## **Instructions**

In this project, you will build a simple Account Management System using Object-Oriented Programming (OOP) principles in Java. The project is divided into three sections, each focusing on different aspects of OOP, such as inheritance, polymorphism, and encapsulation.

### **Section 1: Basic Account Implementation**

You are provided with a base Account class that supports basic operations like debit, withdraw, and deposit. Your task is to complete the implementation of a CreditAccount class that inherits from the Account class. In addition to the existing functionality, the CreditAccount should:

* Implement a credit limit that restricts the amount that can be withdrawn.
* Override the withdraw and deposit methods to accommodate the credit limit.

### **Section 2: Credit Account Enhancement**

After successfully implementing the CreditAccount, you will enhance it further by:

* Ensuring that the credit limit is not exceeded during withdrawals.
* Implementing logic to handle scenarios where deposits bring the account back within the credit limit.

This section will test your understanding of overriding methods and managing state within an object.

### **Section 3: Loan Account Implementation**

In the final section, you will create a LoanAccount class that extends the basic Account class. This account will track a loan balance and include the following features:

* An interest rate, which will be applied to the outstanding balance.
* The ability to track and split payments into principal and interest portions.
* Methods to calculate and track the total payments made towards both principal and interest.

The LoanAccount should provide functionality to:

* Calculate interest on the outstanding balance.
* Apply payments, first to interest and then to the principal.
* Keep a record of how much has been paid towards the interest, the principal, and the total amount paid.

This final section will challenge you to think about how to extend existing classes to introduce new functionality while maintaining clean and maintainable code.

### **Objectives:**

* Apply OOP concepts such as inheritance, method overriding, and encapsulation.
* Understand how to design classes that build upon and extend the functionality of existing classes.
* Practice implementing real-world financial concepts in a programming context.