

## ASSIGNMENT JAVA Day6

**Harshit Kushmakar | 16896**

**1. Create a Bank class and create an array of Customer in the Bank class according to the given class diagram below:**

**The Customer Array should be able to hold object of Customers. Perform the functionalities given in the Bank class.**

```
package Assignment6;

import assignment2.Customer;
import Assignment5.LoanProduct;

public class Bank{

    private Customer[] customers = new Customer[1000];

    private int size = 0;

    public void setCustomers(Customer[] customers) {

        this.customers = customers;

    }

    public Customer[] getCustomers() {

        return customers;

    }

    public boolean registerCustomer(Customer c){

        if(c == null){

            return false;

        }

        customers[size++] = c;

        return true;

    }

    public boolean findCustomer(Customer customer){

        for(Customer c : customers){
```

```
if(c.equals(customer)){  
    return true;  
}  
}  
return false;  
}  
  
public void printAllCustomers(){  
    for(Customer c : customers){  
        System.out.println(c.toString());  
    }  
}  
  
public boolean deleteCustomer(int customerId){  
    int check = 0;  
    for(int i=0; i<customers.length; i++){  
        if(customers[i].getCustomerId() == customerId){  
            check = 1;  
            break;  
        }  
    }  
    if(check == 0){  
        return false;  
    }  
    for (int i = 0; i < customers.length; i++) {  
        customers[i] = customers[i+1];  
    }  
    return true;  
}  
}
```

**2. Extend the functionality of Bank and add an array of LoanProduct as per the diagram below:**

**The array must be able to hold any LoanProduct type objects. Provide the methods for the functionalities mentioned in the above diagram.**

**a. addNewLoanProduct() method to be overloaded for all three types of LoanProduct.**

**b. calculateLTVforLoanProducts() method iterates over the loanProducts array and call**

**the abstract calculateLTVforLoanProduct() method of the LoanProduct class.**

```
package Assignment5;

import Assignment6.Bank;

public abstract class LoanProduct extends Bank {

    private String loanProductCode;

    private String loanProductName;

    private boolean assetBased;

    private String loanSecurityType;

    private double minTenure;

    private double maxTenure;

    private double minLoanAmount;

    private double maxLoanAmount;

    private double roi;

    private double ltv;

    //We Must use upcasting to call this LTVCalculationAsPerCollateralType() Method.

    public abstract double LTVCalculationAsPerCollateralType(double LoanAmountAsked,
double collateral);

}
```

```

package Assignment5;

import allenums.NatureOfProperty;
import allenums.PropertyOwnership;
import allenums.PropertyPurpose;
import allenums.PropertyType;
import assignment1.UtilitiesAll;

public class HomeLoan extends LoanProduct{

    private PropertyType propertyType;

    private NatureOfProperty natureOfProperty;

    private PropertyPurpose propertyPurpose;

    private PropertyOwnership propertyOwnership;

    private double marketValue;

    private double builtUpArea;

    private double carpetArea;

    private int propertyAge;

    @Override

    public double LTVCalculationAsPerCollateralType(double LoanAmountAsked,double
collateral) {

        return UtilitiesAll.calculateLTV(LoanAmountAsked,collateral);

    }

}

```

```

package Assignment5;

import allenums.AssetCategory;
import allenums.AssetVariant;
import assignment1.UtilitiesAll;

public class ConsumerVehicleLoan extends LoanProduct{

    private AssetCategory assetCategory;

    private AsssetVariant assetVariant;

    private String assetModel;

    private String manufacturer;

```

```
private int yearOfManufacture;

private double assetCost;

private double downPayment;

@Override

public double LTVCalculationAsPerCollateralType(double LoanAmountAsked,double collateral) {

    return UtilitiesAll.calculateLTV(LoanAmountAsked,collateral);

}

}
```

```
package Assignment5;

import allenums.CourseType;

import allenums.DegreeType;

import allenums.EducationStream;

import assignment1.UtilitiesAll;

public class EducationLoan extends LoanProduct{

    private String courseName;

    private String collegeName;

    private CourseType courseType;

    private DegreeType degreeType;

    private EducationStream educationStream;

    private double totalFees;

    @Override

    public double LTVCalculationAsPerCollateralType(double LoanAmountAsked,double collateral) {

        return UtilitiesAll.calculateLTV(LoanAmountAsked,collateral);

    }

}
```

**3. Create two interfaces – Maker and Operator. The interfaces are implemented by the Bank class and the functionalities are as shown below:**

**The Bank class will implement these methods. We need to create a Bank object and return the same**

**using the Interface reference so that only the provided methods in the corresponding interface are**

**invoked. There must be a single Bank instance.**

```
package Assignment6;
import assignment2.Customer;
import Assignment5.LoanProduct;
import assignment2.Customer;
public interface Maker {
    public boolean registerCustomer(Customer customer);
    public boolean deleteCustomer(Customer customerId);
    public boolean addNewLoanProduct();
    public boolean removeLoanProduct(LoanProduct loanProductCode);
}
package Assignment6;
import assignment2.Customer;
import Assignment5.LoanProduct;
public interface Operator {
    public void printAllLoanProducts();
    public void printLoanProductDetails(LoanProduct loanProductId);
    public void calculateLTVForLoanProducts();
    public Customer findCustomer(int customerId);
    public boolean findCustomer(String customer);
```

```

    public void printAllCustomer();
}

package Assignment6;

import assignment2.Customer;

import Assignment5.LoanProduct;

public class Bank implements Maker,Operator{

    private Customer[] customers = new Customer[1000];

    private int size = 0;

    public void setCustomers(Customer[] customers) {

        this.customers = customers;

    }

    public Customer[] getCustomers() {

        return customers;

    }

    public boolean registerCustomer(Customer c){

        if(c == null){

            return false;

        }

        customers[size++] = c;

        return true;

    }

    @Override

    public boolean deleteCustomer(Customer customerId) {

        return false;

    }

    @Override

    public boolean addNewLoanProduct() {

        return false;

    }
}

```

**@Override**

```
public boolean removeLoanProduct(LoanProduct loanProductCode) {  
    return false;  
}  
  
public boolean findCustomer(Customer customer){  
    for(Customer c : customers){  
        if(c.equals(customer)){  
            return true;  
        }  
    }  
    return false;  
}  
  
public void printAllCustomers(){  
    for(Customer c : customers){  
        System.out.println(c.toString());  
    }  
}  
  
public boolean deleteCustomer(int customerId){  
    int check = 0;  
    for(int i=0; i<customers.length; i++){  
        if(customers[i].getCustomerId() == customerId){  
            check = 1;  
            break;  
        }  
    }  
    if(check == 0){  
        return false;  
    }  
    for (int i = 0; i < customers.length; i++) {
```



```
customers[i] = customers[i+1];
}
return true;
}

@Override
public void printAllLoanProducts() {
}

@Override
public void printLoanProductDetails(LoanProduct loanProductId) {
}

@Override
public void calculateLTVForLoanProducts() {
}

@Override
public Customer findCustomer(int customerId) {
return null;
}

@Override
public boolean findCustomer(String customer) {
return false;
}

@Override
public void printAllCustomer() {
}
}
```