Your job is to create a simple banking application.

Implement the following classes:

1. Bank

- It has two fields, A String called name and an ArrayList that holds objects of type Branch called branches.

- A constructor that takes a String (name of the bank). It initialises name and instantiates branches.

- And five methods, they are (their functions are in their names):

  - addBranch(), has one parameter of type String (name of the branch) and returns a boolean. It returns true if the branch was added successfully or false otherwise.

  - addCustomer(), has three parameters of type String (name of the branch), String (name of the customer), double (initial transaction) and returns a boolean. It returns true if the customer was added successfully or false otherwise.

  - addCustomerTransaction(), has three parameters of type String (name of the branch), String (name of the customer), double (transaction) and returns a boolean. It returns true if the customers transaction was added successfully or false otherwise.

  - findBranch(), has one parameter of type String (name of the Branch) and returns a Branch. Return the Branch if it exists or null otherwise.

  - listCustomers(), has two parameters of type String (name of the Branch), boolean (print transactions) and returns a boolean. Return true if the branch exists or false otherwise. This method prints out a list of customers.

→ TEST CODE

```
Bank bank = new Bank("National Australia Bank");

bank.addBranch("Adelaide");

bank.addCustomer("Adelaide", "Tim", 50.05);
bank.addCustomer("Adelaide", "Mike", 175.34);
bank.addCustomer("Adelaide", "Percy", 220.12);

bank.addCustomerTransaction("Adelaide", "Tim", 44.22);
bank.addCustomerTransaction("Adelaide", "Tim", 12.44);
bank.addCustomerTransaction("Adelaide", "Mike", 1.65);

bank.listCustomers("Adelaide", true);
```

→ OUTPUT

The list of customers should be printed out in the following format if boolean parameter is true:

```
Customer details for branch Adelaide
Customer: Tim[1]
Transactions
[1]  Amount 50.05
[2]  Amount 44.22
[3]  Amount 12.44
Customer: Mike[2]
```

Transactions
[1] Amount 175.34
[2] Amount 1.65
Customer: Percy[3]
Transactions
[1] Amount 220.12

and if false, only the customers - no transactions:

bank.listCustomers("Adelaide", false);

Customer details for branch Adelaide
Customer: Tim[1]
Customer: Mike[2]
Customer: Percy[3]

2. Branch

- It has two fields, A String called name and an ArrayList that holds objects of type Customer called customers.

- A constructor that takes a String (name of the branch). It initialises name and instantiates customers.

- And five methods, they are (their functions are in their names):

  - getName(), getter for name.

  - getCustomers(), getter for customers.

  - newCustomer(), has two parameters of type String (name of customer), double (initial transaction) and returns a boolean. Returns true if the customer was added successfully or false otherwise.

  - addCustomerTransaction(), has two parameters of type String (name of customer), double (transaction) and returns a boolean. Returns true if the customers transaction was added successfully or false otherwise.

  - findCustomer(), has one parameter of type String (name of customer) and returns a Customer. Return the Customer if they exist, null otherwise.

3. Customer

- It has two fields, A String called name and an ArrayList that holds objects of type Double called transactions.

- A constructor that takes a String (name of customer) and a double (initial transaction). It initialises name and instantiates transactions.

- And three methods, they are (their functions are in their names):

  - getName(), getter for name.

  - getTransactions(), getter for transactions.

  - addTransaction(), has one parameter of type double (transaction) and doesn't return anything.

TIP: In Bank, use the findBranch() method in each of the other four methods to validate a branch. Do the same thing in Branch with findCustomer() - except for the two getters.

TIP: In Customer, think about what else you need to do in the constructor when you instantiate a Customer object.

TIP: Think about what methods you need to call from another class when implementing a method.

TIP: Be extremely careful with the spelling of the names of the fields, constructors and methods.

TIP: Be extremely careful about spaces and spelling in the printed output.

NOTE: All transactions are deposits (no withdraws/balances).

NOTE: All fields are private.

NOTE: All constructors are public.

NOTE: All methods are public (except for findBranch() and findCustomer() which are private).

NOTE: There are no static members.

NOTE: Do not add a main method to the solution code.

NOTE: If you get an error from the Evaluate class, it's most likely the constructor. Check if you've added a constructor or if the constructor has the right arguments.