CS 102

Object Oriented Programming

Extending Bank Account Example

Reyyan Yeniterzi

reyyan.yeniterzi@ozyegin.edu.tr

Current Account Class (Version 16)


```
public class Account {
    private int number;
    private double balance;
    private String currency;

    public Account(int number, double balance, String currency) {
    public Account(int number, String currency) {
    public Account(int number) {

    public int getNumber()  {
    public double getBalance()  {
    public String getCurrency() {

    public void setCurrency(String currency)    {

    private void checkSetCurrency (String c) {

    public void deposit(double d) {
    public void withdraw(double d) {

    public void report() {

    public String toString() {
}
```

Another class

- Lets add another class
  - Customer object

Another class

- Lets add another class
  - Customer object
    - Name
    - Account

Customer class

- Customer object
  - Name
  - Account

```
private String name;
private Account account;

public Customer(String name, Account account) {
    this.name = name;
    this.account = account;
}
```

Customer Class


```
public String getName() {
    return this.name;
}
public Account getAccount() {
    return this.account;
}

public void deposit(double amount) {
    this.account.deposit(amount);
}
public void withdraw(double amount) {
    this.account.withdraw(amount);
}

public void report() {
    System.out.println("Customer " + this.name + " ");
    this.account.report();
}
```

Using Customer Class


```
public static void main(String[] args) {

    Account account1 = new Account(1, 100, "TL");
    Customer customer1 = new Customer("Ali", account1);

    Account account2 = new Account(1, 200, "USD");
    Customer customer2 = new Customer("Veli", account2);

    customer1.report();
    customer2.report();
}
```

Using Customer Class


```
public static void main(String[] args) {

    Account account1 = new Account(1, 100, "TL");
    Customer customer1 = new Customer("Ali", account1);

    Account account2 = new Account(1, 200, "USD");
    Customer customer2 = new Customer("Veli", account2);

    customer1.report();
    customer2.report();
}
```

Problems | Javadoc | Declaration | Console

<terminated> AccountTest (5) [Java Application] C:\Program Fil

```
Customer Ali
Account 1 has 100.0 TL.
Customer Veli
Account 1 has 200.0 USD.
```

Draw the Memory Model

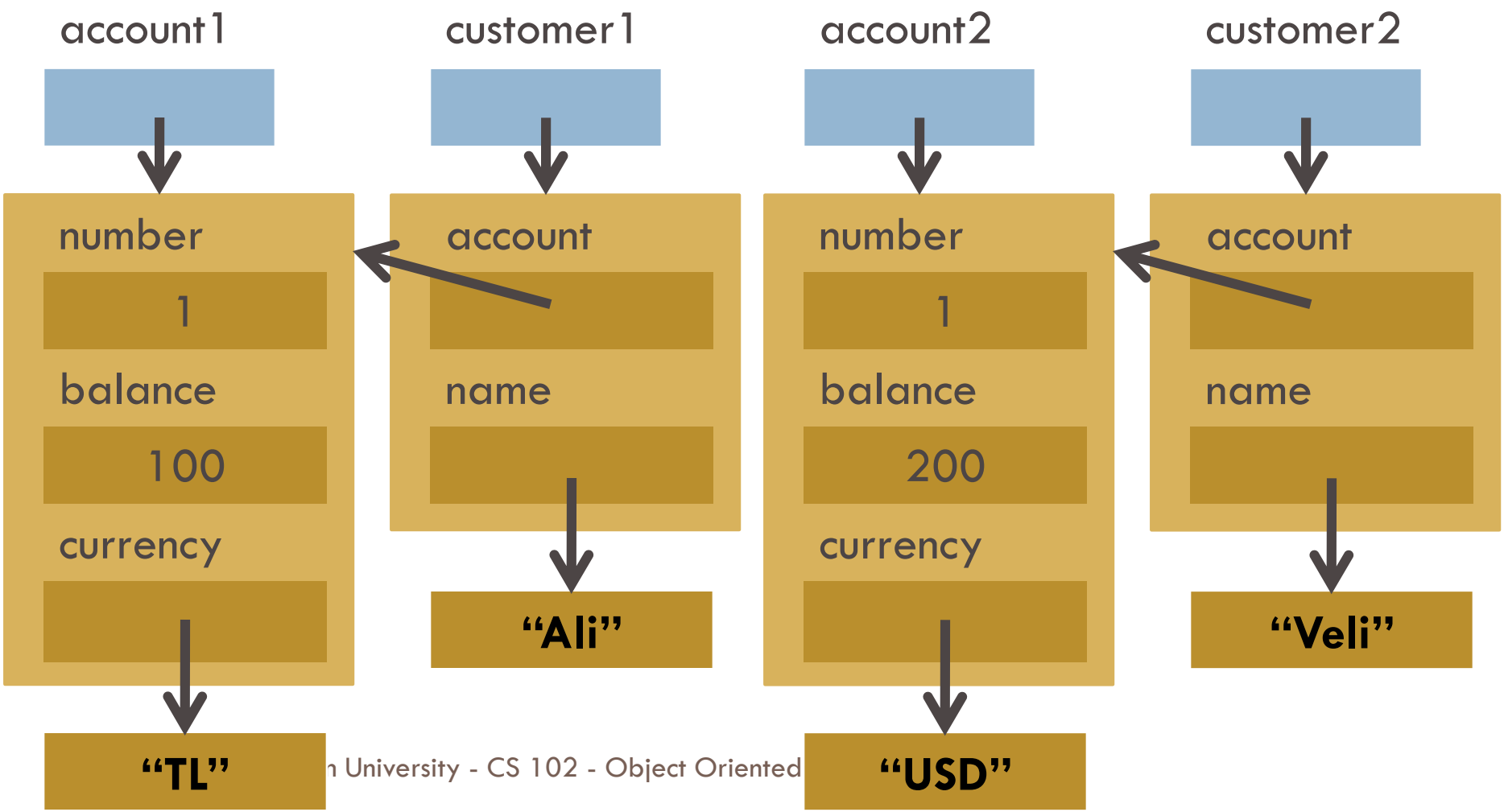

```java
public static void main(String[] args) {

    Account account1 = new Account(1, 100, "TL");
    Customer customer1 = new Customer("Ali", account1);

    Account account2 = new Account(1, 200, "USD");
    Customer customer2 = new Customer("Veli", account2);

    customer1.report();
    customer2.report();
}
```

Problems | Javadoc | Declaration | Console

<terminated> AccountTest (5) [Java Application] C:\Program Fil

```
Customer Ali
Account 1 has 100.0 TL.
Customer Veli
Account 1 has 200.0 USD.
```

```
Account account1 = new Account(1, 100, "TL");
Customer customer1 = new Customer("Ali", account1);

Account account2 = new Account(1, 200, "USD");
Customer customer2 = new Customer("Veli", account2);
```



What is the output?

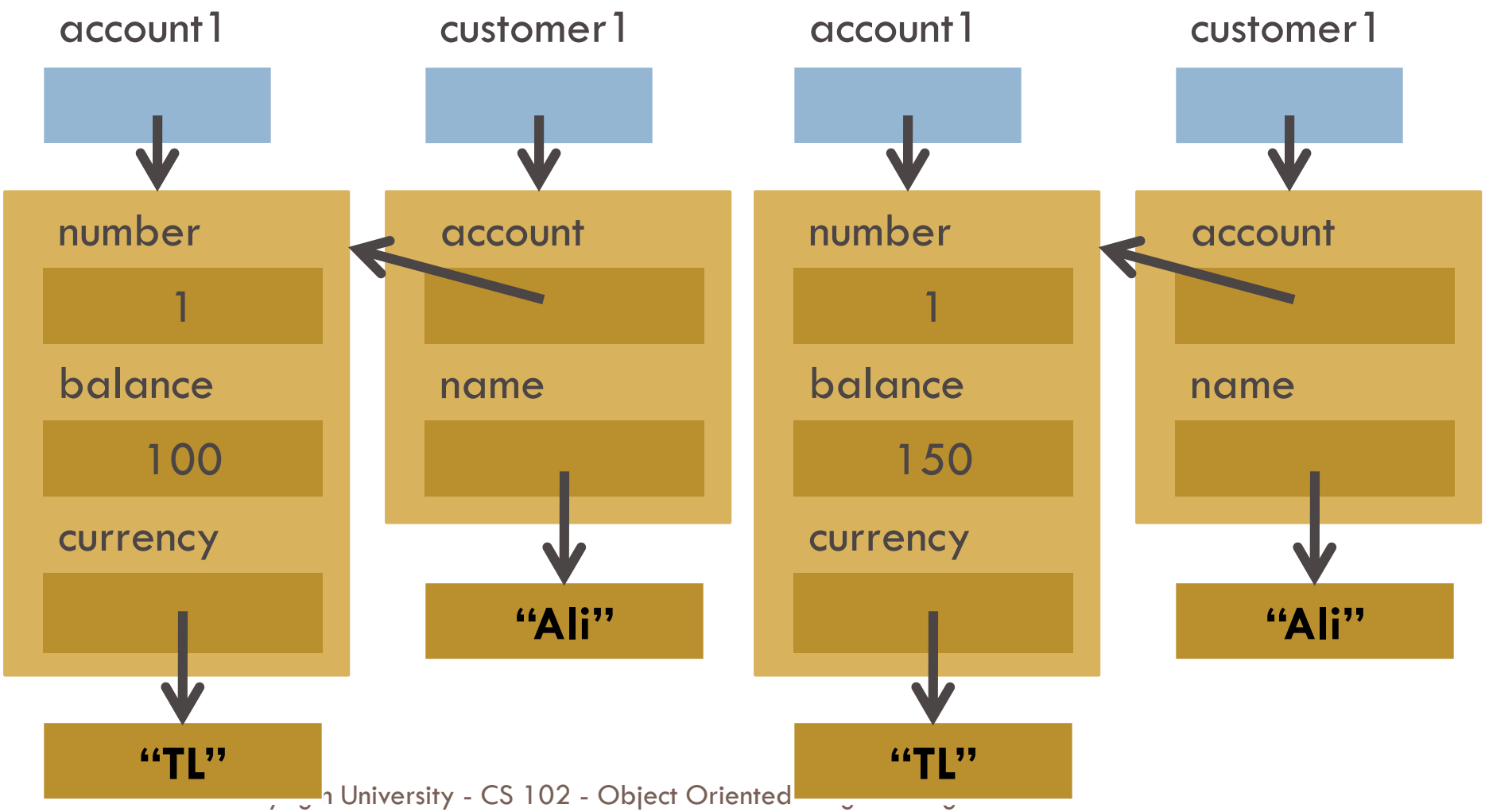

```
public static void main(String[] args) {

    Account account1 = new Account(1, 100, "TL");
    Customer customer1 = new Customer("Ali", account1);

    Account account2 = new Account(1, 200, "USD");
    Customer customer2 = new Customer("Veli", account2);

    customer1.deposit(50);

    customer1.report();
    customer2.report();
}
```

What is the output?


```java
public static void main(String[] args) {

    Account account1 = new Account(1, 100, "TL");
    Customer customer1 = new Customer("Ali", account1);

    Account account2 = new Account(1, 200, "USD");
    Customer customer2 = new Customer("Veli", account2);

    customer1.deposit(50);

    customer1.report();
    customer2.report();
}
```

```
<terminated> AccountTest (5) [Java Application] C:\Program Fil
50.0 TL have been deposited
The balance is 150.0 TL
Customer Ali
Account 1 has 150.0 TL.
Customer Veli
Account 1 has 200.0 USD.
```

Before and After deposit

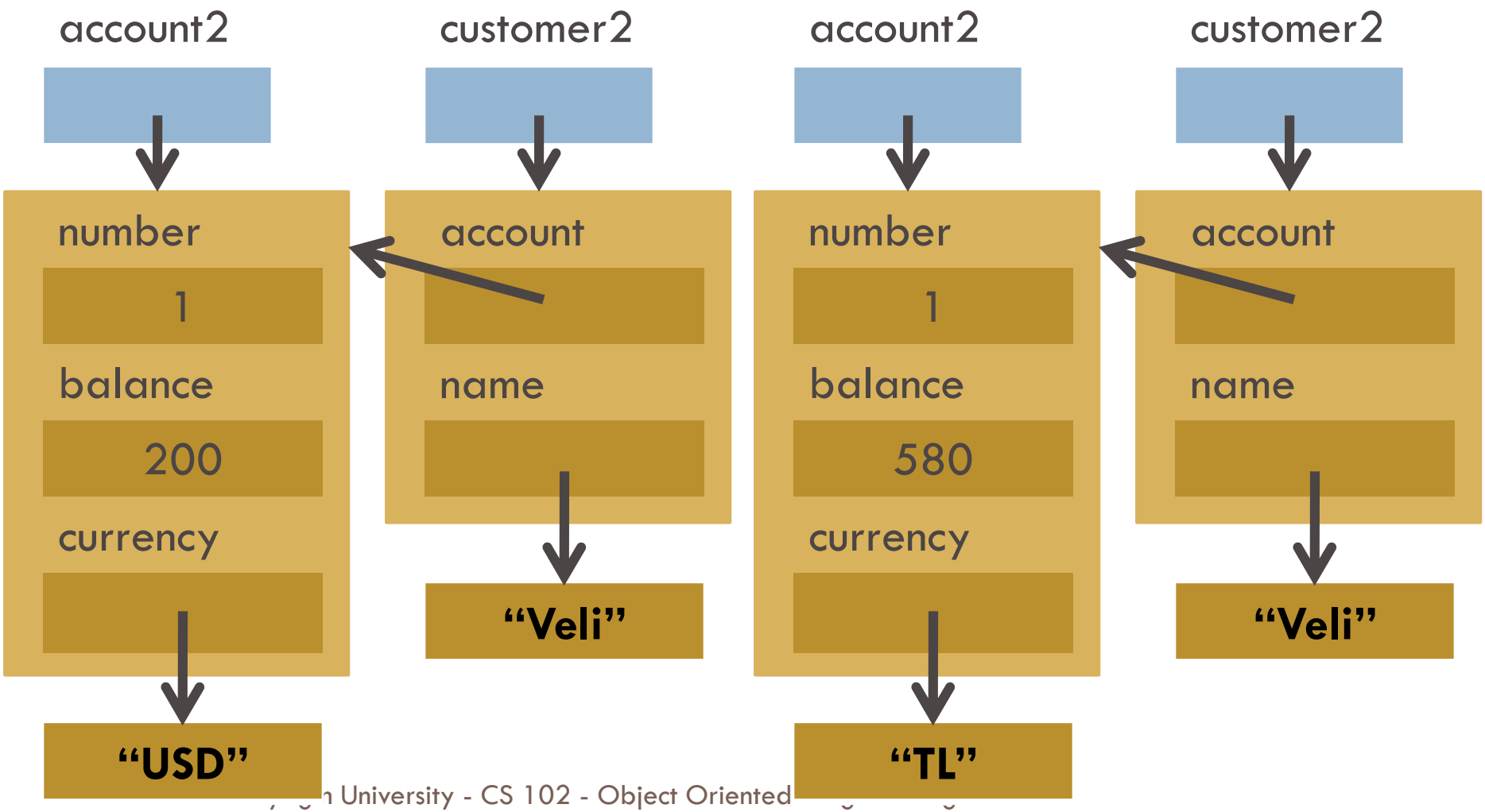


What is the output?


```java
Account account1 = new Account(1, 100, "TL");
Customer customer1 = new Customer("Ali", account1);

Account account2 = new Account(1, 200, "USD");
Customer customer2 = new Customer("Veli", account2);

customer1.deposit(50);
customer2.getAccount().setCurrency("TL");

customer1.report();
customer2.report();
```

```
<terminated> AccountTest (5) [Java Application]
50.0 TL have been deposited
The balance is 150.0 TL
Customer Ali
Account 1 has 150.0 TL.
Customer Veli
Account 1 has 580.0 TL.
```

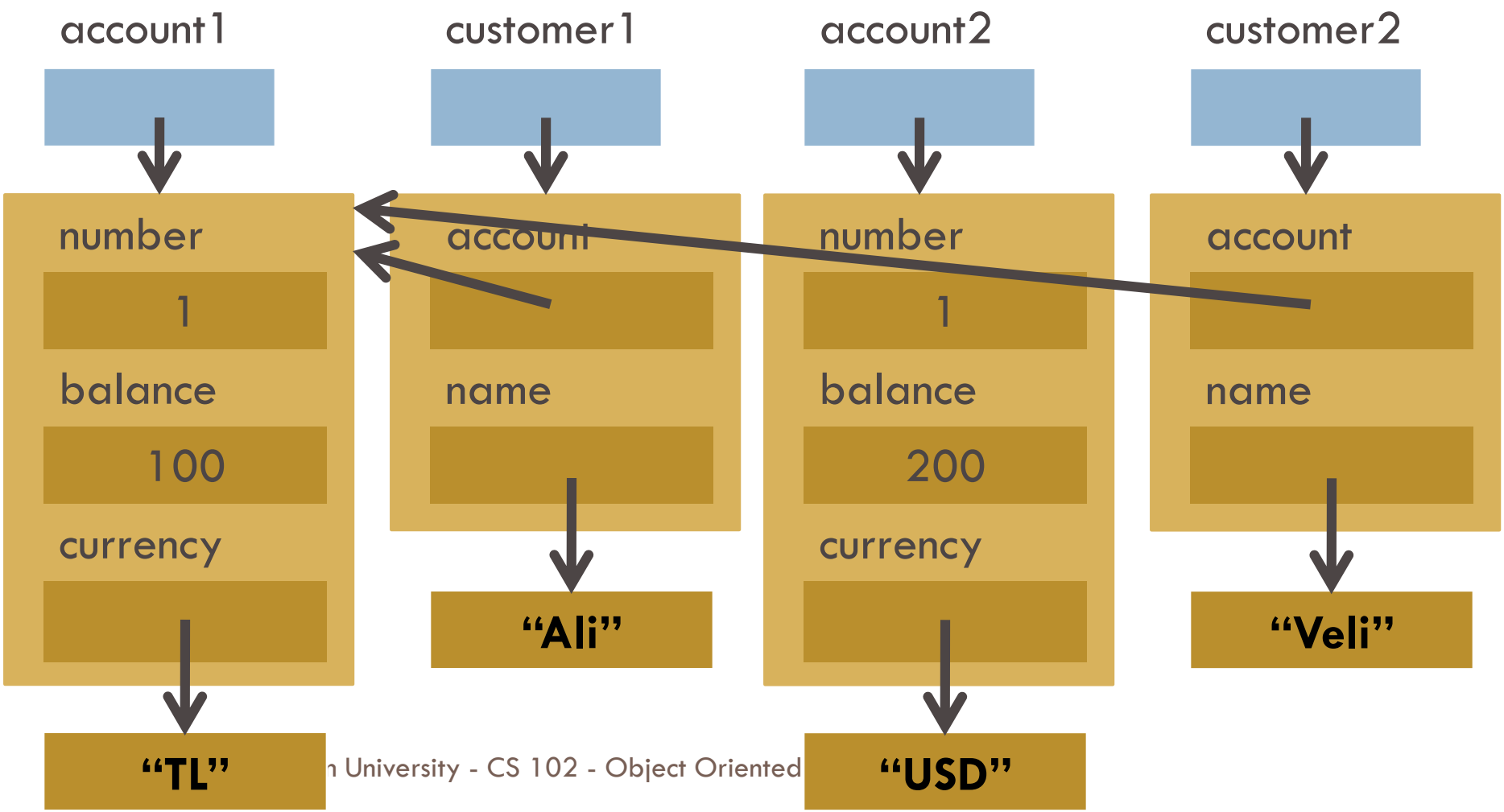
Before and After setCurrency



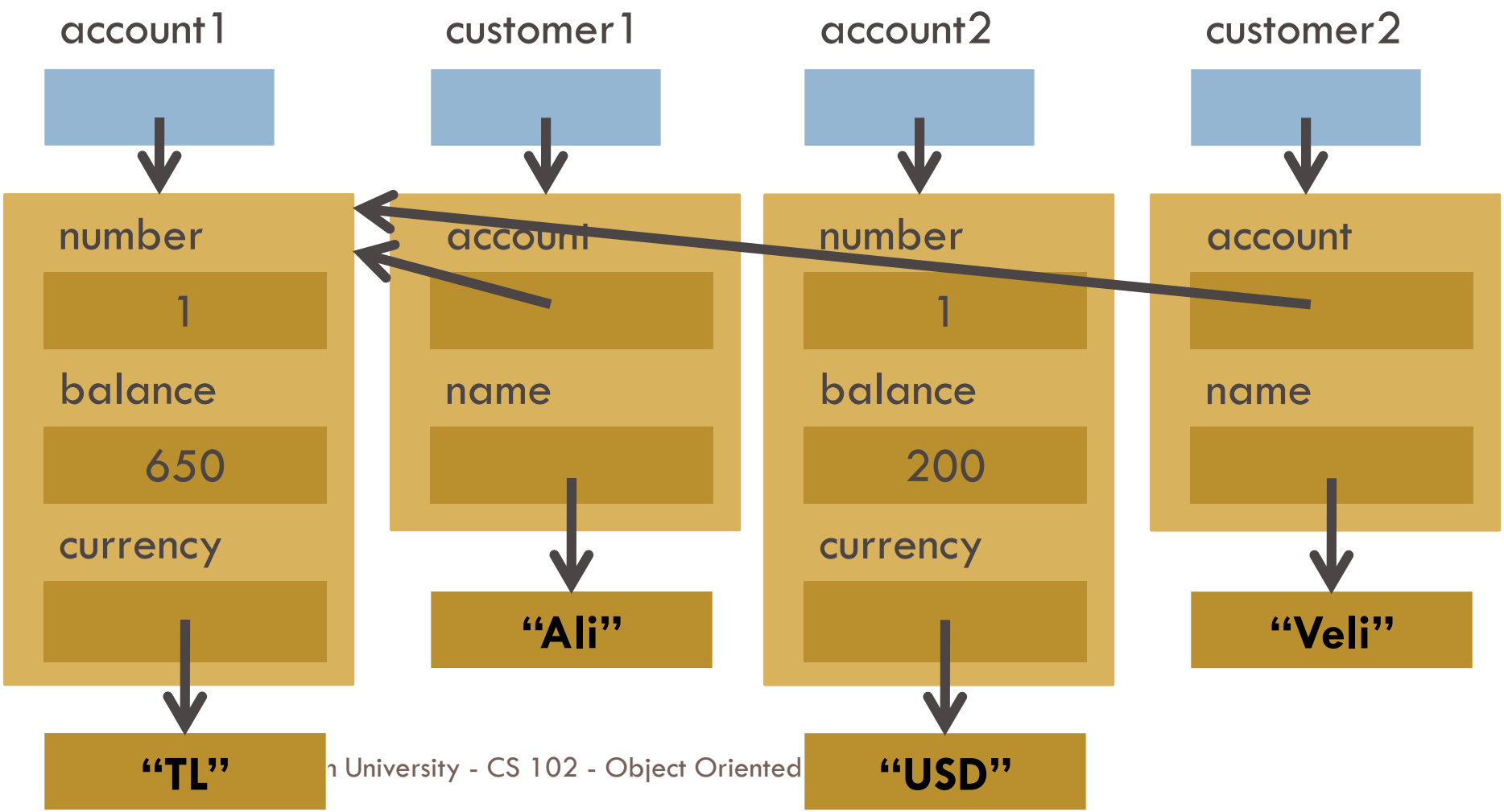
What is the final memory model?


```
Account account1 = new Account(1, 100, "TL");
Customer customer1 = new Customer("Ali", account1);

Account account2 = new Account(1, 200, "USD");
Customer customer2 = new Customer("Veli", account1);

customer1.deposit(50);
customer2.deposit(500);

account1.withdraw(100);
account2.withdraw(200);

customer1.report();
customer2.report();
```

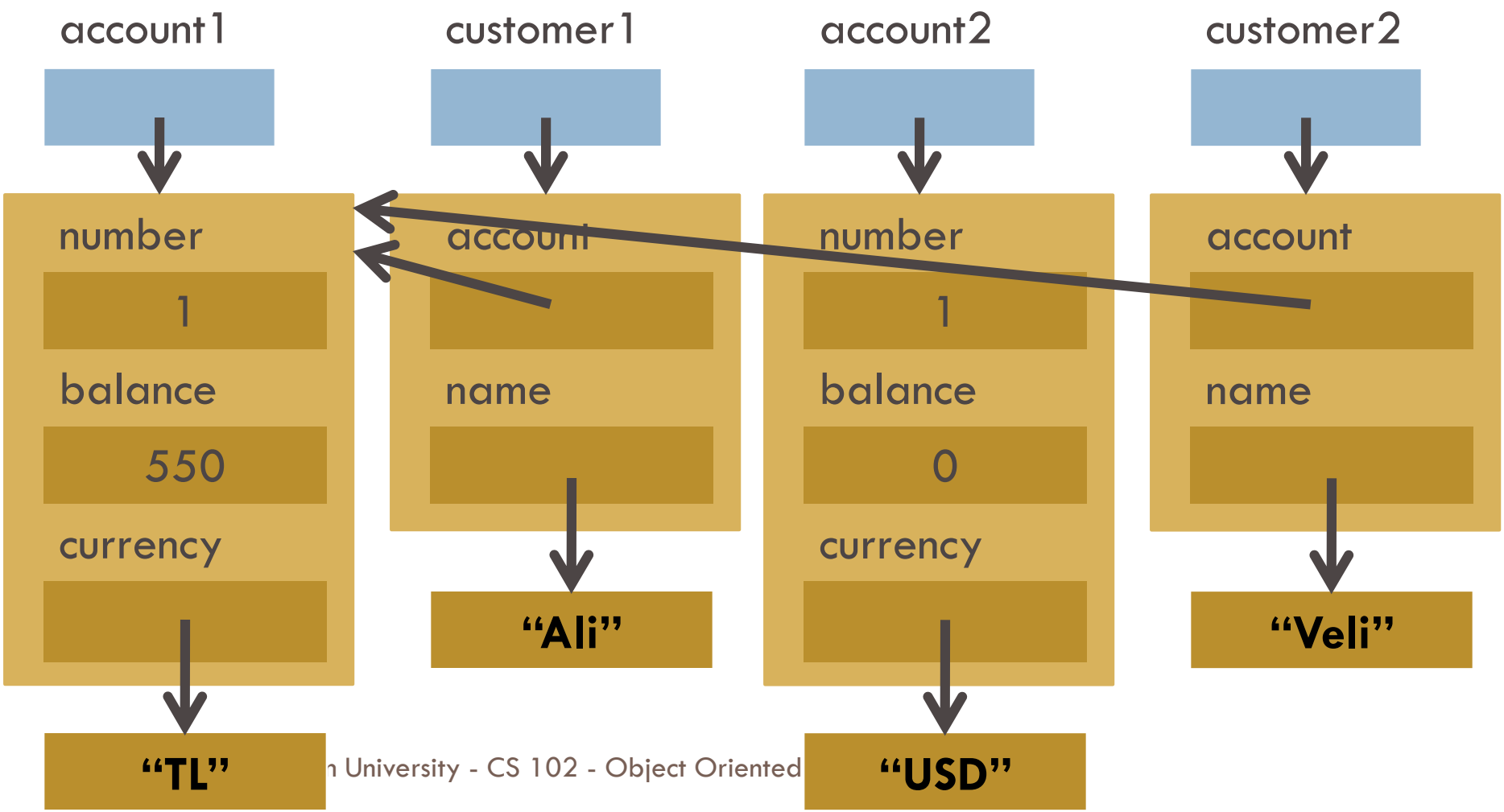
```
Account account1 = new Account(1, 100, "TL");
Customer customer1 = new Customer("Ali", account1);

Account account2 = new Account(1, 200, "USD");
Customer customer2 = new Customer("Veli", account1);
```



```
customer1.deposit(50);
customer2.deposit(500);
```

```
account1.withdraw(100);
account2.withdraw(200);
```

Additional Classes

- We have customer and account, lets have a bank then.
- A bank has a name and customers.

Additional Classes

- We have customer and account, lets have a bank then.
- A bank has a name and customers.
- Only one name but multiple customers.
  - name (String)
  - customers (array)

Bank Class – Class Instances

- Only one name but multiple customers.
  - name (String)
  - customers (array)

```
public class Bank {
        private String name;
        private Customer[] customers;
```

- How many customers?
  - Need to know in advance, why?

Bank Class – Class Instances

- Lets say a bank can have at most 3 customers.
- Create an array of size 3

Bank Class – Class Instances

- Lets say a bank can have at most 3 customers.
- Create an array of size 3
- But you don't have to use all 3 customers. It can be less. Therefore keep the number of customers value in a variable.

```
public class Bank {
        private String name;
        private Customer[] customers;
        private int numCustomers;
```

Bank Class - Constructor

- Initially banks have no customers.
- What should be the constructor arguments?

Bank Class - Constructor

- Initially banks have no customers.
- What should be the constructor arguments?

```
public Bank(String n) {
  name = n;
  customers = new Customer[3];
  numCustomers = 0;
}
```

Bank Class – Adding Customers

- An addCustomer method to add customers.
- This method takes one customer as an argument.
- It updates the array and the numCustomers value.

```
public void addCustomer(Customer c) {
  customers[numCustomers] = c;
  numCustomers++;
}
```

Bank Class – Other Functions


```
public String getName() {
  return name;
}

public void setName(String n) {
  name = n;
}
public void display() {
  System.out.println("---- "+name+" ----");
  for(int i=0; i < numCustomers; i++) {
    customers[i].report();
  }
  System.out.println("------------------");
}
```

Bank Application

- Assume that we have an application which takes customer information in runtime from users.
- We need to use Scanner in order to read the input from the console.

```
import java.util.Scanner;

public class AccountTest {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
```

Bank Application

- For each customer, what kind of information do we need?

Bank Application

- For each customer, what kind of information do we need?
  - Name
  - Account
    - Balance
    - Currency
    - Number?
      - The system can assign the next available account number to the account.
      - Need to keep a counter for account number.

```java
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    Bank bank = new Bank("TrustBank");
    int accountNo = 1;

    System.out.println("Welcome to " + bank.getName());
    while(true) {
      System.out.print("Enter customer name (empty to quit): ");
      String customerName = input.nextLine();
      if(customerName.equals(""))
        break;

      System.out.print("Enter currency: ");
      String curr = input.nextLine();

      System.out.print("Enter initial balance: ");
      double balance = Double.parseDouble(input.nextLine());

      bank.addCustomer(new Customer(customerName,
              new Account(accountNo, balance, curr)));
      accountNo++;
      bank.display();
    }
    System.out.println("Bye!");
}
```

Memory Model?

- What will be the memory model after user enters
  - "Ali" for customer name
  - "TL" for account's currency
  - 100 for initial balance

Memory Model?

- What will be the memory model after user enters
  - "Ali" for customer name
  - "TL" for account's currency
  - 100 for initial balance

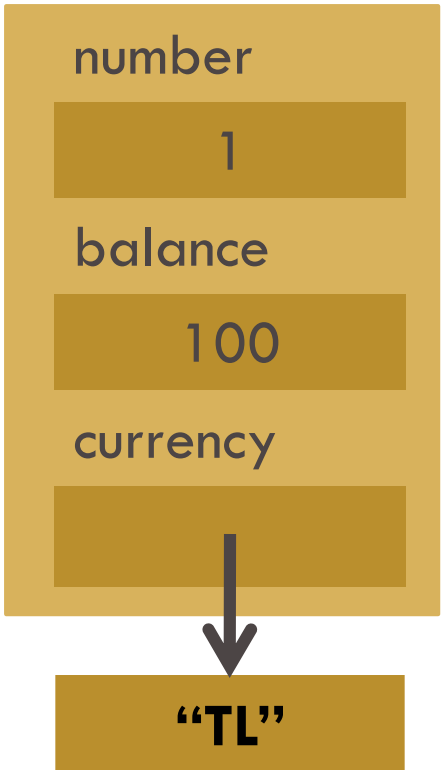
- How many objects are we going to create?

```
bank.addCustomer(new Customer(customerName,
        new Account(accountNo, balance, curr)));
```

Memory Model


```
bank.addCustomer(new Customer(customerName,
        new Account(accountNo, balance, curr)));
```

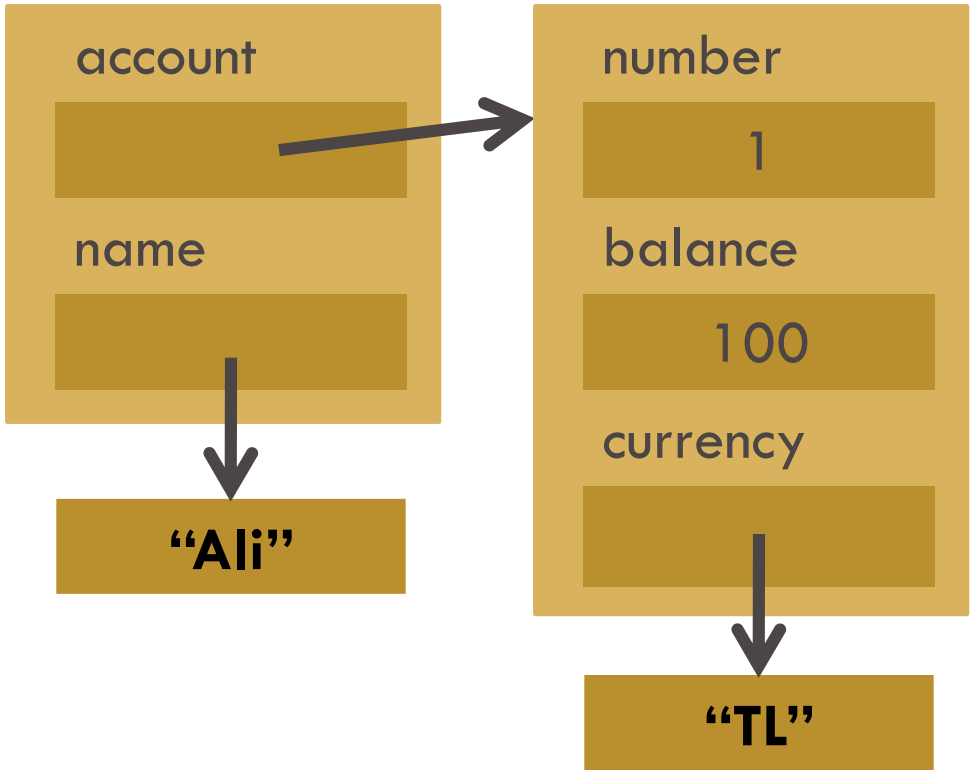
- The path is from inside out.
- User entered
  - "TL" for account's currency
  - 100 for initial balance
- Return its reference to Customer object.



Memory Model

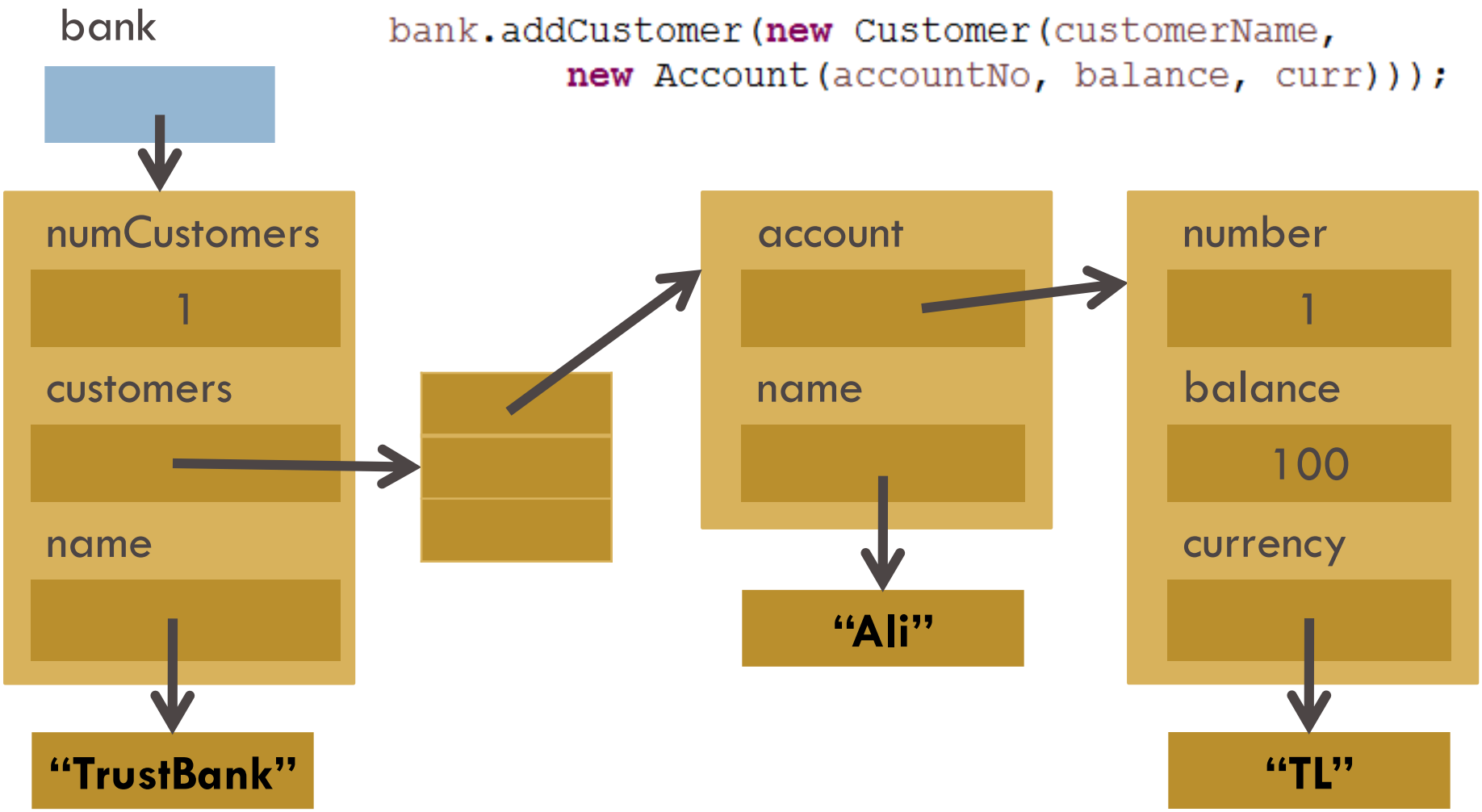

```
bank.addCustomer(new Customer(customerName,
        new Account(accountNo, balance, curr)));
```

- Save customer's address at bank's customer array

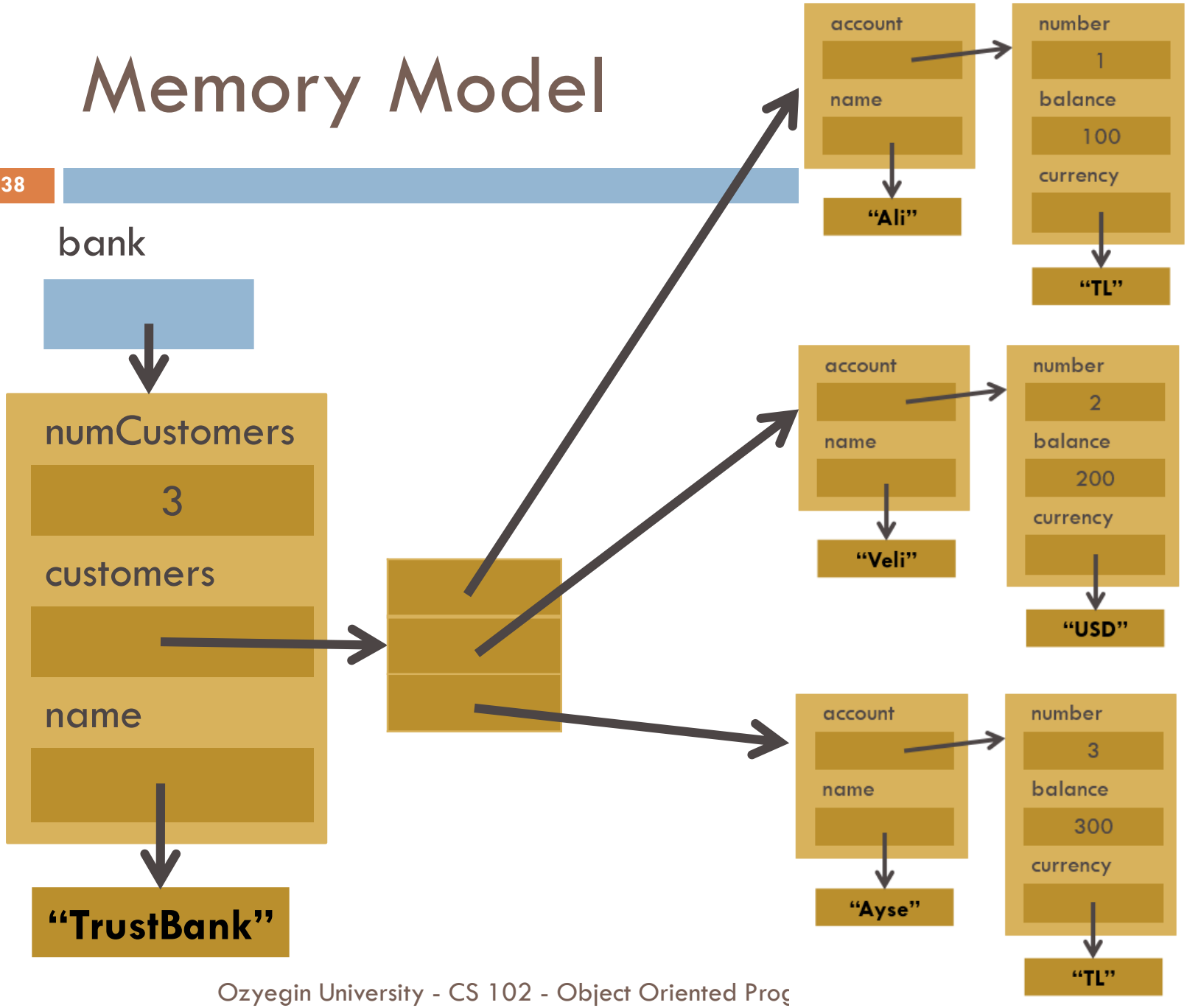


Memory Model


```
bank.addCustomer(new Customer(customerName,
                 new Account(accountNo, balance, curr)));
```



Memory Model



Arrays ☹

- Arrays are fixed length.
- We need a data structure that can be resized.

Arrays ☹

- Arrays are fixed length.
- We need a data structure that can be resized.
  - ArrayList ☺

Arrays ☹

- Arrays are fixed length.
- We need a data structure that can be resized.
  - ArrayList ☺

- ArrayList
  - Dynamic in size
  - See ArrayList slides …

Bank Class

- with ArrayList
- We don't need numCustomers anymore.

```
private String name;
private ArrayList<Customer> customers;

public Bank(String n) {
  name = n;
  customers = new ArrayList<Customer>();
}

public String getName() {
  return name;
}
public void setName(String n) {
  name = n;
}

public void addCustomer(Customer customer) {
  customers.add(customer);
}

public void display() {
  System.out.println("---- "+name+" ----");
  for(Customer customer: customers) {
    customer.report();
  }
  System.out.println("------------------");
}
```

Any Questions ?