1. Assess the Existing COBOL Codebase:

• Analyze the business logic, structure, and modules of the COBOL application.

• Identify which components are critical and must be preserved during the migration.

2. Define Target Architecture:

• Design the architecture for the Java application, ensuring it aligns with modern standards like Object-Oriented principles.

• Consider frameworks, databases, and middleware you will use in the new Java environment.

3. Data Mapping:

• Map COBOL data structures (like records, PIC formats) to Java equivalents (like classes, objects, and data types).

• Handle files and databases properly, converting COBOL file handling routines to Java’s I/O and JDBC for database interactions.

4. Automated Code Conversion (Optional):

• Use automated tools that convert COBOL to Java to accelerate the process, though manual review and refactoring will be required afterward.

• Some tools include:

• Micro Focus Enterprise Developer

• Fujitsu NetCOBOL for Java

5. Manual Refactoring and Enhancement:

• Manually refactor the converted code, replacing COBOL constructs that do not map cleanly into Java (e.g., GOTO statements) with Java equivalents.

• Implement Object-Oriented Design (OOD) principles, breaking down monolithic COBOL programs into Java classes and methods.

6. Preserve Business Logic:

• Ensure that key business rules encoded in COBOL are preserved in the Java application.

• Unit test each piece of logic to validate correctness after migration.

7. Testing:

• Conduct extensive testing (unit, integration, and user acceptance tests) to ensure the Java application works as expected.

• Compare the outputs of the Java application with the COBOL application to confirm accuracy.

8. Deployment and Monitoring:

• Deploy the Java application to the target environment (on-premises or cloud).

• Monitor performance, troubleshoot any issues, and optimize as necessary.