Bank Class

```
public void addCustomer(Customer customer, String branch_name)
```
adds the new customer and the Bank branch into an ArrayList using the add() method.

```
	public void removeCustomer(Customer customer, String branch_name)
```
removes the new customer from the bank's list using the remove() method.

```
	public void print_customers()
```
By using the list iterator, this method prints all the instances of the list.

```
	public void printBranches()
```
Same goes with this method.

```
	public String getBankName()
```
A standard getter that returns the Bank's name.

Account Class:

```
	public String getAccount_id()
```
Getter that returns the account id.

```
	public double getAccountBalance()
```
Getter that returns the account balance.

```
	public void printAccountInfo()
```
Prints the account id and the account balance. It's a more generic method because there are two types of accounts.

Checking Account Class extends Account class

```
	public Checking_Account(String account_id, double account_balance, 
LocalDate creation_date)
```
Checking Account Constructor.

```
	public void deposit(double amount)
```
In this method we just add the parameter to the account balance.

```
	public void withdraw(double amount)
```
In this method we are trying to withdraw money from the account balance. If the asking amount is bigger than the account balance, then a message is being prompted that the balance has insufficient funds. Else, if the asking amount is less than the balance and the balance is greater than 0, then the amount is being substracted from the balance account.

```
		public void printCheckingInfo()
```
It prints the Checking Account information such as it's balance, creation date and it's ID.

Savings Account class extends Account class

```
	public Savings_Account(String account_id, double account_balance, 
LocalDate creation_date) {
```

Savings Account constructor.

```
	public void add_interest(int month)
```

This method calculates the interest for a given month period that has passed. It uses the type :

*Future Value = Primary Value * (1+interest_rate/12)^months_passed.*

For this method to work in general, when you are creating the account you have to insert a date that is before the current dare that you are using this program. That happens because the method takes two LocalDate objects and using the between() method that is provided in LocalDate.class and returns the difference between those months (*e.g: Let's say an account has been created on 01.01.2000 and you want to see the account's balance after let's say 5 months the between() method will return 01.06.2000 - 01.01.2000 = 5.)*

Then it will store this value to a variable called months_passed and this is the number that will be the input for *months_passed* on the above type.

-If 2 months has passed, it will calculate the balance with interest 5%

-If 4 months has passed, it will calculate the balance with interest 10%

-If 6 or more months has passed, it will calculate the balance with interest 12%

The methods withdraw() and deposit() are working exactly like the ones in Checking Account.

```
	public void printInfoSavings()
```

This method prints the Account's info

Branches Class:

```
	public Branches(String name, String bank_name)
```

Constructor

```
	public void setName(String name)

	public String getName()
setters and getters for Branches name.
```

```
	public boolean Belongs()
```

A method that checks if a Branch belongs to a Bank.

```
	public void printBranchInfo()
```

Method to print Branch info such as it's name and the bank it belongs to

Customer Class:

```
	public Customer(String first_name, String last_name, String email, 
int account_id)
```

Constructor for Customer.

```
		public void addAccount(Account account)
```

Method to add accounts to a list using the add() method.

```
	public void removeAccount(Account account)
```

Method to remove accounts to a list using the remove() method.

```
	public void printAllAccounts()
```

Method to print all accounts.

```
public boolean isMailValid()
```

Method to check the validation of the customer's email. An email should have at least one character before '@' and should end with .com, so here we used the contains() and endsWith() methods from String. If email is invalid, the program will exit and will tell the user/programmer to check the input on the email field and try to re-run the program.

Main Class

```
			Bank bank = new Bank("Alpha Bank");
```

Creating a new Bank.

```
			Customer c = new Customer ("John" ,  "Doe ", 
"JohnDoe@example.com", 1001); //John being registered
```

And a Customer.

```
			bank.addCustomer(c,"Kifisia Branch"); //into bank's 
list in Branch of Kifissia
```

calling the add method to add the customer into the Bank's list of customers.

```
                    bank.print_customers();
```

Call the print_customers() method to print all the customers of the Bank.

```
                    Savings_Account savings_account = new
Savings_Account("SA1001", 0,  LocalDate.now());//Create an Account
```

```
                    c.addAccount(savings_account); //add the account to
Customer's List of accounts
```

```
                    savings_account.deposit(200);
```

Calling the deposit method to deposit 200$.

```
Checking_Account checking_account  = new Checking_Account("CA2001", 0,
LocalDate.now()); //Create another Account
                    c.addAccount(checking_account); // add the second
Account into Customer's List as well

checking_account.deposit(145);//1st account deposits 145
                    Checking_Account checking_account2  = new
Checking_Account("CA2002", 0, LocalDate.now()); //Create another Account

                    c.addAccount(checking_account2); // add the second
Account into Customer's List as well

      checking_account2.deposit(160);//2nd account deposits 160
                          checking_account2.printCheckingInfo();//print 2nd
acc's info
checking_account.deposit(20);//1st acc deposits 20
checking_account.withdraw(180);//1st acc tries to withdraw 180
      checking_account.withdraw(150);//1st acc tries to withdraw 150
                          checking_account.printCheckingInfo();
savings_account.add_interest(4);//this works only for the example of 4-9
months.Thats because it adds to the existing month X months with a limit to
12. So if we pass '10'

      //as a parameter, it will throw an exception because it would have a
month with value 13 and thats not allowed. For it to work for every
example, when i create an account

      //i should give the date myself, and remove some code from
Savings_Account line:43
```