Part – A (2-Marks)

1. What is an exception in Java?
2. What is the difference between checked and unchecked exceptions?
3. What happens when no catch block matches the thrown exception?
4. What is the purpose of multiple catch clauses?
5. Can we write a finally block without a catch block?
6. What is multithreading?
7. How do you create a thread in Java?
8. What is the use of Thread.sleep() method?
9. What is synchronization in Java?
10. What is the role of thread priorities?
11. What is Autoboxing in Java?
12. Name any two built-in Java exceptions.
13. What is a user-defined exception?

Part – B (13 / 14 Marks)

1. Explain Java Exception Handling mechanism with a program using try, catch, finally.
2. Write a Java program with **multiple catch blocks** and explain the flow.
3. Explain **nested try blocks** with an example.
4. Describe **Java's built-in exceptions** with at least 5 examples.
5. Write a Java program to create and throw a **user-defined exception**.
6. Explain Java's **thread model**. How do you create threads using Thread and Runnable?
7. Write a program to create **two threads** that run simultaneously using Runnable interface.

8. What is **thread synchronization**? Write a Java program showing a shared resource problem.
9. Write a program that uses **inter-thread communication** using wait(), notify().
10. Explain thread methods to **suspend**, **resume**, and **stop** threads with example.
11. What is **Autoboxing and Unboxing**? Explain with a Java program.
12. How are **wrapper classes** used in Java? Write a short program using Integer, Double.
13. Compare **thread priorities** in Java. Write a program that sets different priorities to threads.
14. Write a Java program to create and throw a user-defined exception.

Case Study-Based Questions (Simple & Scenario-Driven) . When answering these:

- **Start with a brief explanation of the scenario**
- **Write clean, commented code**
- **Highlight where you're applying Java features (e.g., try-catch, synchronized, etc.)**

Exception Handling

Case Study 1: ATM Withdrawal

Scenario:
You are developing an ATM system. If a user tries to withdraw more money than the balance, the system should throw a custom exception called InsufficientFundsException.

Question:
Write a Java program using a user-defined exception, try-catch, and finally block to handle this case.

Case Study 2: Student Grade System

Scenario:
A system takes marks as input. If the entered marks are negative or above 100, it should throw an appropriate built-in Java exception.

Question:
Create a Java program that demonstrates the use of multiple catch clauses to handle invalid input types and invalid mark ranges.

Case Study 3: File Reader

Scenario:
You're reading a file in Java. Show how nested try-catch can be used to handle both FileNotFoundException and IOException.

Question:
Write a Java program using nested try blocks and explain how exception handling works in nested structure.

Multithreading

Case Study 4: Printing Even and Odd Numbers

Scenario:
Design a program where two threads print numbers:

- One prints even numbers
- The other prints odd numbers
  They should synchronize so that the numbers appear in proper order.

Question:
Create a Java program using threads, synchronization, and proper method locking to implement this.

Case Study 5: Bank Account Access

Scenario:
Two threads are trying to access the same bank account to withdraw money. Demonstrate how using or not using synchronization affects the output.

Question:
Write a Java program that shows thread interference and then fix it using synchronized.

Case Study 6: Chat App Simulation

Scenario:
Simulate a basic two-way chat where Thread A types a message, and Thread B responds only after receiving the message. Use wait() and notify() for communication.

Question:
Develop a Java program showing inter-thread communication using wait() and notify().

Case Study 7: Resume and Stop Threads

Scenario:
A music app uses a thread to play a song. You want to simulate pause, resume, and stop functionalities.

Question:
Create a Java program where a thread can be paused, resumed, and stopped (simulate using flags and sleep).

Case Study Question 8: Online Ticket Booking System

Scenario:
You are developing an online ticket booking system. The system has a limited number of tickets. Multiple users (threads) are trying to book tickets at the same time.

Without proper synchronization, the same ticket may be booked more than once (overbooking issue).

Question:
Write a Java program that:

1. Creates a shared ticket counter with a fixed number of tickets (e.g., 5).
2. Uses multiple threads to simulate users trying to book tickets.
3. Uses synchronized methods to ensure that no two users can book the same ticket.

Wrapper Classes & Autoboxing

Case Study 9: Data Entry System

Scenario:
In a student marks entry system, input is taken as int, but stored in an ArrayList that only accepts Integer objects.

Question:
Write a program that uses Autoboxing to automatically convert int to Integer and store values in a list.

Case Study 10: Wrapper Operations

Scenario:
Create a program that calculates the average of 5 numbers using wrapper classes (Integer, Double). Show autoboxing and unboxing.

Question:
Demonstrate how wrapper classes are used in calculation and how Java converts between primitive and object types.