1. Write a program to print "Hello, Java!" to the console. [Low]
2. Write a program to declare an integer variable age and initialize it with your current age. Then, print its value. [Low]
3. Write a program to declare a double variable for temperature and a boolean variable for isRaining. Assign appropriate values and print them. [Low]
4. Write a program to take your name as a string input from the console and print a personalized greeting message. [Low]
5. Write a program that takes two integer numbers as input, calculates their sum, and prints the result. [Low]
6. Write a program to convert a given temperature from Celsius to Fahrenheit. Take Celsius as input. Formula: F=C∗(9.0/5.0)+32. [Low]
7. Write a program to demonstrate the use of arithmetic operators (+, -, \*, /, %). Take two numbers as input and show the result of each operation. [Low]
8. Write a program to demonstrate the use of relational operators (==, !=, <, >, <=, >=). Compare two numbers and print the boolean result for each operator. [Low]
9. Write a program to demonstrate the use of logical operators (&&, ||, !). Use boolean variables to show their behaviour. [Low]
10. Write a program to find the largest among three numbers using if-else if-else statements. [Low]
11. Write a program to check if a given year is a leap year. (A year is a leap year if it is divisible by 4, but not by 100, unless it is also divisible by 400). [Medium]
12. Write a program that uses a switch statement to print the day of the week (e.g., "Monday") given a number (1 for Monday, etc.). Handle invalid input. [Low]
13. Write a program to print numbers from 1 to 10 using a for loop. [Low]
14. Write a program to calculate the sum of natural numbers from 1 to N (user input) using a while loop. [Low]
15. Write a program to reverse a given integer number (e.g., 123 becomes 321) using a while or do-while loop. [Medium]
16. Write a program to check if a number is prime using loops. [Medium]
17. Write a program to print the multiplication table of a number entered by the user. [Low]
18. Write a program to generate the Fibonacci series up to N terms. [Medium]
19. Write a program to calculate the factorial of a number using a loop. [Low]
20. Write a program that takes an array of 5 integers as input, stores them, and then prints the sum of all elements in the array. [Low]
21. Write a program to find the maximum element in an array of integers. [Medium]
22. Write a program to sort an array of integers using the Bubble Sort algorithm. [Medium]
23. Write a program to count the number of vowels and consonants in a given string. [Medium]
24. Write a program to demonstrate explicit and implicit type casting. Convert an int to double implicitly and a double to int explicitly, printing values at each step. [Low]
25. Write a program to swap the values of two integer variables without using a third variable. [Medium]

**Module 2: Object-Oriented Programming in Java**

1. Design a Car class with make, model, and year as instance variables. Create an object of Car, set its attributes, and print them. [Low]
2. Enhance the Car class (Q26) by adding a default constructor and a parameterized constructor that initializes all three instance variables. Create objects using both constructors. [Low]
3. Add a method displayInfo() to the Car class (Q27) that prints all the car's details. Call this method from your main program. [Low]
4. Implement encapsulation for the Car class (Q28) by making instance variables private and providing public getter and setter methods for them. [Medium]
5. Design a Person class with name and age. Create a Student class that inherits from Person and adds studentId and major. Demonstrate inheritance by creating Student objects. [Medium]
6. Override the displayInfo() method in the Student class (Q30) to include student-specific details along with person details. [Medium]
7. Demonstrate polymorphism by creating an array of Person references, where some refer to Person objects and others to Student objects. Call displayInfo() on each. [High]
8. Design an Animal abstract class with an abstract method makeSound(). Create concrete Dog and Cat classes that extend Animal and implement makeSound(). Demonstrate polymorphism. [Medium]
9. Define an interface Shape with abstract methods getArea() and getPerimeter(). Implement Circle and Rectangle classes that implement the Shape interface. [Medium]
10. Create an interface Flyable with a method fly(). Make a Bird class implement Flyable and a Plane class implement Flyable. Demonstrate interface usage. [Medium]
11. Discuss method overloading. Write a class with overloaded methods (e.g., add(int a, int b) and add(double a, double b)). [Low]
12. Discuss method overriding. Provide an example where a subclass overrides a method from its superclass. [Low]
13. Create a class Counter with a static variable count initialized to 0. Increment count in the constructor. Demonstrate that count keeps track of the number of objects created. [Medium]
14. Write a program to demonstrate the use of the super keyword to call a superclass constructor and a superclass method. [Medium]
15. Implement a class hierarchy: Vehicle -> TwoWheeler -> Motorcycle. Add appropriate methods and fields to each, demonstrating inheritance. [Medium]
16. Design an abstract class Employee with name, id, and an abstract method calculateSalary(). Create FullTimeEmployee and PartTimeEmployee subclasses, each with their own salary calculation logic. [High]
17. Create an interface Comparable with a method compareTo(Object o). Implement this interface in a Product class to enable sorting products based on their price. [High]
18. Discuss the difference between interface and abstract class in terms of multiple inheritance, fields, and constructors. (Conceptual, in comments). [Medium]
19. Write a program to demonstrate final keywords with variables, methods, and classes. Explain their purpose. [Medium]
20. Implement a singleton design pattern for a Logger class, ensuring only one instance of the class can be created. [High]
21. Design a class Rectangle with methods to calculate area and perimeter. Add a constructor to initialize dimensions. [Low]
22. Create a Manager class that extends Employee (from Q41). Add a bonus field and override calculateSalary() to include the bonus. [High]
23. Write a program that uses an anonymous inner class to implement an interface. [High]
24. Demonstrate static polymorphism (method overloading) and dynamic polymorphism (method overriding) in a single program with clear examples. [High]
25. Design a simple Bank class with methods deposit and withdraw that modify an account balance. Handle invalid operations (e.g., withdrawing more than balance). [Medium]

**Module 3: Java Collections Framework**

1. Write a program to create an ArrayList of strings, add 5 names to it, and print all elements using an enhanced for loop. [Low]
2. Write a program to demonstrate common ArrayList operations: add, remove, get, size, contains, indexOf. [Low]
3. Write a program to create a LinkedList of integers, add elements, and demonstrate addFirst(), addLast(), removeFirst(), removeLast() operations. [Medium]
4. Compare and contrast ArrayList and LinkedList in terms of internal implementation, performance for random access vs. insertions/deletions, and memory usage. (Conceptual, in comments). [Medium]
5. Write a program to create a HashSet of integers, add duplicate elements, and observe that duplicates are not stored. Print the set. [Low]
6. Write a program to demonstrate common HashSet operations: add, remove, contains, size. [Low]
7. Compare and contrast HashSet and LinkedHashSet and TreeSet. Demonstrate their behavior with a few elements. [Medium]
8. Write a program to create a HashMap where keys are String (e.g., student names) and values are Integer (e.g., scores). Add a few entries, retrieve a value by key, and iterate through all key-value pairs. [Medium]
9. Write a program to demonstrate common HashMap operations: put, get, containsKey, remove, size. [Medium]
10. Compare and contrast HashMap, LinkedHashMap, and TreeMap. Demonstrate their behavior with a few entries. [High]
11. Write a program to sort an ArrayList of integers in ascending and descending order using Collections.sort(). [Low]
12. Write a program to sort an ArrayList of custom Student objects (with name, rollNo) by rollNo using the Comparable interface. [Medium]
13. Write a program to sort an ArrayList of custom Product objects (with name, price) by price using a Comparator interface. [High]
14. Write a program to search for an element in an ArrayList using Collections.binarySearch() (ensure the list is sorted first). [Medium]
15. Write a program that uses an Iterator to traverse and remove elements from an ArrayList while iterating. [Medium]
16. Write a program to convert an ArrayList to an array and vice-versa. [Low]
17. Write a program to remove all occurrences of a specific element from an ArrayList. [Medium]
18. Write a program to find the frequency of each word in a given text using a HashMap. [High]
19. Write a program to implement a simple Stack using java.util.Stack and demonstrate its push, pop, peek, isEmpty operations. [Low]
20. Write a program to implement a simple Queue using java.util.LinkedList (as a Deque or Queue interface) and demonstrate its offer, poll, peek, isEmpty operations. [Low]
21. Write a program to count the number of unique words in a given text file using a Set. [High]
22. Write a program to check if two ArrayLists are equal (contain the same elements in the same order). [Medium]
23. Implement a program that reverses the order of elements in a LinkedList without using Collections.reverse(). [Medium]
24. Write a program to find the intersection and union of two Sets. [Medium]
25. Write a program to group objects by an attribute using a Map (e.g., group Student objects by major). [High]

**Module 4: Exception Handling and I/O**

1. Write a program that attempts to divide two numbers, where the divisor can be zero. Use a try-catch block to handle the ArithmeticException. [Low]
2. Write a program that accesses an array element at an invalid index. Use a try-catch block to handle the ArrayIndexOutOfBoundsException. [Low]
3. Write a program that converts a non-numeric string to an integer. Use a try-catch block to handle the NumberFormatException. [Low]
4. Demonstrate the use of multiple catch blocks to handle different types of exceptions in a single try block. [Medium]
5. Write a program that uses a finally block to ensure a resource (e.g., a file stream) is closed, regardless of whether an exception occurred. [Medium]
6. Write a program that defines a custom MyCustomException (a checked exception) and throws it under a specific condition. Handle this exception in the calling method. [High]
7. Write a program that demonstrates the throws keyword in a method signature to declare that a method might throw a checked exception. [Medium]
8. Write a program that creates a text file named "output.txt" and writes "This is some text." into it using FileWriter. [Low]
9. Write a program to read the entire content of "output.txt" (created in Q83) line by line using FileReader and BufferedReader, and print it to the console. [Low]
10. Write a program to copy the content of one text file ("source.txt") to another text file ("destination.txt") using FileReader and FileWriter. [Medium]
11. Write a program that appends new text to an existing file without overwriting its content. [Medium]
12. Write a program that checks if a file or directory exists at a given path using java.io.File. [Low]
13. Write a program to list all files and subdirectories within a given directory. [Medium]
14. Implement a program to count the number of words in a text file. [Medium]
15. Write a program to count the number of lines in a text file. [Medium]
16. Write a program to demonstrate object serialization. Create a simple Student class (must implement Serializable), create an object, serialize it to a file, and then deserialize it back into an object. [High]
17. Write a program that reads primitive data types (int, double, boolean) from a file using DataInputStream and writes them using DataOutputStream. [High]
18. Write a program to read characters from console input using BufferedReader and write them to a file. [Medium]
19. Implement a program that processes command-line arguments and handles potential NumberFormatException if an argument is expected to be a number but isn't. [Medium]
20. Write a program to demonstrate the try-with-resources statement for automatic resource management (e.g., BufferedReader, FileWriter). [High]
21. Discuss the difference between checked and unchecked exceptions, providing examples of when each should be used. (Conceptual, in comments). [Medium]
22. Write a program that uses System.err.println() to print error messages to the standard error stream. [Low]
23. Write a program to delete a specific file if it exists. [Low]
24. Implement a program to read and parse comma-separated values (CSV) from a file. [High]
25. Write a program that uses PrintWriter to write formatted output to a file. [Medium]

**Module 5: Multithreading and Concurrency**

1. Write a program to create a new thread by extending the Thread class. The thread should print a message multiple times. [Low]
2. Write a program to create a new thread by implementing the Runnable interface. The thread should print a message multiple times. [Low]
3. Compare and contrast creating threads using Thread class vs. Runnable interface. Discuss which is generally preferred and why. (Conceptual, in comments). [Medium]
4. Write a program to demonstrate the sleep() method. Create a thread that pauses for a few seconds. [Low]
5. Write a program to demonstrate join() method. Create two threads, where the main thread waits for the completion of another thread before continuing. [Medium]
6. Write a program to simulate a race condition where multiple threads try to increment a shared counter without synchronization. Observe the incorrect results. [Medium]
7. Solve the race condition problem (Q106) using the synchronized keyword on a method. [Medium]
8. Solve the race condition problem (Q106) using the synchronized block. [Medium]
9. Write a program to demonstrate wait(), notify(), and notifyAll() for inter-thread communication (e.g., Producer-Consumer problem with a single item buffer). [High]
10. Implement a simple Producer-Consumer problem using wait() and notifyAll() with a bounded buffer. [Very High]
11. Write a program that uses ReentrantLock from java.util.concurrent.locks for explicit locking instead of synchronized keyword. [High]
12. Write a program that uses Semaphore from java.util.concurrent to control access to a limited number of resources. [High]
13. Implement a ThreadPoolExecutor to manage a pool of threads for executing a set of tasks. [High]
14. Write a program to demonstrate the use of Callable and Future interfaces for tasks that return a result and/or throw an exception. [High]
15. Write a program to use ExecutorService to execute a fixed number of tasks concurrently. [Medium]
16. Write a program to count the occurrences of words in a large text file using multiple threads to process different parts of the file concurrently. [Very High]
17. Implement a program that uses CountDownLatch to make one or more threads wait until a set of operations being performed in other threads completes. [High]
18. Implement a program that uses CyclicBarrier to synchronize multiple threads at a common barrier point. [High]
19. Discuss thread states (New, Runnable, Running, Blocked, Waiting, Timed Waiting, Terminated) in Java. (Conceptual, in comments). [Medium]
20. Explain the concept of deadlocks in multithreading and provide a simple code example that could lead to a deadlock. [High]
21. Write a program that uses AtomicInteger from java.util.concurrent.atomic to safely increment a counter without explicit locking. [Medium]
22. Implement a simple chat application server and client where the server handles multiple clients using separate threads. (Basic networking concepts apply here too). [Very High]
23. Write a program to demonstrate the use of ThreadLocal variables. [High]
24. Implement a program to sort an array of integers using a parallel merge sort approach (divide array, sort sub-arrays in separate threads, then merge). [Very High]
25. Write a program that implements a simple ReadWriteLock to allow multiple readers but only one writer at a time. [Very High]

**Module 6: Advanced Java Basics**

1. Write a program to demonstrate Generics by creating a generic class Box<T> that can hold any type of object. Create Box<Integer> and Box<String> objects. [Medium]
2. Write a program to create a generic method printArray(T[] array) that can print elements of any type of array. [Medium]
3. Explain the concept of type erasure in Java Generics. (Conceptual, in comments). [High]
4. Write a program to demonstrate a simple Lambda Expression for implementing a functional interface (e.g., Runnable or a custom interface with a single abstract method). [Low]
5. Write a program that uses Lambda Expressions with Collections.sort() to sort a list of strings by length. [Medium]
6. Write a program that uses a Lambda Expression with forEach loop on a List or Set. [Low]
7. Write a program that uses the Stream API to filter a list of integers and collect only even numbers into a new list. [Medium]
8. Write a program that uses the Stream API to map a list of strings to their uppercase versions and collect them. [Medium]
9. Write a program that uses the Stream API to perform a reduction operation (e.g., sum of all elements in a list). [Medium]
10. Write a program that uses the Stream API to find the maximum element in a list of integers. [Medium]
11. Write a program that uses the Stream API to count the occurrences of each word in a list of strings. [High]
12. Create a custom Annotation @MyAnnotation with a single element. Apply it to a class and a method. [Medium]
13. Write a program that uses Java Reflection to inspect a class at runtime. Get its methods, fields, and constructors. [High]
14. Write a program that uses Java Reflection to create an instance of a class and invoke a method dynamically. [High]
15. Set up a simple database (e.g., H2, SQLite, or MySQL local instance). Write a program to connect to the database using JDBC. [Medium]
16. Write a program using JDBC to create a new table (e.g., Students table with id, name, age). [Medium]
17. Write a program using JDBC to insert a few records into the Students table (Q141). [Medium]
18. Write a program using JDBC to retrieve and display all records from the Students table (Q141). [Medium]
19. Write a program using JDBC to update a record in the Students table based on id. [Medium]
20. Write a program using JDBC to delete a record from the Students table based on id. [Medium]
21. Write a program using JDBC prepared statements to prevent SQL injection vulnerabilities when inserting data. [High]
22. Write a program that demonstrates basic Optional usage in Java 8 to handle potential null values gracefully. [Medium]
23. Write a program that uses the Stream API with Collectors.groupingBy to group objects. [High]
24. Implement a custom annotation that is retained at runtime and read its value using Reflection. [High]
25. Write a program to perform a simple join operation between two tables using JDBC. [High]