**Day End Assignment Options**

|  |
| --- |
| Day 1:   1. Task 3: Synchronization and Inter-thread Communication Implement a producer-consumer problem using wait() and notify() methods to handle the correct processing sequence between threads. 2. Task 4: Synchronized Blocks and Methods Write a program that simulates a bank account being accessed by multiple threads to perform deposits and withdrawals using synchronized methods to prevent race conditions. 3. Task 5: Thread Pools and Concurrency Utilities Create a fixed-size thread pool and submit multiple tasks that perform complex calculations or I/O operations and observe the execution. 4. Task 6: Executors, Concurrent Collections, CompletableFuture Use an ExecutorService to parallelize a task that calculates prime numbers up to a given number and then use CompletableFuture to write the results to a file asynchronously. 5. Task 7: Writing Thread-Safe Code, Immutable Objects Design a thread-safe Counter class with increment and decrement methods. Then demonstrate its usage from multiple threads. Also, implement and use an immutable class to share data between threads.  Day 1: 6. Task 1: Generics and Type Safety Create a generic Pair class that holds two objects of different types, and write a method to return a reversed version of the pair. 7. Task 2: Generic Classes and Methods Implement a generic method that swaps the positions of two elements in an array, regardless of their type, and demonstrate its usage with different object types. 8. Task 3: Reflection API  Use reflection to inspect a class's methods, fields, and constructors, and modify the access level of a private field, setting its value during runtime 9. Task 4: Lambda Expressions Implement a Comparator for a Person class using a lambda expression, and sort a list of Person objects by their age. 10. Task 5: Functional Interfaces   Create a method that accepts functions as parameters using Predicate, Function, Consumer, and Supplier interfaces to operate on a Person object.  Day 2 and 3:   1. Task 1: Java IO Basics   Write a program that reads a text file and counts the frequency of each word using FileReader and FileWriter.   1. Task 2: Serialization and Deserialization   Serialize a custom object to a file and then deserialize it back to recover the object state.   1. Task 3: New IO (NIO)   Use NIO Channels and Buffers to read content from a file and write to another file.   1. Task 4: Java Networking   Write a simple HTTP client that connects to a URL, sends a request, and displays the response headers and body.   1. Task 5: Java Networking and Serialization   Develop a basic TCP client and server application where the client sends a serialized object with 2 numbers and operation to be performed on them to the server, and the server computes the result and sends it back to the client. for eg, we could send 2, 2, "+" which would mean 2 + 2   1. Task 6: Java 8 Date and Time API   Write a program that calculates the number of days between two dates input by the user.   1. Task 7: Timezone   Create a timezone converter that takes a time in one timezone and converts it to another timezone.  Day 4:   1. Task 1: Establishing Database Connections   Write a Java program that connects to a SQLite database and prints out the connection object to confirm successful connection.   1. Task 2: SQL Queries using JDBC   Create a table 'User' with a following schema 'User ID' and 'Password' stored as hash format (note you have research on how to generate hash from a string), accept "User ID" and "Password" as input and check in the table if they match to confirm whether user access is allowed or not.   1. Task 3: PreparedStatement   Modify the SELECT query program to use PreparedStatement to parameterize the query and prevent SQL injection.   1. Task 1: Write a set of JUnit tests for a given class with simple mathematical operations (add, subtract, multiply, divide) using the basic @Test annotation. 2. Task 2: Extend the above JUnit tests to use @Before, @After, @BeforeClass, and @AfterClass annotations to manage test setup and teardown. 3. Task 3: Create test cases with assertEquals, assertTrue, and assertFalse to validate the correctness of a custom String utility class. 4. Task 1: Research and present a comparison of different garbage collection algorithms (Serial, Parallel, CMS, G1, ZGC) in Java. 5. Task 1: Singleton   Implement a Singleton class that manages database connections. Ensure the class adheres strictly to the singleton pattern principles.   1. Task 2: Factory Method   Create a ShapeFactory class that encapsulates the object creation logic of different Shape objects like Circle, Square, and Rectangle.   1. Task 3: Proxy   Create a proxy class for accessing a sensitive object that contains a secret key. The proxy should only allow access to the secret key if a correct password is provided.   1. Task 4: Strategy   Develop a Context class that can use different SortingStrategy algorithms interchangeably to sort a collection of numbers |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |