**Write a Java Program with all the JUNIT test cases. TDD approach will be preferred.**

* **Framework: Spring boot, build tool : maven or gradle**
* **REST APIs need to be exposed like below:**
* **Get All Loans (/loans)**
* **Add loan (/loans/add)**
* **get loan by loan id (/loans/{loanId})**
* **get loan by customer id (/loans/{customerId)**
* **get loans by lenderId (/loans/{lenderId})**
* **get aggregate loans by Lender (**aggregate remaining amount, Interest and Penalty**)(/loans/aggregate/lender)**
* **get aggregate loans by customer id (**aggregate remaining amount, Interest and Penalty**)(/loans/aggregate/customer)**
* **get the aggregate loans by interest (**aggregate remaining amount, Interest and Penalty**)(/loans/aggregate/interest)  
    
  do the exception handling as well.**

**Problem Statement**

There is a scenario where thousands of loans are flowing into one store, assume any way of transmission of Loans. We need to create a one loan store, which store the loans in the following order

| Loan ID | Customer Id | Lender Id | Amount | Remaining Amount | Payment Date | Interest Per Day(%) | Due Date | Penalty/Day  (%Day) | Cancel |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| L1 | C1 | LEN1 | 10000 | 10000 | 05/06/2023 | 1 | 05/07/2023 | 0.01% |  |
| L2 | C1 | LEN1 | 20000 | 5000 | 01/06/2023 | 1 | 05/08/2023 | 0.01% |  |
| L3 | C2 | LEN2 | 50000 | 30000 | 04/04/2023 | 2 | 04/05/2023 | 0.02 % |  |
| L4 | C3 | LEN2 | 50000 | 30000 | 04/04/2023 | 2 | 04/05/2023 | 0.02 % |  |

***There are couple of requirement/validation***

1. The payment date can’t be greater than the Due Date. If its greater we have to reject the Loan and thrown the exception
2. We need to write an aggregation on the remaining amount, Interest and Penalty Group by Lender, Group by Interest and Group by Customer ID.
3. If the Loan crosses the due date, it should write an alert in the log message.

**FAQ’s**

**Can I use build management, is it mandatory?**

Build management tool is not mandatory, but preference is that you should use any one of the build managements (Gradle or maven). This helps to build the code offline on interviewers’ machine, without worrying about any dependencies

**How can I share the code with interviewer?**

In case you are going through offline code pairing session (asynchronous) i.e., you get the assignment from resourcing team 2 days before, then preferred option is to commit the code in your GitHub repository. Make the repository public in read only mode and share it with resourcing team. Interviewer will offline asses the code at the time of discussion will discuss with you.

**Can I use open-source framework(spring-boot) etc?**

Yes, that will be preferred if you can use spring-boot, but principally the concept of Open Close principle should be extremely clear.

**Do I really need to follow TDD?**

The preferred option is to follow TDD. Your code should evolve with Red-Green Refactor concept. In case you have not done TDD in past but writing test cases are mandatory in this problem statement. What ever the code you are writing should be properly covered with the Unit Test Cases or Integrated Test cases. Interview will be first checking the test cases written.

**Can I use cany database or cache in assignment?**

You are open to use any database or cache, but remember it’s a test assignment not a project. Keeping technology stack simple will be helpful to build the code and will be self-explanatory.