


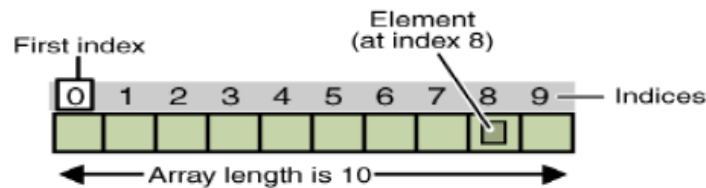
PBO 7 : Array & Arraylist

Royana Afwani



Konsep Array (Larik)

- **Array** dalam java diperlakukan sebagai **objek**.
- **Array** adalah objek yang dapat digunakan untuk menyimpan sejumlah data dalam tipe sama dengan **jumlah elemen tetap**



- Elemen yang disimpan pada array dapat berupa **tipe primitif** (int, float, etc) atau **objek** (instan dari class)
- Langkah menciptakan array:
 1. Mendeklarasikan variabel array
 2. Menciptakan objek array

Deklarasi Variabel Array

- Bentuk Deklarasi:

`tipePrimitif[] namaVariabel;`

`namaKelas[] namaVariabel;`

- Contoh:

`String[] kota;`

`int[] nomor;`

Menciptakan Objek Array

- Bentuk Deklarasi:
`namaVariabel = new tipePrimitif[jumlahElemen];`
`namaVariabel = new namaKelas[jumlahElemen];`
- Contoh:
`nomor = new int[7];`
`kota = new String[8];`
- Bentuk singkat deklarasi variable dan objek array:
`String[] kota = new String[8];`
`int[] nomor = new int[7];`

Deklarasi dan Pemberian Nilai Array

Syntax To construct an array: `new typeName[length]`
To access an element: `arrayReference[index]`

Example

Diagram illustrating array declaration and access:

Declaration: `double[] values = new double[10];`

- Type of array variable:** `double[]`
- Name of array variable:** `values`
- Element type:** `double`
- Length:** `10`
- Initialized with zero** (Callout for `new double[10]`)

Declaration: `double[] moreValues = { 32, 54, 67.5, 29, 35 };`

- Initialized with these elements** (Callout for the list of values)

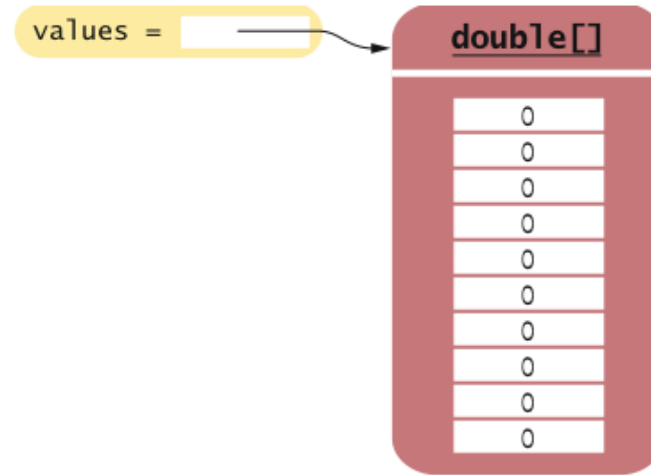
Access: `values[i] = 29.95;`

- Use brackets to access an element.** (Callout for `values[i]`)
- The index must be ≥ 0 and $<$ the length of the array.** (Callout for `i`)

Deklarasi dan Pemberian Nilai Array

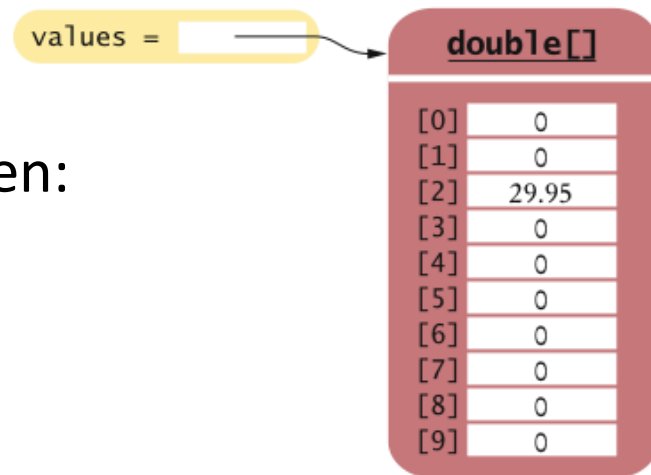
1. Deklarasikan array:

`double[] value = new double[10];`



2. Gunakan [] untuk mengakses elemen:

`value[2] = 29.95;`



Deklarasi Array

Table 1 Declaring Arrays

<pre>int[] numbers = new int[10];</pre>	An array of ten integers. All elements are initialized with zero.
<pre>final int NUMBERS_LENGTH = 10; int[] numbers = new int[NUMBERS_LENGTH];</pre>	It is a good idea to use a named constant instead of a “magic number”.
<pre>int valuesLength = in.nextInt(); double[] values = new double[valuesLength];</pre>	The length need not be a constant.
<pre>int[] squares = { 0, 1, 4, 9, 16 };</pre>	An array of five integers, with initial values.
<pre>String[] names = new String[3];</pre>	An array of three string references, all initially null.
<pre>String[] friends = { "Emily", "Bob", "Cindy" };</pre>	Another array of three strings.
<pre>double[] values = new int[10]</pre>	Error: You cannot initialize a double[] variable with an array of type int[].

Mengakses Elemen Array

- Bentuk Deklarasi `namaVariabelArray[nomorElemen];`
- Contoh:
`kota[0] = "Surabaya";`

ArrayKota.java

```
public class ArrayKota{  
    public static void main(String[] args){  
        String[] kota;           //deklarasi variabel array  
        kota = new String[3];    // membuat objek array  
  
        // mengisi elemen array  
        kota[0] = "Jakarta";  
        kota[1] = "Surabaya";  
        kota[2] = "Semarang";  
        // menampilkan elemen array  
        System.out.println(kota[0]);  
        System.out.println(kota[1]);  
        System.out.println(kota[2]);  
    }  
}
```

Pemberian Nilai Array Langsung

```
public class ArrayKota2{  
    public static void main(String[] args){  
        String[] kota = {"Jakarta", "Surabaya", "Semarang"};  
  
        // menampilkan elemen array  
        System.out.println(kota[0]);  
        System.out.println(kota[1]);  
        System.out.println(kota[2]);  
    }  
}
```

Mengetahui Jumlah Elemen Array

```
public class ArrayKota3{  
    public static void main(String[] args){  
        String[] kota = {"Jakarta", "Surabaya", "Semarang"};  
  
        // menampilkan elemen array  
        for(int i=0; i<kota.length; i++){  
            System.out.println(kota[i]);  
        }  
    }  
}
```

Array Multidimensi

```
class ArrayMultidimensi {  
    public static void main(String[] args) {  
        String[][] nama = {  
            {"Pak ", "Bu ", "Mbak"},  
            {"Joko", "Susi"}  
        };  
        System.out.println(nama[0][0] + nama[1][0]);  
        System.out.println(nama[0][1] + nama[1][1]);  
        System.out.println(nama[0][2] + nama[1][0]);  
    }  
}
```

Array multidimensi adalah array dari array , dengan konsep pengaksesan [**noBaris**][**noKolom**]

Latihan: Buat Array Multidimensi

1. Buat class **NegaraKota**
2. Buat array multidimensi untuk **nama negara** dan **ibukotanya**
3. Masukkan dalam list array:
nama negara = **Amerika, Inggris, Jepang, Perancis, Indonesia, Iran, Irak**
ibukota = **Teheran, Bekasi, Jakarta, Bantar Gebang, Tokyo**
4. Akses array dan tampilkan di layar sebagai berikut:
Ibukota Indonesia adalah Jakarta
Ibukota Jepang adalah Tokyo
Ibukota Iran adalah Teheran

Array Resizing

- Setelah array dibuat, array tidak dapat di-resize. Tapi, bisa dideklarasikan ulang dengan menggunakan referensi yang sama.
- Contoh :

```
int[] myArray = new int[6];  
myArray = new int[10];
```

Copy Array

`System.arraycopy()` method

- Java Programming language menyediakan spesial method pada `System` class, `arraycopy()`, untuk menyalin arrays.
- Contoh :

```
//source array
int src[] = {1,2,3,4,5,6};

//destination array
int dest[] = new int[src.length];
System.arraycopy(src,0,dest,0,src.length);

for (int i=0;i<dest.length;i++ ) {
    System.out.println(dest[i]);
}
```

ArrayList

ArrayList

- `ArrayList` class mengelola urutan object, yang **dapat bertambah dan berkurang** sesuai dengan keperluan
- `ArrayList` class menyediakan banyak method untuk berbagi keperluan, misalnya menambah dan menghapus elemen
- `ArrayList` adalah suatu **generic class**:
- `ArrayList<T>` mengumpulkan object yang bertipe T:

```
ArrayList<String> names = new ArrayList<String>();  
names.add("Emily");  
names.add("Bob");  
names.add("Cindy");
```

- `size` method untuk menghitung jumlah elemen

ArrayList

Syntax

To construct an array list: `new ArrayList<typeName>()`

To access an element: `arraylistReference.get(index)`
`arraylistReference.set(index, value)`

Example

Variable type

Variable name

An array list object of size 0

```
ArrayList<String> friends = new ArrayList<String>();
```

Use the
get and set methods
to access an element.

```
friends.add("Cindy");  
String name = friends.get(i);  
friends.set(i, "Harry");
```

The add method
appends an element to the array list,
increasing its size.

The index must be
 ≥ 0 and $< \text{friends.size}()$.



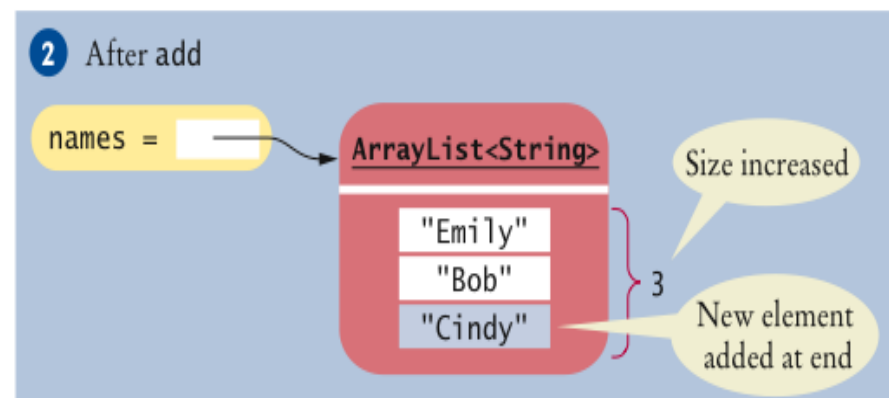
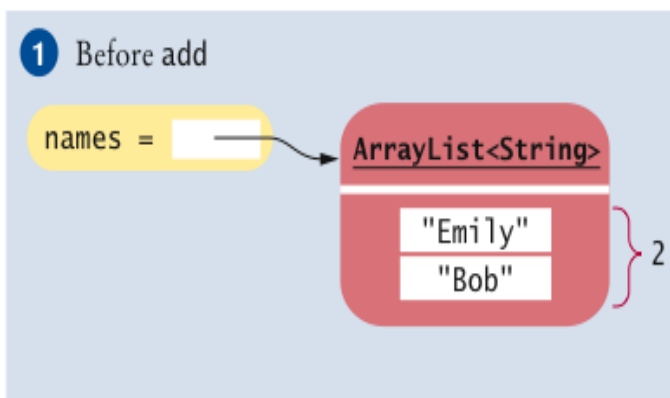
Menambahkan Elemen

- Untuk menambahkan sebuah elemen pada bagian akhir dari `ArrayList`, gunakan method `add` di bawah:

```
names.add("Emily");
```

```
names.add("Bob");
```

```
names.add("Cindy");
```



Menghapus Elemen

- Untuk menghapus elemen pada suatu indeks, menggunakan **method** `remove`:
`names.remove(1) ;`

Mendapatkan Nilai Elemen

- Untuk mendapatkan nilai elemen pada indeks, menggunakan metode `get`, dimana indeks dimulai dari 0

```
String name = names.get(2);  
//dapatkan elemen ketiga dari ArrayList
```

- Bila indeks keluar dari jangkauan, error akan keluar:

```
int i = names.size();  
name = names.get(i); // Error  
// legal index values are 0 ... i-1
```

Menambah Nilai Baru ke Elemen

- Untuk menambahkan nilai baru ke elemen, digunakan **method** `set`:
`names.set (2, "Carolyn");`

Menambah dan Menghapus Elemen

```
names.add("Emily");
```

```
names.add("Bob");
```

```
names.add("Cindy");
```

```
names.set(2, "Carolyn");
```

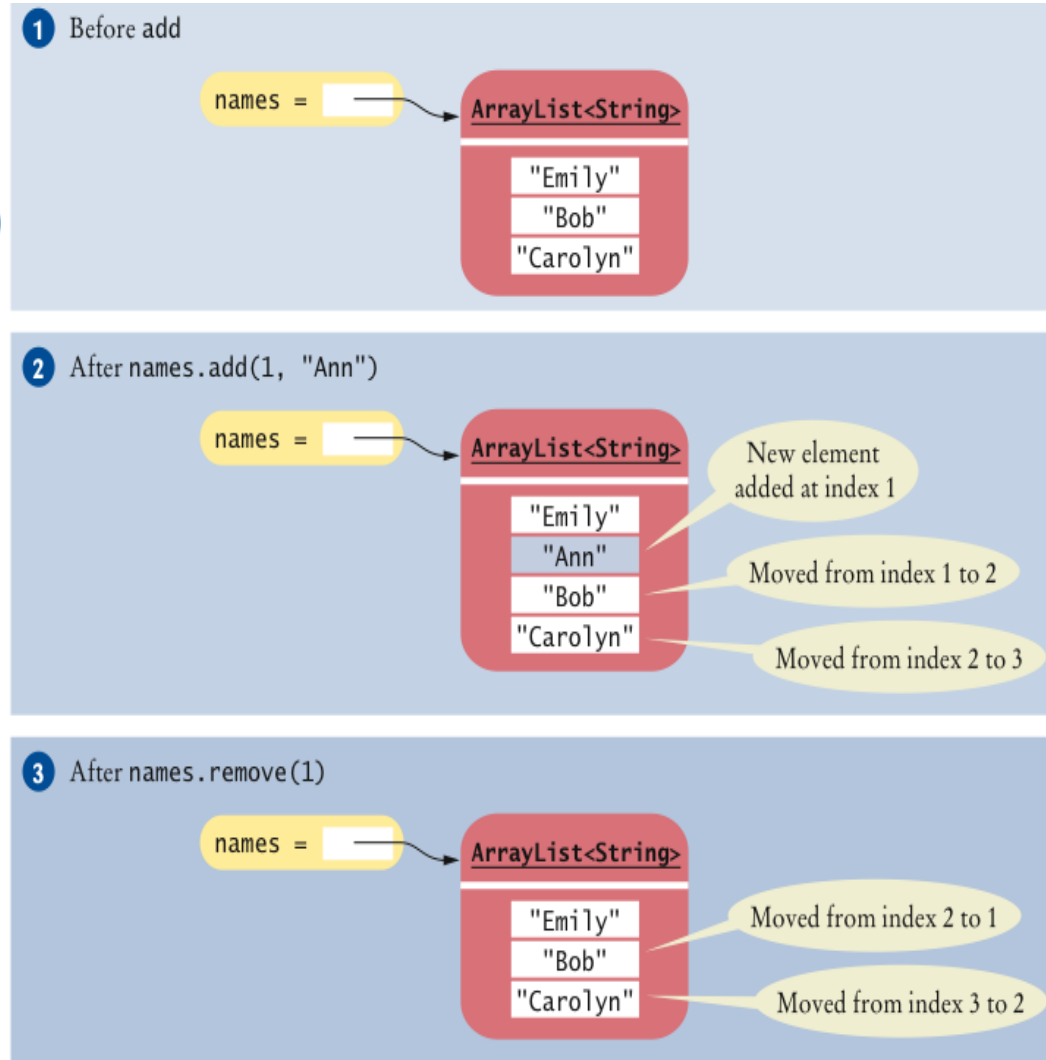
```
names.add(1, "Ann");
```

```
names.remove(1);
```

1

2

3



<code>ArrayList<String> names = new ArrayList<String>();</code>	Constructs an empty array list that can hold strings
<code>names.add("Ann");</code> <code>names.add("Cindy");</code>	Adds elements to the end
<code>System.out.println(names);</code>	Prints [Ann, Cindy]
<code>names.add(1, "Bob");</code>	Inserts an element at index 1. names is now [Ann, Bob, Cindy]
<code>names.remove(0);</code>	Removes the element at index 0. names is now [Bob, Cindy]
<code>names.set(0, "Bill");</code>	Replaces an element with a different value. names is now [Bill, Cindy]
<code>String name = names.get(i);</code>	Gets an element
<code>String last =</code> <code>names.get(names.size() - 1);</code>	Gets the last element

BankAccount.java

```
public class BankAccount {  
    private double balance;  
    private int accountNumber;  
  
    public BankAccount(int accountNumber){  
        balance = 0;  
        this.accountNumber = accountNumber;  
    }  
  
    public void deposit(double amount){  
        balance = balance + amount;  
    }  
  
    public void withdraw(double amount){  
        balance = balance - amount;  
    }  
  
    public int getAccountNumber(){  
        return accountNumber;  
    }  
  
    public double getBalance(){  
        return balance;  
    }  
}
```

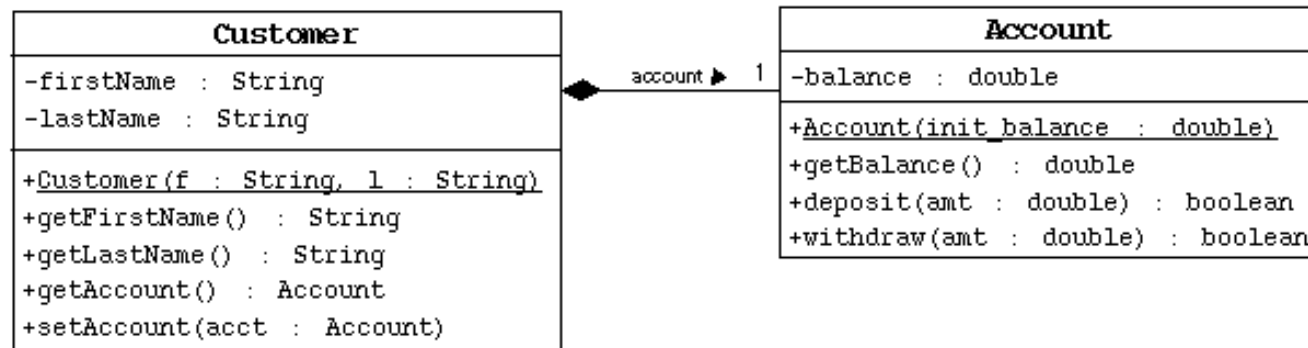
BankAccountArrayBeraksi.java

```
public class BankAccountArrayBeraksi{
    public static void main(String[] args) {
        ArrayList<BankAccount> accounts = new ArrayList<BankAccount>();
        accounts.add(new BankAccount(1001));
        accounts.add(new BankAccount(1015));
        accounts.add(new BankAccount(1729));
        accounts.add(1, new BankAccount(1008));
        accounts.remove(0);

        System.out.println("Size: " + accounts.size());
        System.out.println("Expected: 3");
        BankAccount first = accounts.get(0);
        System.out.println("First account number: " + first.getAccountNumber());
        System.out.println("Expected: 1008");
        BankAccount last = accounts.get(accounts.size() - 1);
        System.out.println("Last account number: " + last.getAccountNumber());
        System.out.println("Expected: 1729");
    }
}
```

Kode belum
lengkap, lengkapi
dengan import
java.util.ArrayList

Latihan



Exercise

Create the **Account** class in the file **Account.java**

- declare one private object attribute: **balance**; this attribute will hold the current (or "running") balance of the bank account
- declare a public constructor that takes one parameter (**init_balance**); that populates the **balance** attribute
- declare a public method **getBalance** that retrieves the current balance
- declare a public method **deposit** that adds the **amount** parameter to the current balance
- declare a public method **withdraw** that removes the **amount** parameter from the current balance

Create the **Customer** class in the file **Customer.java**

- declare three private object attributes: **firstName**, **lastName**, and **account**
- declare a public constructor that takes two parameters (**f** and **l**) that populate the object attributes
- declare two public accessors for the object attributes; these methods **getFirstName** and **getLastName** simply return the appropriate attribute
- declare the **setAccount** method to assign the **account** attribute
- declare the **account** method to retrieve the **account** attribute

Solution

Account Class :

```
1. public class Account {
2.     protected double balance;
3.     public Account(double bal) {
4.         balance = bal;
5.     }
6.     public double getBalance() {
7.         return balance;
8.     }
9.     public boolean deposit(double amount) {
10.        if (amount>0) {
11.            balance = balance + amount;
12.            return true;
13.        } else
14.            return false;
15.    }
16.    public boolean withdraw(double amount) {
17.        if ( balance >= amount ) {
18.            balance = balance - amount;
19.            return true;
20.        } else
21.            return false;
22.    }
23.}
```

Create the Account class in the file Account.java

- ❑ declare one private object attribute: balance; this attribute will hold the current (or "running") balance of the bank account
- ❑ declare a public constructor that takes one parameter (init_balance); that populates the balance attribute
- ❑ declare a public method getBalance that retrieves the current balance
- ❑ declare a public method deposit that adds the amount parameter to the current balance
- ❑ declare a public method withdraw that removes the amount parameter from the current balance

Account	
1	-balance : double
+Account(<u>init_balance</u> : double)	
+getBalance() : double	
+deposit(amt : double) : boolean	
+withdraw(amt : double) : boolean	

Solution

Customer Class :

```
1.  public class Customer {
2.      private String    firstName;
3.      private String    lastName;
4.      private Account[] accounts = new Account[5];
5.      private int numberOfAccounts = 0;

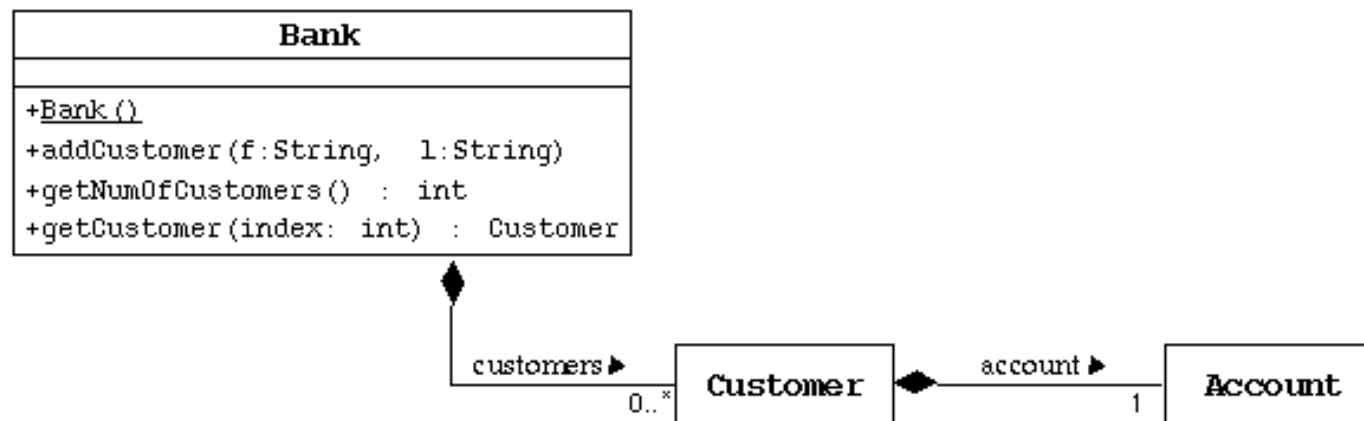
6.      public Customer(String f, String l) {
7.          firstName = f;
8.          lastName = l;
9.      }
10.     public String getFirstName() {
11.         return firstName;
12.     }
13.     public String getLastName() {
14.         return lastName;
15.     }
16.     public void setAccount(Account acct) {
17.         if (numberOfAccounts<5)
18.             accounts[numberOfAccounts++] = acct;
19.     }
20.     public Account getAccount(int account_index) {
21.         return accounts[account_index];
22.     }
23.     public int getNumOfAccounts() {
24.         return numberOfAccounts;
25.     }
26. }
```

Create the Customer class in the file Customer.java

- declare three private object attributes: firstName, lastName, and account
- declare a public constructor that takes two parameters (f and l) that populate the object attributes
- declare two public accessors for the object attributes; these methods getFirstName and getLastName simply return the appropriate attribute
- declare the setAccount method to assign the account attribute
- declare the account method to retrieve the account attribute

Customer
-firstName : String -lastName : String
+Customer(f : String, l : String) +getFirstName() : String +getLastName() : String +getAccount() : Account +setAccount(acct : Account)

Exercise



Exercise

Create the Bank class in the file Bank.java

- Add two attributes to the Bank class: customers (an array of Customer objects) and numberOfCustomers (an integer that keeps track of the next customers array index).
- Add a public constructor that initializes the customers array with some appropriate maximum size (at least bigger than 5).
- Add the addCustomer method. This method must construct a new Customer object from the parameters (first name, last name) and place it on the customers array. It must also increment the numberOfCustomers attribute.
- Add the getNumOfCustomers accessor method, which returns the numberOfCustomers attribute.
- Add the getCustomer method. This method returns the customer associated with the given index parameter.

Solution

Bank Class :

```
1.    public class Bank {
2.        private Customer[] customers = new Customer[5];
3.        private int numberOfCustomers;

4.        public Bank() {
5.            numberOfCustomers = 0;
6.        }
7.        public void addCustomer(String f, String l) {
8.            if (numberOfCustomers<5)
9.                customers[numberOfCustomers++] = new Customer(f, l);
10.        }
11.        public int getNumOfCustomers() {
12.            return numberOfCustomers;
13.        }
14.        public Customer getCustomer(int customer_index) {
15.            return customers[customer_index];
16.        }
17.    }
```

Tambahkan sebuah kelas yang memiliki method main tempat contoh pembuatan objek array dan pemanggilan method Dari kelas-kelas tersebut
(Contoh bebas, kreatifitas masing-masing)

TERIMA KASIH