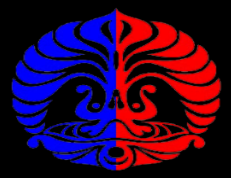
# Array

Alfan

# Arrays

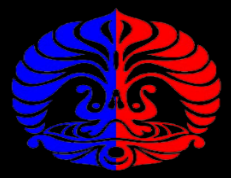
- **Misalkan kita ingin menyimpan 100 buah integer**
  - Buat 100 variabel bertipe integer
    - `int angka1 = 0;`
    - `int angka2 = 1;`
    - Dst
- **Tidak efisien!**
- **Kita dapat menyimpannya dalam sebuah variabel dengan tipe `array`**
- **Array adalah sebuah object juga**



Program yang menerima sejumlah angka **1 – 500**, dan menghitung **kemunculan** angka-angka tersebut !

```
int angka1   = 0;
int angka2   = 0;
...
int angka500 = 0;

int input = in.nextInt();
while (!selesai) {
      if (input == 1) {
            angka1++;
      } else if (input == 2) {
            angka2++;
      } ...
        ...
      } else if (input == 499) {
            angka499++;
      }
      ...
}
```



# Deklarasi array

```
double[] array1;

int[] values;

String[] names;

Point[] points;
```

# Membentuk array

```
double[] array1 = new double[5];

int[] values = new int[10];

String[] names = new String[5];

Point[] points = new Point[5];
```

# Membentuk array

```
double[] array1 = {1.0, 2.5, 2.3, 5.5};

int[] values = {3, 4, 5, 6, 10, 34};
```

Seandainya panjang Array konstan, lebih baik gunakan **KONSTANTA** ☺
Daripada menggunakan **magic number**.

```
final int NUMBERS_LENGTH = 10;

int[] values = new int[NUMBERS_LENGTH];
```

Panjang dari array bisa juga ditentukan dari **masukan user**.

```
Scanner in = new Scanner(System.in);

int valuesLength = in.nextInt();

double[] values = new double[valuesLength];
```

Apa yang terjadi di memori dari keempat statement berikut ?

```
String[] names = {“ani”, “budi”, “ceppy”};

String[] args  = new String[3];

int[] values = new int[3];

int[] arr1 = {5, 2, 9};
```

Apa yang terjadi ?

```
double val = 1000;                    //OK ?

double[] values  = new int[10];       //OK ?
```

Cara akses array (mengisi dan melihat nilai)

```
int[] values = new int[3];

values[0] = 45;
values[1] = 10;
Values[2] = 4;

System.out.println(“elemen pertama : ” + values[0]);
System.out.println(“elemen kedua : ” + values[1]);
System.out.println(“elemen ketiga : ” + values[2]);
```

Output ?

```
int[] values = {4, 5, 89};
System.out.println(values[2]);   // ?


int[] numbers;
System.out.println(numbers[2]);   // ?
```

Bagaimana cara mengetahui **panjang** array ?

```
int[] values = new int[10];
int panjangArray = values.length;   //10
```

```
String str = “abcdef”;
int panjangString = str.length();   //6
```

Output ?

```
String[] values = {“ani”, “bobby”, “coky”};
int i = 2;
System.out.println(values[i]);      // ?


int[] numbers = {4, 5, 34, 7, 8};
System.out.println(numbers[5]);     // ?
```

Output ?

```
int[] numbers = {4, 5, 34, 7, 8};
System.out.println(numbers[numbers.length]);    // ?
```

Output ?

```
double[] values  = {3.4, 5.5, 8.4, 5.5};

double[] values2 = values;

values2[1] = 7.0;

System.out.println(values[1]);
```

Panjang Array, sekali dihidupkan, akan bersifat **statik.**
**apa yang terjadi di memori ?**

```
double[] values  = {3.4, 5.5, 8.4, 5.5};

values = new double[10];

System.out.println(values[0]);
```

Memproses Array akan lebih natural jika menggunakan **loop**

```
double[] values  = {3.4, 5.5, 8.4, 5.5};

for (int i = 0; i < values.length; i++) {

     //proses ...

}
```

Apakah isi dari array **values** berikut ?

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

for (int i = 0; i < values.length; i++) {

     values[i] = i * i;

}
```

Buatlah program yang menjumlahkan semua elemen pada array of integers.

```
int[] values  = new int[10];
//kode untuk mengisi array

//tuliskan kode Anda di bawah...
```

Buatlah program yang mencetak semua elemen pada array of floating points secara terbalik !

```
double[] values  = new double[10];
//kode untuk mengisi array

//tuliskan kode Anda di bawah...
```

Lengkapi *static method* berikut untuk menghitung banyaknya elemen positif pada array of integers.

```
public static int numPositif(int[] values) {

    //tuliskan kode disini...

}
```

Implementasikan static method yang menerima array of integers, dan mengembalikan array yang sama. Tetapi, elemen yang bernilai **n** diganti dengan elemen **m**.

```
public static int[] changeNM(int[] values, int n,
int m) {

    //tuliskan kode disini...

}
```

Implementasikan *static method* yang menerima array of String dan sebuah String **str**. *Static method* akan mengembalikan banyaknya kemunculan **str** pada array tersebut !

```
public static int countStr(String[] names, String str) {

      //tuliskan kode disini...

}
```

Buatlah program yang menerima masukan sebuah integer positif N, kemudian diikuti dengan mengisi setiap elemen pada array berukuran N tersebut.

Di akhir, program akan menampilkan semua elemen pada array tersebut !

contoh: di slide berikutnya.

```
Masukkan ukuran array : 5
Masukkan isi elemen ke-1 : 4
Masukkan isi elemen ke-2 : 6
Masukkan isi elemen ke-3 : 70
Masukkan isi elemen ke-4 : 0
Masukkan isi elemen ke-5 : 2

4 6 70 0 2
```

```
System.out.print(“Masukkan ukuran array : ”);
int arraySize = in.nextInt();
int[] values  = new int[arraySize];

for (int i = 0; i < arraySize; i++) {
	System.out.print(“Masukkan isi elemen ke-”+(i+1)+” : ”);
	values[i] = in.nextInt();
}

//tampilkan isi array
for(int i = 0; i < arraySize; i++) {
	System.out.print(values[i]);
}
```

# Arrays of Objects

- Kita dapat membuat array yang memiliki tipe ***objects***

```
BankAccount[] accounts = new BankAccount[10];
accounts[3] = new BankAccount(…);
```

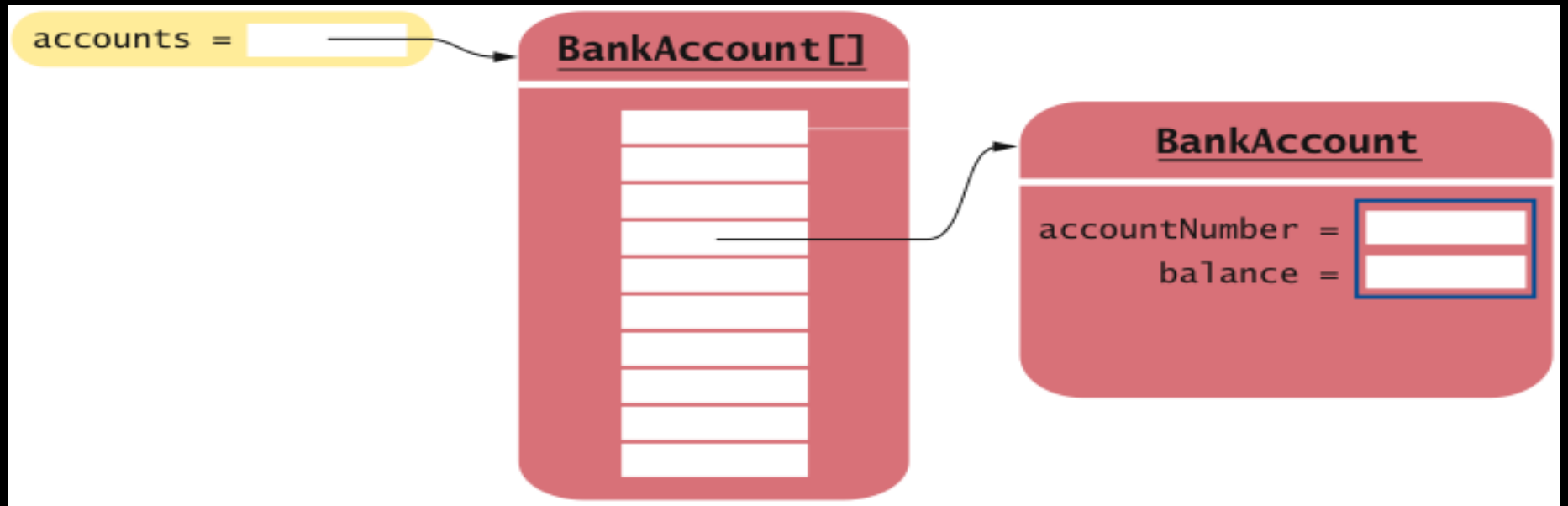
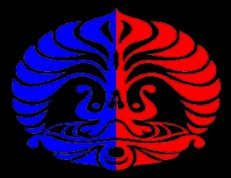



**Figure 4** Reorganizing Parallel Arrays into an Array of Objects



# Don’t: Parallel Arrays!

- **Jangan memisahkan data yang merupakan suatu kesatuan di array yang berbeda.**
  - Gunakan arrays of objects!

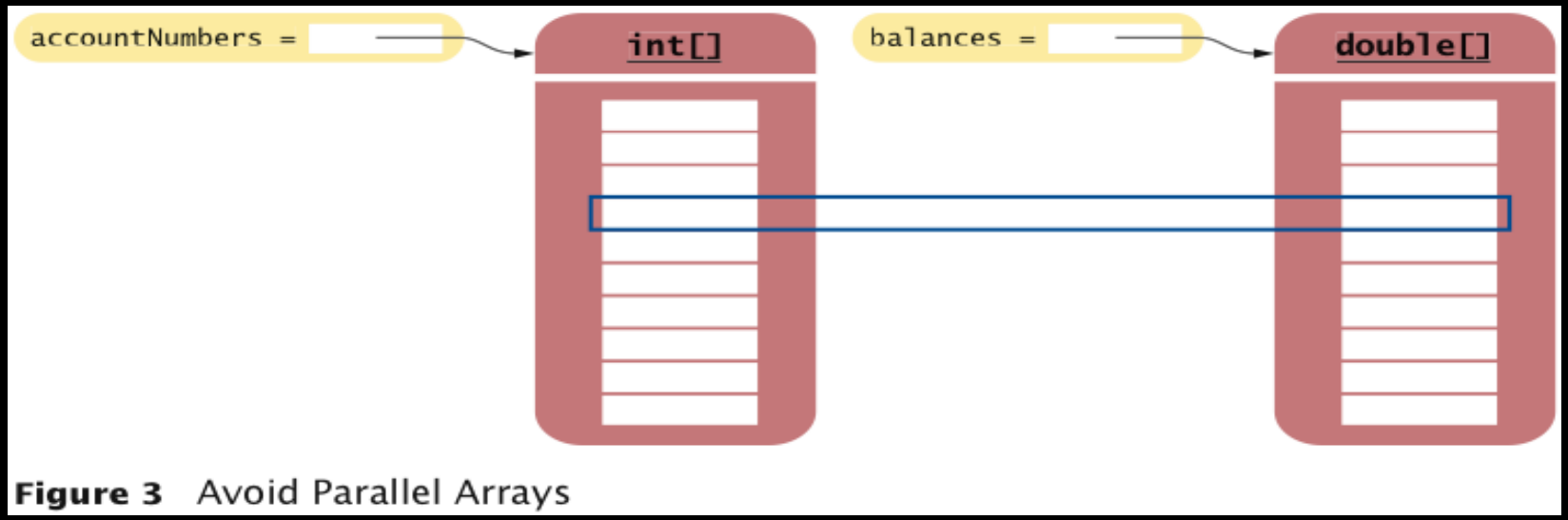
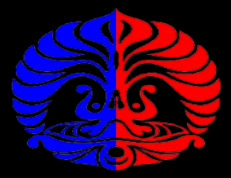



**Figure 3** Avoid Parallel Arrays

- **Why?**
  - Apabila kita ingin menambahkan `accountName`, hanya class `BankAccount` saja yang perlu diubah



To be continued...

Sementara, berlanjut ke slide yang ada di Scele dahulu

Gambarkan kondisi akhir **ArrayList** !

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

names.add(“dodi”);
names.add(“rudi”);
System.out.println(names.get(1));
names.add(1, “ani”);
names.add(“anto”);
names.set(0, “rani”);

System.out.println(names);
```

Gambarkan kondisi akhir **ArrayList** !

```
ArrayList<Integer> values = new ArrayList<Integer>();

values.add(5);
values.add(8);
values.remove(0);
values.add(0, 3);
values.add(3);
values.remove(1);
values.set(0, 2);
System.out.println(values.size());
```

```
Masukkan isi elemen ke-1 : 4
Masukkan isi elemen ke-2 : 6
Masukkan isi elemen ke-3 : 70
Masukkan isi elemen ke-4 : 0
Masukkan isi elemen ke-5 : 2
Masukkan isi elemen ke-6 : -1

2 0 70 6 4
```

Buatlah program yang menjumlahkan semua elemen pada arraylist.

```
ArrayList<Integer> values = new ArrayList<Integer>();
//kode untuk mengisi arraylist

//tuliskan kode Anda di bawah...
```

Lengkapi *static method* berikut untuk menghitung banyaknya elemen ganjil pada arraylist

```
public static int numGanjil(ArrayList<Integer> values) {

      //tuliskan kode disini...

}
```

Lengkapi *static method* berikut untuk mengembalikan elemen yang paling kecil pada arraylist

```
public static int min(ArrayList<Integer> values) {

      //tuliskan kode disini...

}
```

Lengkapi *static method* berikut untuk mengembalikan indeks yang berisi elemen yang paling kecil pada arraylist

```
public static int indeksMin(ArrayList<Integer> values) {

      //tuliskan kode disini...

}
```

```
public class KoleksiNilai {
	private int[] values = new int[10];

	...

	//jika target ditemukan pada kumpulan nilai,
	//kembalikan indeksnya, jika tidak ada, kembalikan -1
	public int search(int target) {
		...
	}

}
```

```
public static void main(String[] args) {
      int[] arr = {1,2,3,4,5};
      System.out.println(search(arr, 3)); //2
}

//sama seperti sebelumnya, kali ini search adalah
//static method
public static int search(int[] vals, int target) {

      //...

}
```