TP à Rendre :

Class Agence :

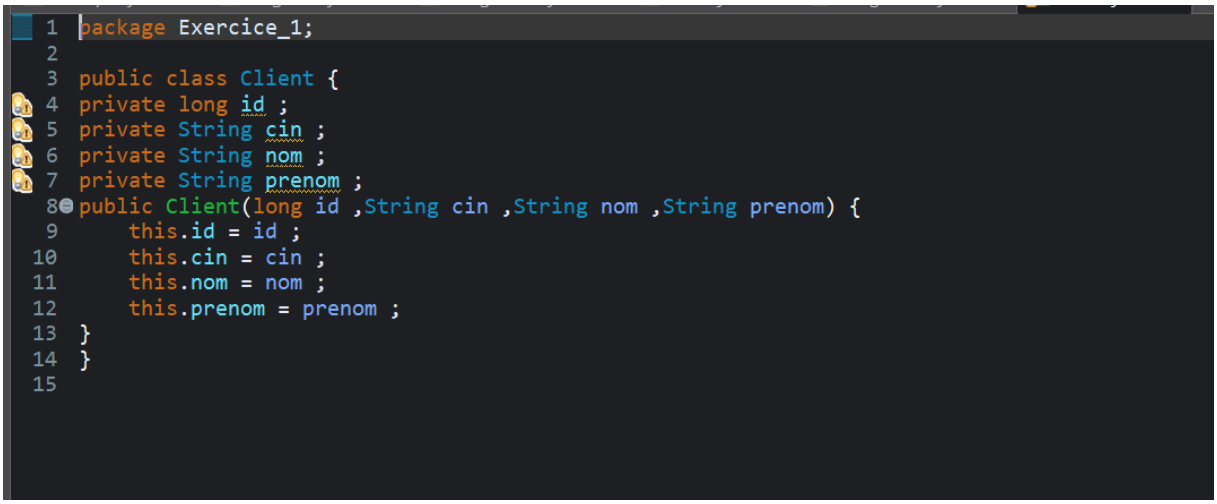
```java
package Exercice_1;
import java.time.LocalDate;

public class Agence implements IAgence{
Map<String , Compte>comptes = new HashMap<>();
Agence(){
Compte c1 = new Compte(123,"A123",LocalDate.now());
comptes.put(c1.getNumero(), c1) ;
Compte c2 = new Compte(124,"A124",LocalDate.now());
comptes.put(c2.getNumero(), c2) ;
Compte c3 = new Compte(125,"A124",LocalDate.now());
comptes.put(c3.getNumero(), c3) ;
Compte c4 = new Compte(126,"A125",LocalDate.now());
comptes.put(c4.getNumero(), c4);
}
@Override
public void addCompte(Compte c) {
    // TODO Auto-generated method stub
    comptes.put(c.getNumero(), c);
}
@Override
public Compte getCompte(String numero) {
    // TODO Auto-generated method stub
    return comptes.get(numero);
}
}

```

Cette classe représente une agence bancaire. Elle contient une liste de clients et de comptes, et permet la gestion des opérations bancaires à travers l'interface IAgence

Class Client :



```java
package Exercice_1;

public class Client {
private long id ;
private String cin ;
private String nom ;
private String prenom ;
public Client(long id ,String cin ,String nom ,String prenom) {
    this.id = id ;
    this.cin = cin ;
    this.nom = nom ;
    this.prenom = prenom ;
}
}

```

Cette classe modélise un client de la banque. Elle contient les informations personnelles du client et une référence à ses comptes.

Class Compte :

```java
package Exercice_1;

import java.time.LocalDate;

public class Compte {
private long id ;
private String numero ;
private LocalDate dateCreation;
List<Operation>operations = new ArrayList<>();
public List<Operation> getOperations() {
    return operations;
}
public void setOperations(List<Operation> operations) {
    this.operations = operations;
}
public Compte(long id ,String numero ,LocalDate dateCreation) {
    this.id = id ;
    this.numero = numero ;
    this.dateCreation = dateCreation ;
}
public String getNumero() {
    return numero;
}
public void setNumero(String numero) {
    this.numero = numero;
}
public long getId() {
    return id;
}
public void setId(long id) {
    this.id = id;
}
public LocalDate getDateCreation() {
    return dateCreation;
}
public void setDateCreation(LocalDate dateCreation) {
    this.dateCreation = dateCreation;
}
}

```

Cette classe représente un compte bancaire. Elle possède un identifiant, un solde et une liste d'opérations liées au compte.

Class Operation :

```java
package Exercice_1;

import java.time.LocalDate;

public class Operation {
private long id ;
private LocalDate dateOperation ;
private double montant ;
TypeOperation type ;

public Operation(long id ,LocalDate dateOperation,double montant ,TypeOperation type) {
    this.id = id ;
    this.dateOperation = dateOperation ;
    this.montant = montant ;
}

public LocalDate getDateOperation() {
    return dateOperation;
}

public void setDateOperation(LocalDate dateOperation) {
    this.dateOperation = dateOperation;
```

Classe abstraite représentant une opération bancaire (comme un dépôt ou un retrait). Elle est spécialisée par des classes concrètes selon le type d'opération.

Class TypeOperation :

```java
package Exercice_1;

public enum TypeOperation {
VERS,
RETR ,
VIRM
}
```

Enumération définissant les types d'opérations possibles (DÉPÔT, RETRAIT, etc.).

Class IAgence :

```java
package Exercice_1;

public interface IAgence {
void addCompte(Compte c) ;
Compte getCompte(String numero);
}
```

Class ICompteManagement :

```java
package Exercice_1;

import java.time.LocalDate;

public interface ICompteManagement {
void addOperatin(Compte c,Operation op);
List<Operation>getOperation(Compte c ,LocalDate d1 ,LocalDate d2) ;
void PrintDetail(Compte c);
}
```

Elles définissent les comportements attendus des classes qui les implémentent, respectivement pour la gestion des agences et des comptes.

Class CompteManagements :

```java
package Exercice_1;
import java.time.LocalDate;

public class CompteManagements implements ICompteManagement {
Agence agence ;
public CompteManagements(Agence agence){
this.agence = agence ;
}
    @Override
    public void addOperatin(Compte c, Operation op) {
        // TODO Auto-generated method stub
        c.getOperations().add(op);
    }

    @Override
    public List<Operation> getOperation(Compte c, LocalDate d1, LocalDate d2) {
        // TODO Auto-generated method stub
        List<Operation>result = new ArrayList<>();
        for (Operation op:c.getOperations()) {
            LocalDate date = op.getDateOperation() ;
            if((date.isEqual(d1)||date.isAfter(d1))&&(date.isEqual(d2) || date.isBefore(d2))){
                result.add(op);
            }
        }
        return result;
    }
    @Override
    public void PrintDetail(Compte c) {
        // TODO Auto-generated method stub
        System.out.println("l'Id du Compte :"+c.getId());
        System.out.println("le numero du Compte"+c.getNumero());
        System.out.println("la date de création :"+c.getDateCreation());
        List<Operation> operations = c.getOperations();
        int start = Math.max(0, operations.size() - 10);
        for (int i = start; i < operations.size(); i++) {
            System.out.println(operations.get(i));
        }
    }

}
```

Classe qui implémente ICompteManagement et qui gère les opérations de base telles que les dépôts, retraits et virements.

Class Main :

```java
package Exercice_1;

import java.time.LocalDate;

public class Programm {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
Agence a = new Agence() ;
CompteManagements compteM = new CompteManagements(a);
Compte compte = a.getCompte("A123");
compteM.addOperatin(compte, new Operation(1L ,LocalDate.of(2025, 5, 10),-150.0,TypeOperation.VERS));
compteM.addOperatin(compte, new Operation(2L ,LocalDate.of(2025, 5, 12),200.0,TypeOperation.VIRM));
List<Operation>operation = compteM.getOperation(compte, LocalDate.of(2025, 5, 1),LocalDate.of(2025, 5, 20));
for(Operation op : operation) {
    System.out.println(op);
}
compteM.PrintDetail(compte);
    }

}
```