple Catch Blocks:

Question: Extend the previous scenario by introducing multiple catch blocks to handle different types of exceptions. For example, catch both ArithmeticException and InputMismatchException exceptions separately.

1. Handling Checked Exceptions with Throws:

Task: Design a method that performs file I/O operations and declares the throws clause for checked exceptions like IOException. Demonstrate how to use this method and handle or propagate the declared exceptions in the calling code. Additionally, explore the use of the throw keyword within the method for custom exception scenarios.

1. Simple Custom Exception:

Question: Create a custom exception class named CustomException that extends the Exception class. Implement a simple constructor that takes a message as a parameter. In your main program, throw an instance of this custom exception and catch it in a catch block.

1. Suppose you are working on a multi-threaded Java application where threads are used to perform various tasks concurrently. The application has a critical section that should only be executed by one thread at a time. Explain how you would use thread states in Java to ensure proper synchronization and coordination between threads within the critical section. Provide code examples to illustrate your solution.
2. WAP to create thread using different ways in java.
3. Write a Java program that creates two threads. The first thread, named "ThreadA," should print odd numbers from 1 to 10 at intervals of 0.7 seconds, and the second thread, named "ThreadB," should print even numbers from 2 to 20 at intervals of 1.2 seconds. Ensure proper synchronization to avoid interference between the two threads. Provide the code for your program, including the main method.
4. WAP to execute the threads according to thread priorities in java.
5. Question:You are tasked with implementing a simple Java program that reads data from a binary file and performs some processing on it. Design a program using byte stream classes to achieve the following:

* File Reading:

1. Utilize FileInputStream to read data from a binary file named "inputData.dat."

* Data Processing:

1. Read the binary data and interpret it as a sequence of integers (4 bytes each).
2. Calculate the sum of these integers.

* Output:

1. Use FileOutputStream to write the calculated sum to an output file named "outputSum.dat."
2. Question:

You are tasked with creating a Java program that reads a text file, processes its content, and then writes the modified data to a new text file. Design a program using character stream classes to achieve the following:

1. File Reading:

* Utilize FileReader and BufferedReader to read lines from a text file named "inputText.txt."

1. Data Processing:

* For each line read, reverse the characters of the line.

1. File Writing:

* Use FileWriter and BufferedWriter to write the modified lines to an output file named "outputReversed.txt."

1. Question:

Suppose you are developing a Java application that manages a collection of student records using a Random Access File. Each student record has the following fields: student ID (int), name (String), and grade (double). The Random Access File is structured in fixed-length records.

* Design a method in Java to retrieve and display the details of a specific student given their student ID. Include error handling to address scenarios where the student ID is not found in the Random Access File. Additionally, discuss the potential challenges and considerations when implementing this method in terms of file positioning, record size, and data representation.[after database connection]

1. Question:You are working on a Java application that involves reading and writing objects to and from a file. Assume you have a class named Employee with attributes such as employeeID (int), name (String), and salary (double).

* Design a method to write an ArrayList of Employee objects to a binary file named "employees.dat".
* Create another method to read the ArrayList of Employee objects from "employees.dat" and display the details of employees whose salary is above a certain threshold (let's say $50,000).
* Discuss the importance of implementing the Serializable interface in the Employee class and explain how it facilitates the process of object serialization and deserialization.

1. Write a Java program to reverse the content of a text file using character streams. Create a method that takes the name of the input file and the name of the output file as parameters. The method should read the content of the input file, reverse each line, and write the reversed content to the output file. Include error handling to manage scenarios such as file not found or IOException during file operations.