Dasar Dasar Pemrograman 2

Acuan: Introduction to Java Programming and Data Structure, Bab 7 dan Bab 8

Sumber Slide: Liang,

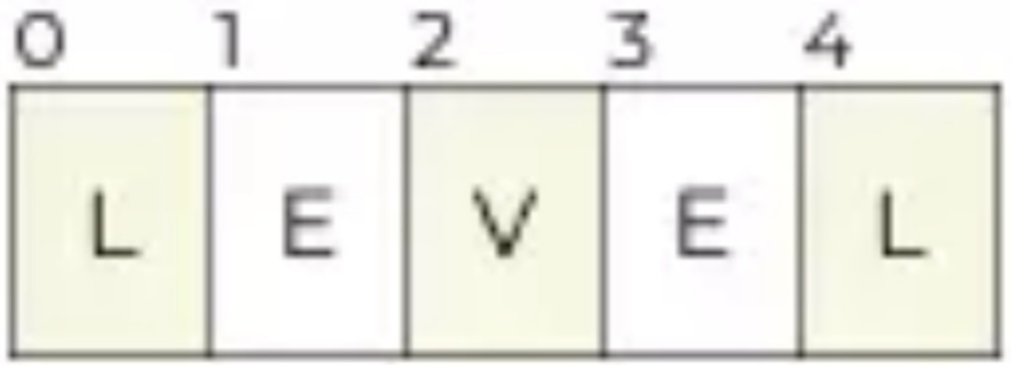
Dimodifikasi untuk Fasilkom UI oleh Ade Azurat

Topik: Array

Ingat Palindrom

- Kita melakukan pengecekan untuk setiap index dari elemen (char) sebuah string.

- Kita bisa memeriksa,
- Kita bisa menukar-nukar
- Kita bisa menyimpan element dalam kesatuan.
  Hmm, ini maksudnya apa ya?



Motivasi Array

- Kita bisa menyimpan banyak data dalam kesatuan.
- Ambil contoh nanti pada TP2, kita diminta menyimpan nama pemain.
- Kita bisa menyimpan dalam beberapa variable seperti:

```
String pemain1 = “Amir”;
String pemain2 = “Budi”;
String pemain3 = “Cica”;
```

Namun bagaimana bila jumlah nya banyak, atau bagaimana kalau kita tidak tahu persis berapa jumlah nya? Ini salah satu manfaat Array!

Objectives

- To describe why arrays are necessary in programming (§7.1).
- To declare array reference variables and create arrays (§§7.2.1–7.2.2).
- To obtain array size using **arrayRefVar.length** and know default values in an array (§7.2.3).
- To access array elements using indexes (§7.2.4).
- To declare, create, and initialize an array using an array initializer (§7.2.5).
- To program common array operations (displaying arrays, summing all elements, finding the minimum and maximum elements, random shuffling, and shifting elements) (§7.2.6).
- To simplify programming using the foreach loops (§7.2.7).
- To apply arrays in application development (**AnalyzeNumbers**, **DeckOfCards**) (§§7.3–7.4).
- To copy contents from one array to another (§7.5).
- To develop and invoke methods with array arguments and return values (§§7.6–7.8).
- To define a method with a variable-length argument list (§7.9).
- To search elements using the linear (§7.10.1) or binary (§7.10.2) search algorithm.
- To sort an array using the selection sort approach (§7.11).
- To use the methods in the **java.util.Arrays** class (§7.12).
- To pass arguments to the main method from the command line (§7.13).

Objectives

- ❑To give examples of representing data using two-dimensional arrays (§8.1).
- ❑To declare variables for two-dimensional arrays, create arrays, and access array elements in a two-dimensional array using row and column indexes (§8.2).
- ❑To program common operations for two-dimensional arrays (displaying arrays, summing all elements, finding the minimum and maximum elements, and random shuffling) (§8.3).
- ❑To pass two-dimensional arrays to methods (§8.4).
- ❑To write a program for grading multiple-choice questions using two-dimensional arrays (§8.5).
- ❑To solve the closest-pair problem using two-dimensional arrays (§8.6).
- ❑To check a Sudoku solution using two-dimensional arrays (§8.7).
- ❑To use multidimensional arrays (§8.8).

FAKULTAS
ILMU
KOMPUTER

Dasar Dasar Pemrograman 2

Acuan: Introduction to Java Programming and Data Structure, Bab 7 dan Bab 8

Sumber Slide: Liang,

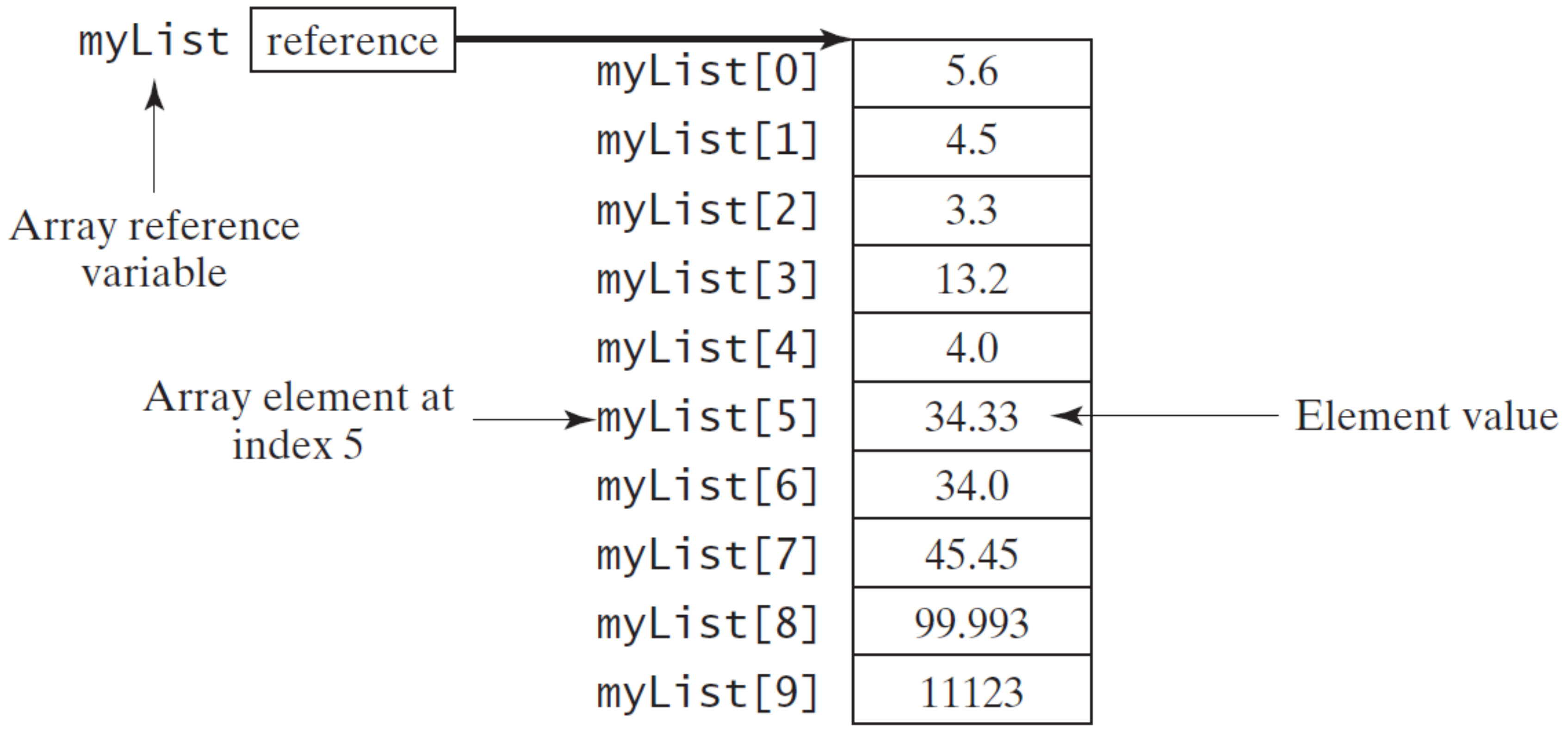
Dimodifikasi untuk Fasilkom UI oleh Ade Azurat

Topik: Single Array

Introducing Arrays

Array is a data structure that represents a collection of the same types of data.

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



Declaring Array Variables

- `datatype[] arrayRefVar;`

  Example:

  ```
  double[] myList;
  ```

- `datatype arrayRefVar[];` // This style is allowed, but not preferred

  Example:

  ```
  double myList[];
  ```

Creating Arrays

```
arrayRefVar = new datatype[arraySize];
```

Example:

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

`myList[0]` references the first element in the array.

`myList[9]` references the last element in the array.

Declaring and Creating in One Step

- `datatype[] arrayRefVar = new datatype[arraySize];`

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

- `datatype arrayRefVar[] = new datatype[arraySize];`

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

The Length of an Array

Once an array is created, its size is fixed. It cannot be changed. You can find its size using

```
arrayRefVar.length
```

For example,

```
myList.length returns 10
```

Default Values

When an array is created, its elements are assigned the default value of

0 for the numeric primitive data types,
'\u0000' for char types, and
false for boolean types.

Indexed Variables

The array elements are accessed through the index. The array indices are *0-based*, i.e., it starts from 0 to arrayRefVar.length-1.

Each element in the array is represented using the following syntax, known as an *indexed variable*:

```
arrayRefVar[index];
```

Using Indexed Variables

After an array is created, an indexed variable can be used in the same way as a regular variable.

For example, the following code adds the value in myList[0] and myList[1] to myList[2].

```
myList[2] = myList[0] + myList[1];
```

Array Initializers

Declaring, creating, initializing in one step:

```
double[] myList = {1.9, 2.9, 3.4, 3.5};
```

This shorthand syntax must be in one statement.

FACULTY OF COMPUTER SCIENCE

Declaring, creating, initializing Using the Shorthand Notation

```
double[] myList = {1.9, 2.9, 3.4, 3.5};
```

This shorthand notation is equivalent to the following statements:

```
double[] myList = new double[4];
myList[0] = 1.9;
myList[1] = 2.9;
myList[2] = 3.4;
myList[3] = 3.5;
```

CAUTION

Using the shorthand notation, you have to declare, create, and initialize the array all in one statement.

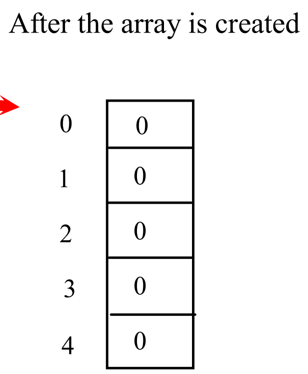
Splitting it would cause a syntax error. For example, the following is wrong:

```
double[] myList;

myList = {1.9, 2.9, 3.4, 3.5};
```

Trace Program with Arrays

Declare array variable values, create an array, and assign its reference to values

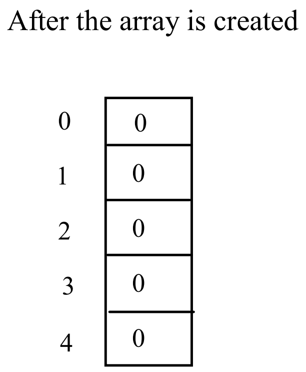
```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```



Trace Program with Arrays

i becomes 1

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

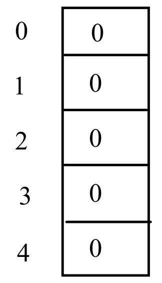


Trace Program with Arrays

i (=1) is less than 5

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

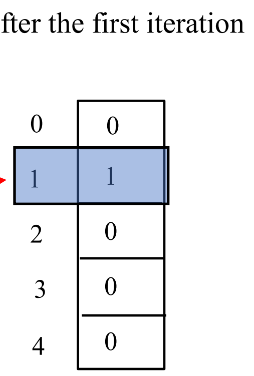
After the array is created



Trace Program with Arrays

After this line is executed, value[1] is 1

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

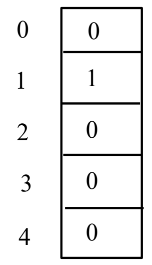


Trace Program with Arrays

After i++, i becomes 2

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the first iteration



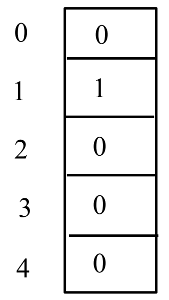
0	0
1	1
2	0
3	0
4	0

Trace Program with Arrays

i (= 2) is less than 5

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the first iteration



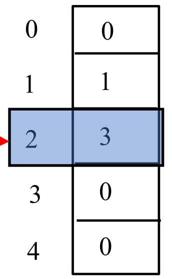
0	0
1	1
2	0
3	0
4	0

Trace Program with Arrays

After this line is executed,
values[2] is 3 (2 + 1)

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the second iteration



0	0
1	1
2	3
3	0
4	0

Trace Program with Arrays

After this, i becomes 3.

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the second iteration

0	0
1	1
2	3
3	0
4	0

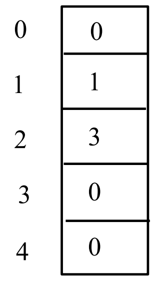
FACULTY OF COMPUTER SCIENCE

Trace Program with Arrays

i (=3) is still less than 5.

```
public class Test {
  public static void main(String[]   gs) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

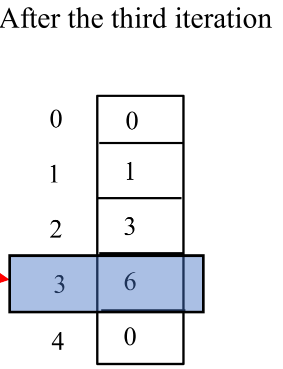
After the second iteration



Trace Program with Arrays

After this line, values[3] becomes 6 (3 + 3)

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```



Trace Program with Arrays

After this, i becomes 4

```
public class Test {
  public static void main(String[] ar
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the third iteration

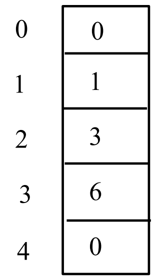
0	0
1	1
2	3
3	6
4	0

Trace Program with Arrays

i (=4) is still less than 5

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the third iteration

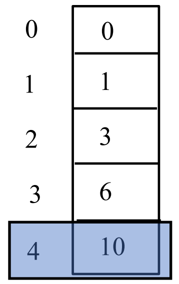


Trace Program with Arrays

After this, values[4] becomes 10 (4 + 6)

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the fourth iteration

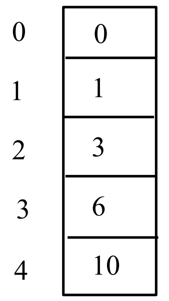


Trace Program with Arrays

After i++, i becomes 5

```
public class Test {
  public static void main(String[] ar
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

After the fourth iteration



Trace Program with Arrays

i (=5) < 5 is false. Exit the loop

```
public class Test {
  public static void main     ing[] args) {
    int[] values = new ir  .[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```

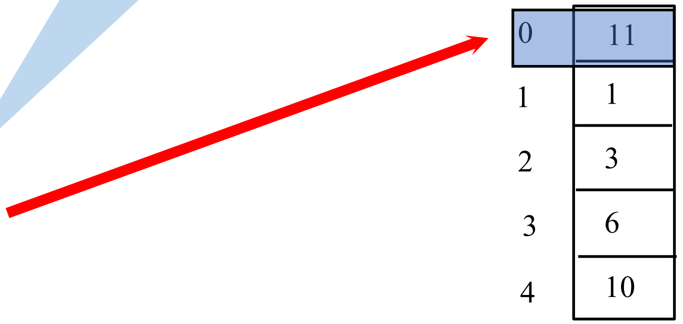
After the fourth iteration

0	0
1	1
2	3
3	6
4	10

Trace Program with Arrays

After this line, values[0] is 11 (1 + 10)

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```



Proses tracing ini penting untuk memahami cara kerja dan menguji logika berfikir kita dengan eksekusi komputer.

Coba lakukan trace serupa dengan ***debugger***!

Processing Arrays

See the examples in the text.

1. (Initializing arrays with input values)
2. (Initializing arrays with random values)
3. (Printing arrays)
4. (Summing all elements)
5. (Finding the largest element)
6. (Finding the smallest index of the largest element)
7. (*Random shuffling*)
8. (*Shifting elements*)

Initializing arrays with input values

```
java.util.Scanner input = new java.util.Scanner(System.in);
System.out.print("Enter " + myList.length + " values: ");
for (int i = 0; i < myList.length; i++)
  myList[i] = input.nextDouble();
```

Initializing arrays with random values

```
for (int i = 0; i < myList.length; i++) {
  myList[i] = Math.random() * 100;
}
```

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FACULTY OF COMPUTER SCIENCE

Printing arrays

```
for (int i = 0; i < myList.length; i++) {
  System.out.print(myList[i] + " ");
}
```

Summing all elements

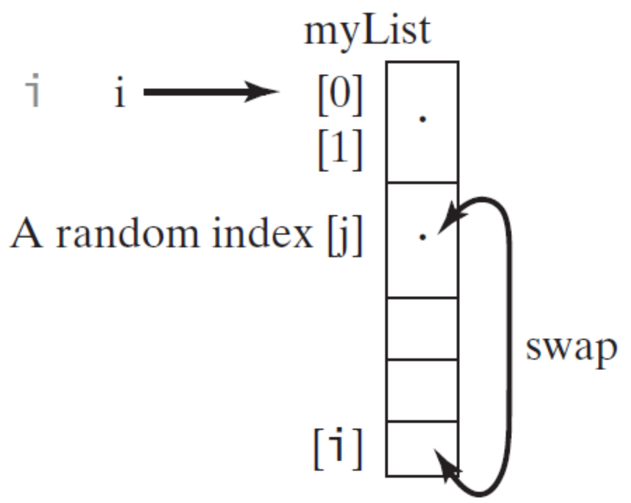
```
double total = 0;
for (int i = 0; i < myList.length; i++) {
  total += myList[i];
}
```

Finding the largest element

```
double max = myList[0];
for (int i = 1; i < myList.length; i++) {
  if (myList[i] > max) max = myList[i];
}
```

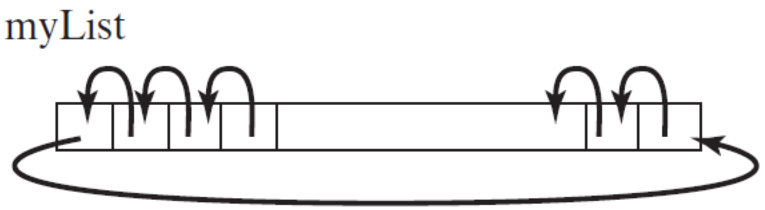
Random shuffling

```
for (int i = myList.length - 1; i > 0; i--) {
  // Generate an index j randomly with 0 <= j <= i
  int j = (int)(Math.random()
    * (i + 1));

  // Swap myList[i] with myList[j]
  double temp = myList[i];
  myList[i] = myList[j];
  myList[j] = temp;
}
```



Shifting Elements

```
double temp = myList[0]; // Retain the first element

// Shift elements left
for (int i = 1; i < myList.length; i++) {
  myList[i - 1] = myList[i];
}

// Move the first element to fill in the last position
myList[myList.length - 1] = temp;
```



Enhanced for Loop (for-each loop)

JDK 1.5 introduced a new for loop that enables you to traverse the complete array sequentially without using an index variable. For example, the following code displays all elements in the array myList:

```
for (double value: myList)
  System.out.println(value);
```

In general, the syntax is

```
for (elementType value: arrayRefVar) {
  // Process the value
}
```

You still have to use an index variable if you wish to traverse the array in a different order or change the elements in the array.

Opening Problem

https://replit.com/@AdeAzurat/DDP2-Pekan05-Array#AnalyzeNumbers.java

Read one hundred numbers,
compute their average,
and find out how many numbers are above the average.

Example:
8 9 4 3 0 2 5 1 6 7
Average: 4.5
Numbers above the average: 5 numbers

Kita akan sulit membuat implementasinya kecuali kita menyimpan data tersebut dalam array.

FACULTY OF COMPUTER SCIENCE

Problem: Deck of Cards

https://replit.com/@AdeAzurat/DDP2-Pekan05-Array#DeckOfCards.java

The problem is to write a program that picks four cards randomly from a deck of 52 cards. All the cards can be represented using an array named deck, filled with initial values 0 to 51, as follows:

```
int[] deck = new int[52];
// Initialize cards
for (int i = 0; i < deck.length; i++)
  deck[i] = i;
```

Problem: Deck of Cards

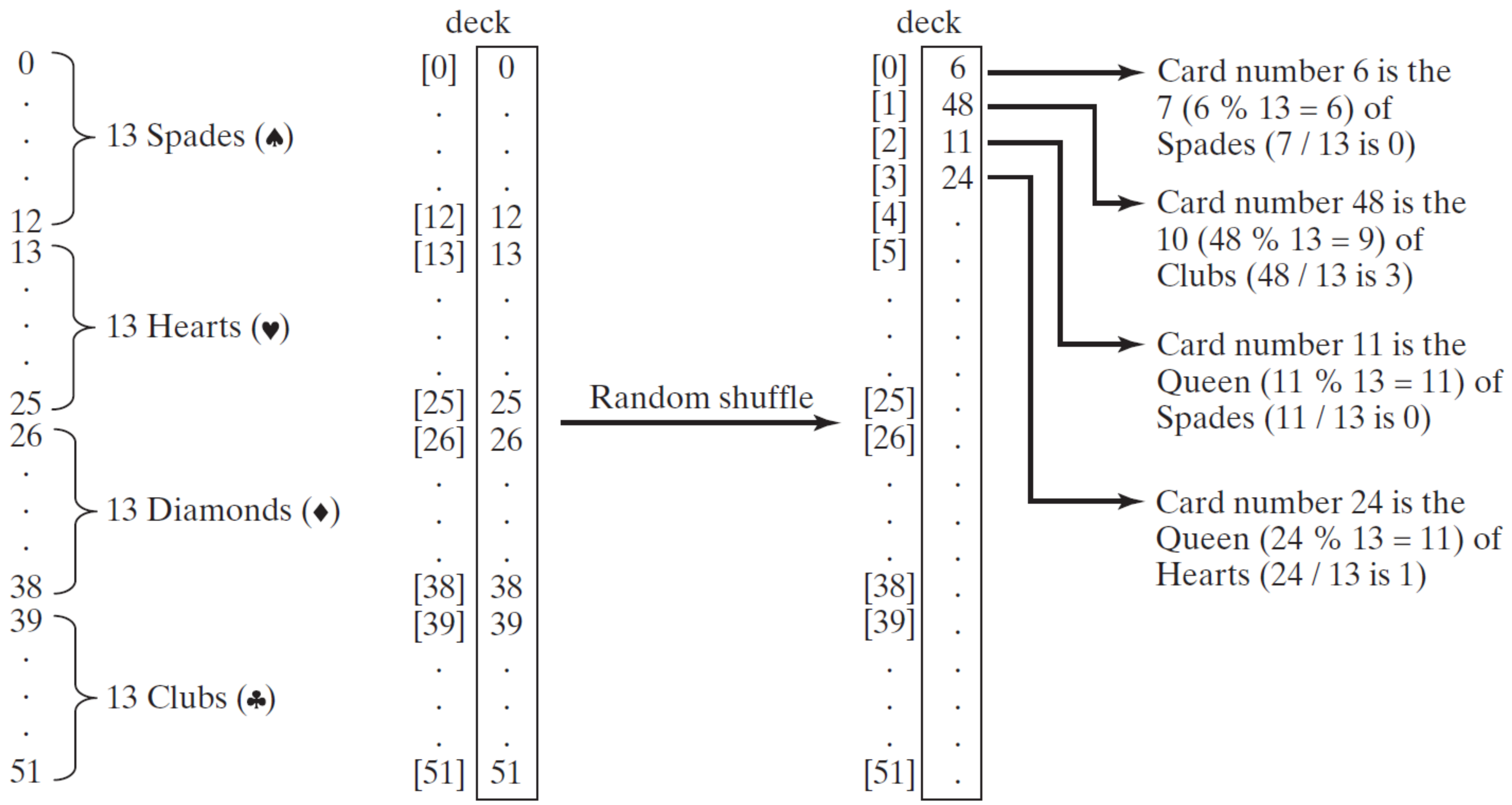
https://replit.com/@AdeAzurat/DDP2-Pekan05-Array#DeckOfCards.java

$$\text{cardNumber / 13} = \begin{cases} 0 \longrightarrow \text{Spades} \\ 1 \longrightarrow \text{Hearts} \\ 2 \longrightarrow \text{Diamonds} \\ 3 \longrightarrow \text{Clubs} \end{cases}$$

$$\text{cardNumber \% 13} = \begin{cases} 0 \longrightarrow \text{Ace} \\ 1 \longrightarrow 2 \\ . \\ . \\ 10 \longrightarrow \text{Jack} \\ 11 \longrightarrow \text{Queen} \\ 12 \longrightarrow \text{King} \end{cases}$$

Problem: Deck of Cards

https://replit.com/@AdeAzurat/DDP2-Pekan05-Array#DeckOfCards.java



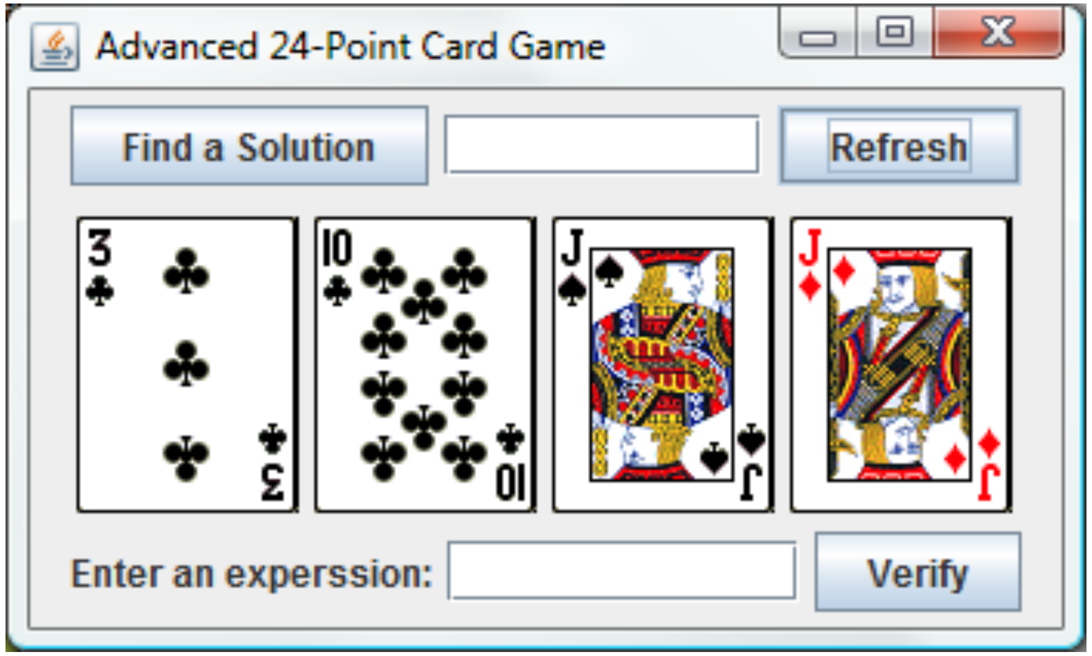
0 . . . 12 13 Spades (♠)
13 . . . 25 13 Hearts (♥)
26 . . . 38 13 Diamonds (♦)
39 . . . 51 13 Clubs (♣)
deck
[0] 0
[12] 12
[13] 13
[25] 25
[26] 26
[38] 38
[39] 39
[51] 51
Random shuffle
deck
[0] 6
[1] 48
[2] 11
[3] 24
[4]
[5]
[25]
[26]
[38]
[39]
[51]
Card number 6 is the 7 (6 % 13 = 6) of Spades (7 / 13 is 0)
Card number 48 is the 10 (48 % 13 = 9) of Clubs (48 / 13 is 3)
Card number 11 is the Queen (11 % 13 = 11) of Spades (11 / 13 is 0)
Card number 24 is the Queen (24 % 13 = 11) of Hearts (24 / 13 is 1)

Problem: Deck of Cards

https://replit.com/@AdeAzurat/DDP2-Pekan05-Array#DeckOfCards.java

This problem builds a foundation for future more interesting and realistic applications:

See Exercise 22.15.



enum

- To define a set of value, we can use enumerated type.
- In Java, we can use enum

```
/** The suits a card can belong to */
public enum Suit {CLUBS, HEARTS, SPADES, DIAMONDS};

private Suit suit;
suit = Suit.CLUBS;
```

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER

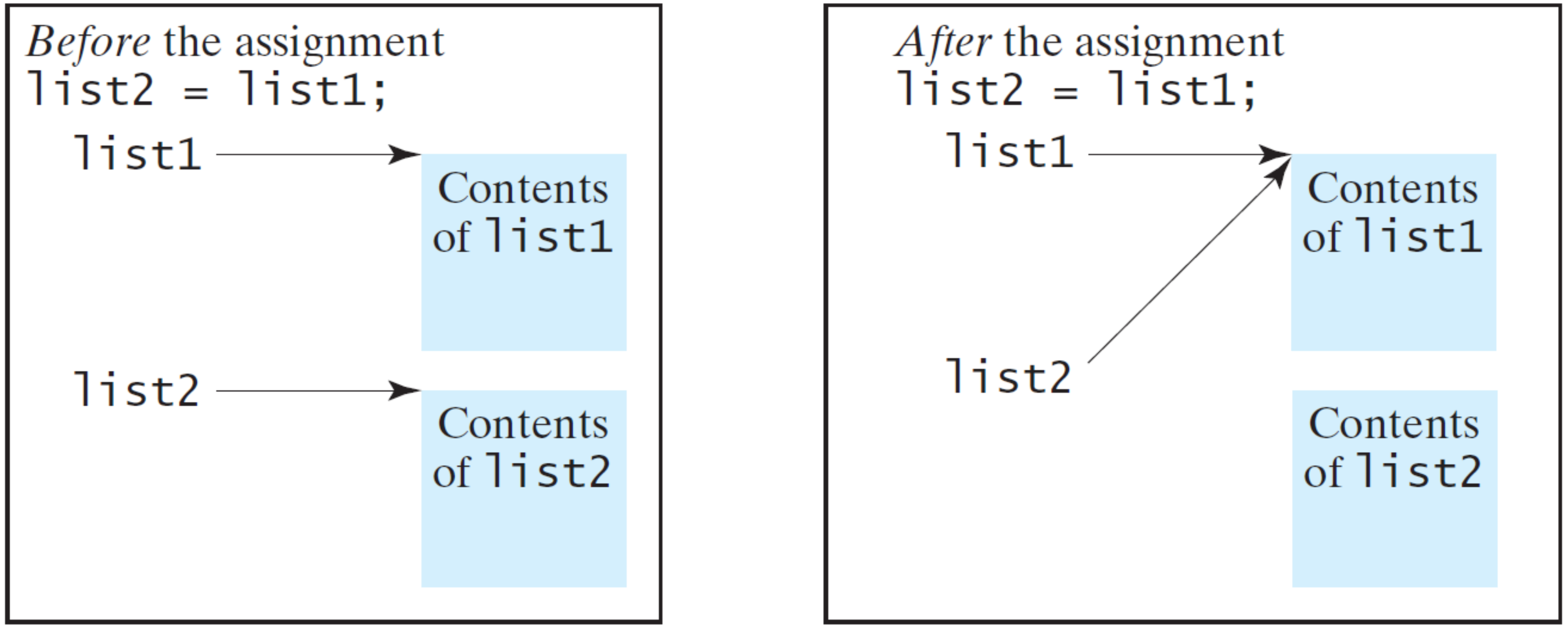
Ada Pertanyaan?

- ❑ apakah datatype array?
- ❑ deklarasi membuat array
- ❑ menginisiasi array, mencopy array
- ❑ indeks array
- ❑ tracing program menggunakan array
- ❑ Enhanced Loop
- ❑ Latihan dan Algoritma terkait array

Copying Arrays

Often, in a program, you need to duplicate an array or a part of an array. In such cases you could attempt to use the assignment statement (=), as follows:

list2 = list1;



FACULTY OF COMPUTER SCIENCE

Copying Arrays

Using a loop:

```
int[] sourceArray = {2, 3, 1, 5, 10};
int[] targetArray = new
 int[sourceArray.length];


for (int i = 0; i < sourceArrays.length; i++)
   targetArray[i] = sourceArray[i];
```

The `arraycopy` Utility

```
arraycopy(sourceArray, src_pos, targetArray, tar_pos, length);
```

Example:

```
System.arraycopy(sourceArray, 0, targetArray, 0, sourceArray.length );
```

Passing Arrays to Methods

```
public static void printArray(int[] array) {
  for (int i = 0; i < array.length; i++) {
    System.out.print(array[i] + " ");
  }
}
```

Invoke the method

```
int[] list = {3, 1, 2, 6, 4, 2};
printArray(list);
```

Invoke the method

```
printArray(new int[]{3, 1, 2, 6, 4, 2});
```

Anonymous array

Anonymous Array

The statement

```
printArray(new int[]{3, 1, 2, 6, 4, 2});
```

creates an array using the following syntax:

```
new dataType[]{literal0, literal1, ..., literalk};
```

There is no explicit reference variable for the array. Such array is called an *anonymous array*.

Pass By Value

Java uses *pass by value* to pass arguments to a method. There are important differences between passing a value of variables of ***primitive data types*** and ***passing arrays***.

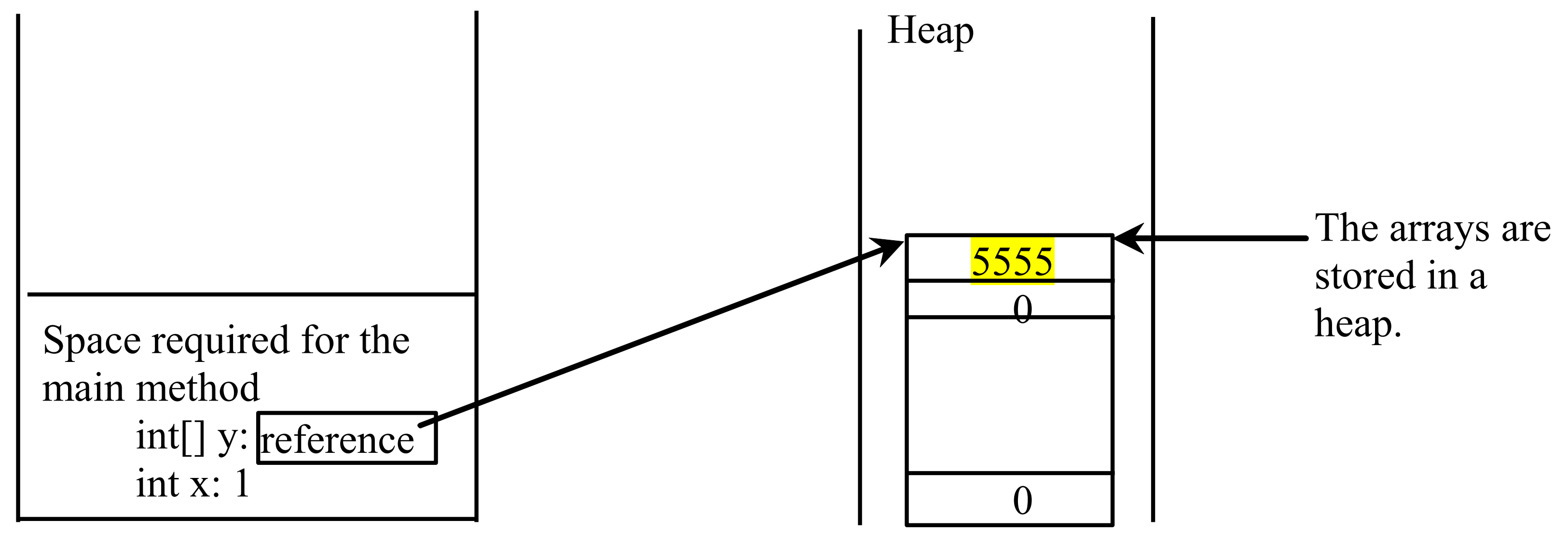
- For a parameter of a ***primitive type value***, the actual value is passed. Changing the value of the local parameter inside the method does not affect the value of the variable outside the method.

- For a parameter of ***an array type***, the value of the parameter contains a reference to an array; this reference is passed to the method. Any changes to the array that occur inside the method body will affect the original array that was passed as the argument.

Simple Example

```
public class Test {
  public static void main(String[] args) {
    int x = 1; // x represents an int value
    int[] y = new int[10]; // y represents an array of int values

    m(x, y); // Invoke m with arguments x and y

    System.out.println("x is " + x);
    System.out.println("y[0] is " + y[0]);
  }

  public static void m(int number, int[] numbers) {
    number = 1001; // Assign a new value to number
    numbers[0] = 5555; // Assign a new value to numbers[0]
  }
}
```

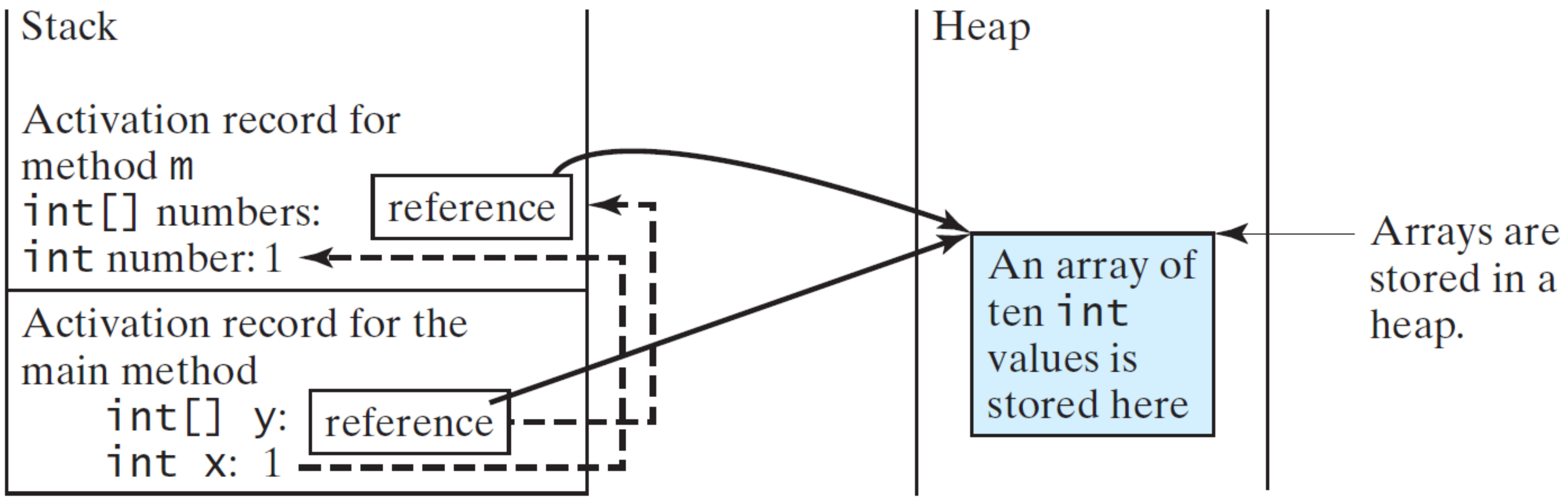
Heap



Heap
5555
0
0
Space required for the main method
int[] y: reference
int x: 1
The arrays are stored in a heap.

The JVM stores the array in an area of memory, called *heap*, which is used for dynamic memory allocation where blocks of memory are allocated and freed in an arbitrary order.

Call Stack



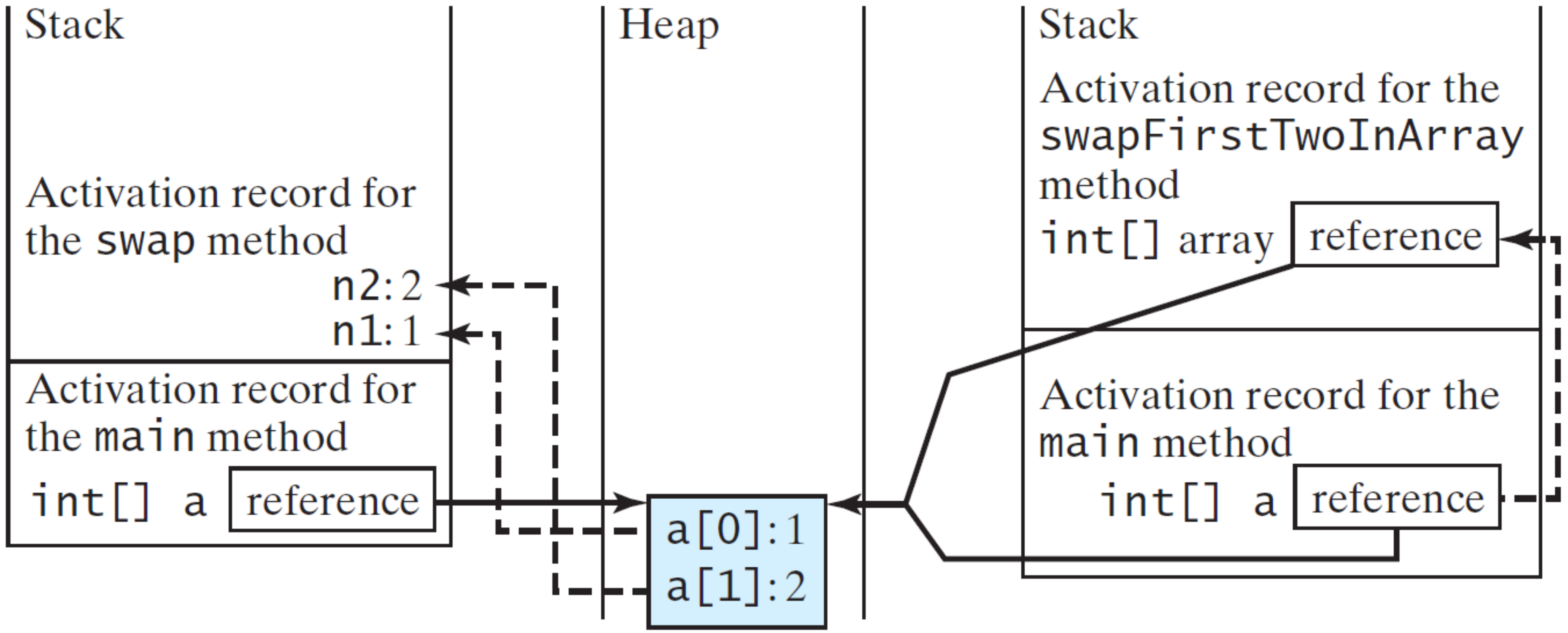
Stack
Activation record for method m
int[] numbers:
reference
int number: 1
Activation record for the main method
int[] y:
reference
int x: 1
Heap
An array of ten int values is stored here
Arrays are stored in a heap.

When invoking m(x, y), the values of x and y are passed to number and numbers. Since y contains the reference value to the array, numbers now contains the same reference value to the same array.

Passing Arrays as Arguments

- Objective: Demonstrate differences of passing primitive data type variables and array variables.
- Try to understand how it works.
- Why those two calls are different?

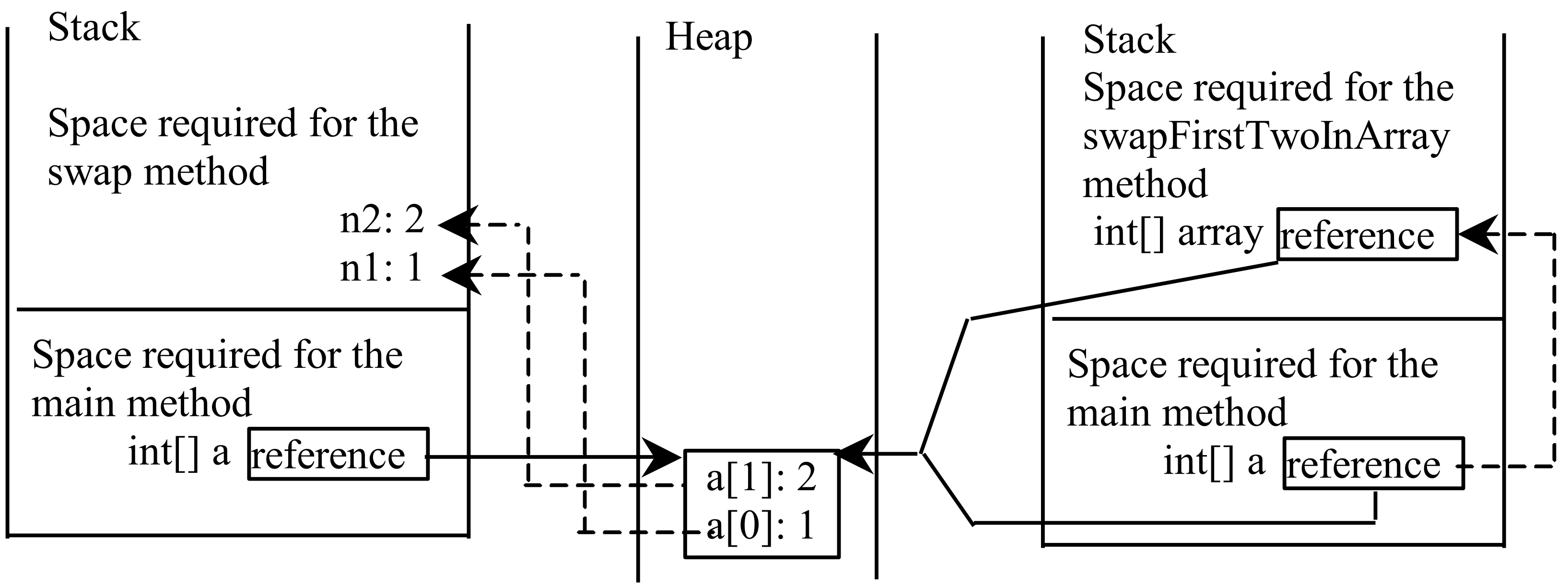
Call Stack


Stack
Activation record for the swap method
n2: 2
n1: 1
Activation record for the main method
int[] a
reference
Heap
a[0]: 1
a[1]: 2
Stack
Activation record for the swapFirstTwoInArray method
int[] array
reference
Activation record for the main method
int[] a
reference

When invoking m(x, y), the values of x and y are passed to number and numbers.

Since y contains the reference value to the array, numbers now contains the same reference value to the same array.

Ilustration: TestPassArray.java


Stack
Space required for the swap method
n2: 2
n1: 1
Space required for the main method
int[] a
reference
Heap
a[1]: 2
a[0]: 1
Stack
Space required for the swapFirstTwoInArray method
int[] array
reference
Space required for the main method
int[] a
reference

Invoke swap(int n1, int n2). The primitive type values in a[0] and a[1] are passed to the swap method.

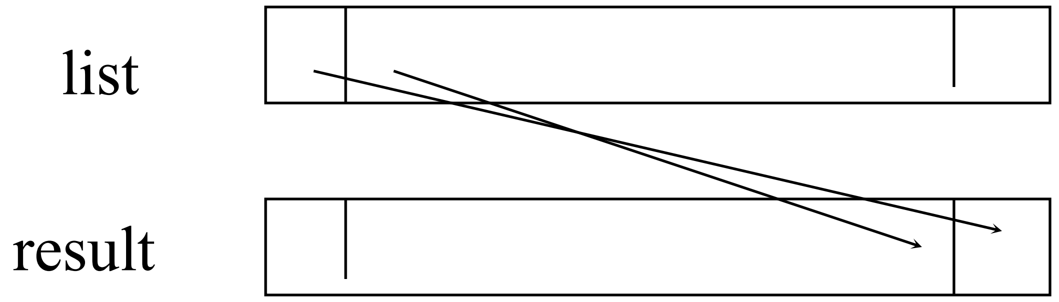
The arrays are stored in a heap.

Invoke swapFirstTwoInArray(int[] array). The reference value in a is passed to the swapFirstTwoInArray method.

Returning an Array from a Method

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

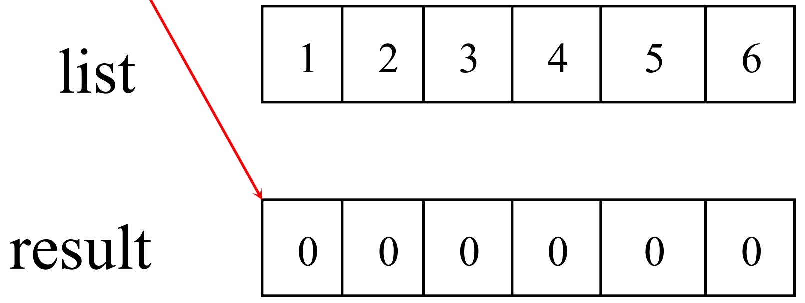


Trace the reverse Method

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

Declare result and create array

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

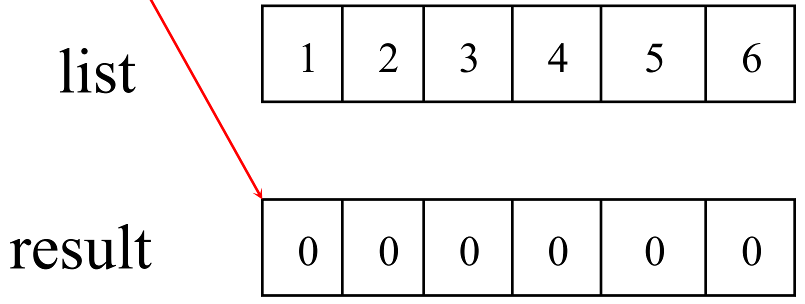


Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

i = 0 and j = 5

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

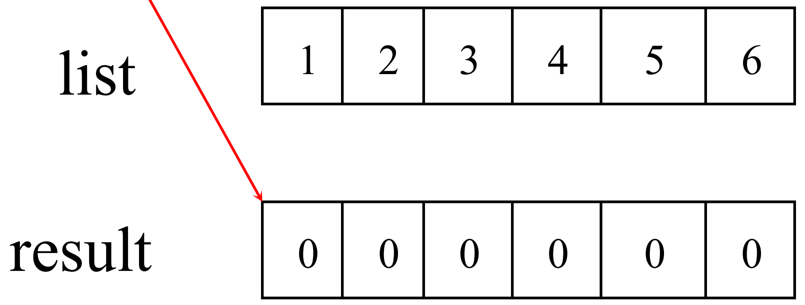


Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

i (= 0) is less than 6

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.lengt
  
  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }
  
  return result;
}
```

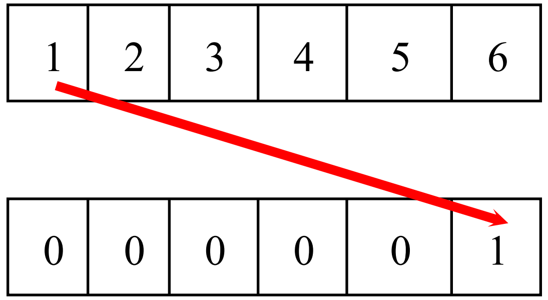


Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i = 0 and j = 5
Assign list[0] to result[5]



Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length   1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

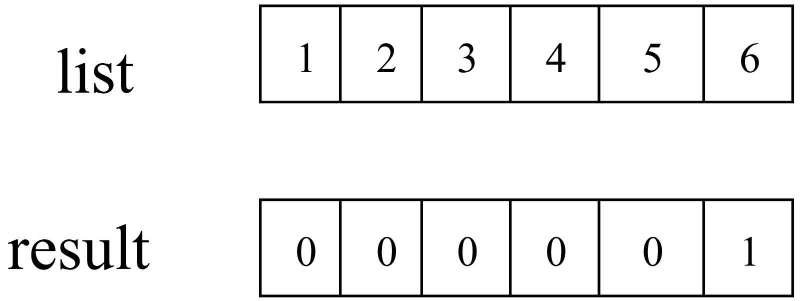
After this, i becomes 1 and j becomes 4

list	1	2	3	4	5	6
result	0	0	0	0	0	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);


  public static int[] reverse(int[] list) {
    int[] result = new int[list.length];

    for (int i = 0, j = result.length - 1;
         i < list.length; i++, j--) {
      result[j] = list[i];
    }

    return result;
  }
```

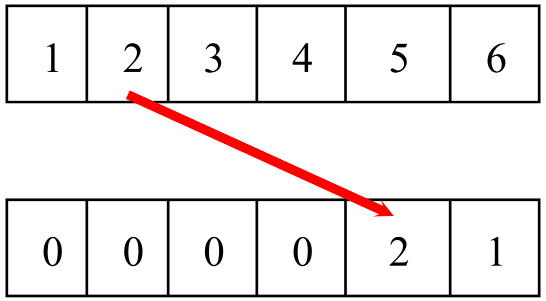
i (=1) is less than 6



Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);


  public static int[] reverse(int[] list) {
    int[] result = new int[list.length];

    for (int i = 0, j = result.length - 1;
         i < list.length; i++, j--) {
      result[j] = list[i];
    }

    return result;
  }
```

i = 1 and j = 4
Assign list[1] to result[4]



Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);


  public static int[] reverse(int[] list) {
    int[] result = new int[list.length];

    for (int i = 0, j = result.length   1;
         i < list.length; i++, j--) {
      result[j] = list[i];
    }

    return result;
  }
```

After this, i becomes 2 and j becomes 3

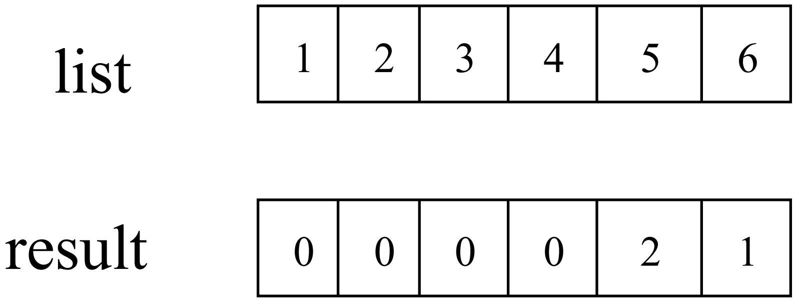
list	1	2	3	4	5	6
result	0	0	0	0	2	1

FACULTY OF COMPUTER SCIENCE

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);


  public static int[] reverse(int[] list) {
    int[] result = new int[list.length];

    for (int i = 0, j = result.length - 1;
          i < list.length; i++, j--) {
      result[j] = list[i];
    }

    return result;
  }
```

i (=2) is still less than 6

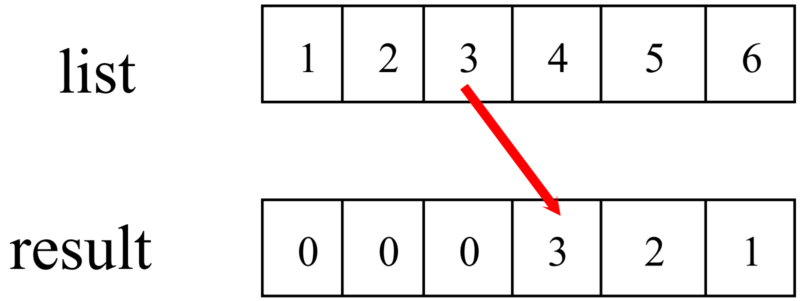


Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i = 2 and j = 3
Assign list[i] to result[j]



Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);


  public static int[] reverse(int[] list) {
    int[] result = new int[list.length];

    for (int i = 0, j = result.length - 1;
         i < list.length; i++, j--) {
      result[j] = list[i];
    }

    return result;
  }
```

After this, i becomes 3 and j becomes 2

list	1	2	3	4	5	6
result	0	0	0	3	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

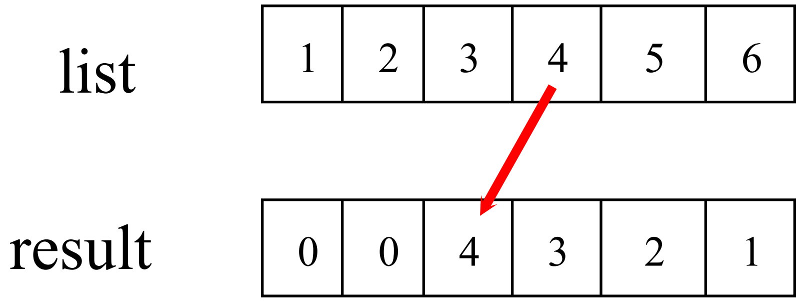
i (=3) is still less than 6

list	1	2	3	4	5	6
result	0	0	0	3	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);


  public static int[] reverse(int[] list) {
    int[] result = new int[list.length];

    for (int i = 0, j = result.length - 1;
         i < list.length; i++, j--) {
      result[j] = list[i];
    }

    return result;
  }
```

i = 3 and j = 2
Assign list[i] to result[j]



Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);


  public static int[] reverse(int[] list) {
    int[] result = new int[list.length];

    for (int i = 0, j = result.length - 1;
         i < list.length; i++, j--) {
      result[j] = list[i];
    }

    return result;
  }
```

After this, i becomes 4 and j becomes 1

list	1	2	3	4	5	6
result	0	0	4	3	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

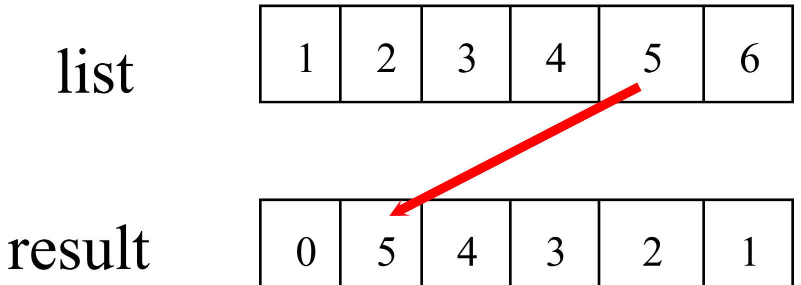
i (=4) is still less than 6

list	1	2	3	4	5	6
result	0	0	4	3	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);


  public static int[] reverse(int[] list) {
    int[] result = new int[list.length];

    for (int i = 0, j = result.length - 1;
         i < list.length; i++, j--) {
      result[j] = list[i];
    }

    return result;
  }
```

i = 4 and j = 1
Assign list[i] to result[j]



Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

After this, i becomes 5 and j becomes 0

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length   1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

list	1	2	3	4	5	6
result	0	5	4	3	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);


  public static int[] reverse(int[] list) {
    int[] result = new int[list.length]

    for (int i = 0, j = result.length - 1;
         i < list.length; i++, j--) {
      result[j] = list[i];
    }

    return result;
  }
```

i (=5) is still less than 6

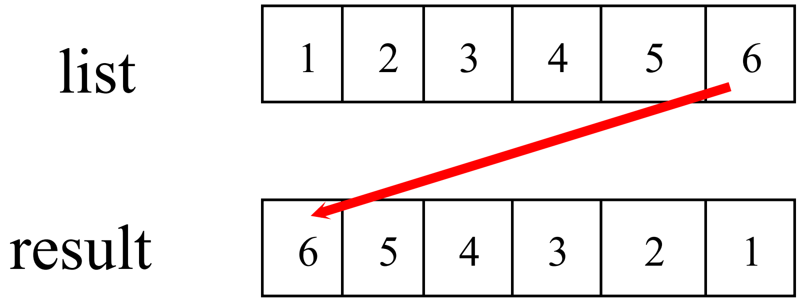
list	1	2	3	4	5	6
result	0	5	4	3	2	1

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

i = 5 and j = 0
Assign list[i] to result[j]



Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);


  public static int[] reverse(int[] list) {
    int[] result = new int[list.length];

    for (int i = 0, j = result.length - 1;
         i < list.length; i++, j--) {
      result[j] = list[i];
    }

    return result;
  }
```

After this, i becomes 6 and j becomes -1

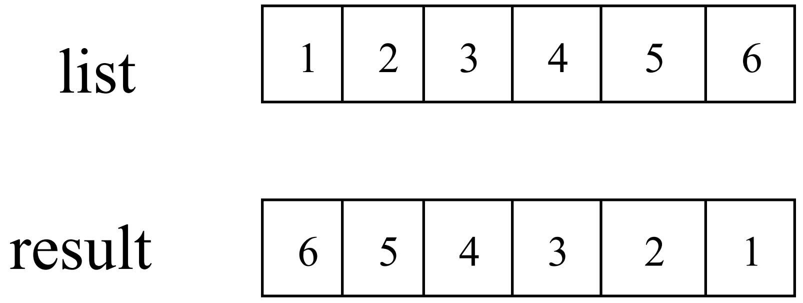
list	1	2	3	4	5	6
result	6	5	4	3	2	1

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FACULTY OF COMPUTER SCIENCE

Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);


  public static int[] reverse(int[] list) {
    int[] result = new int[list.length]

    for (int i = 0, j = result.length - 1;
         i < list.length; i++, j--) {
      result[j] = list[i];
    }

    return result;
  }
```

i (=6) < 6 is false. So exit the loop.

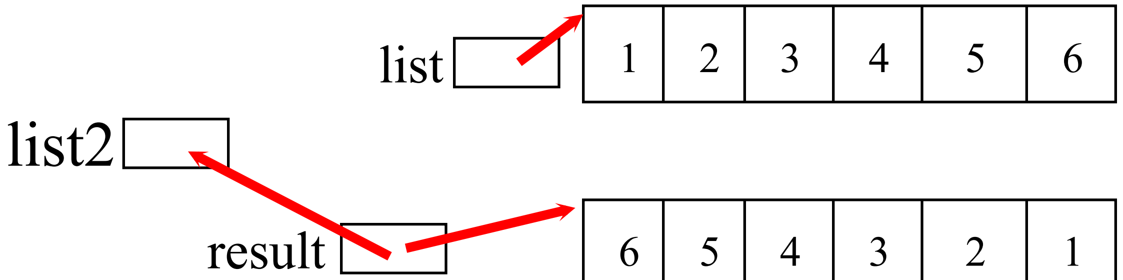


Trace the reverse Method, cont.

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

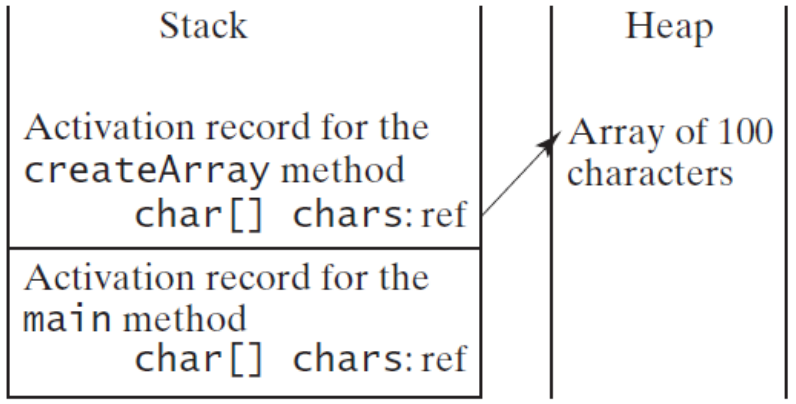
```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1;
       i < list.length; i++, j--) {
    result[j] = list[i];
  }

  return result;
}
```

Return result



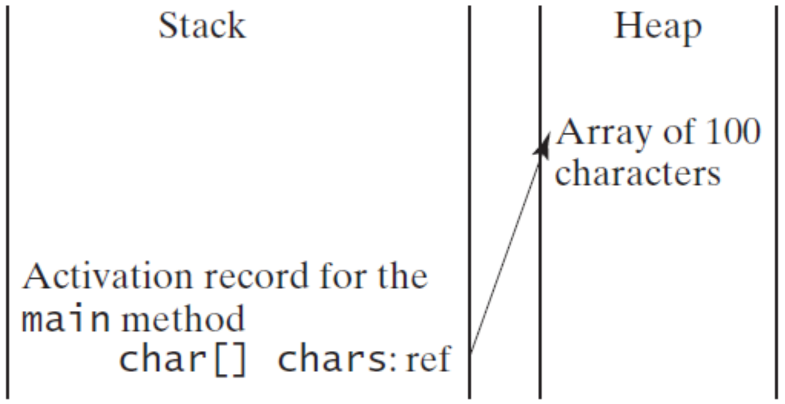
Problem: Counting Occurrence of Each Letter

- Generate 100 lowercase letters randomly and assign to an array of characters.
- Count the occurrence of each letter in the array.



Stack
Heap
Activation record for the createArray method
char[] chars: ref
Activation record for the main method
char[] chars: ref
Array of 100 characters

(a) Executing createArray in line 5



(b) After exiting createArray in line 5

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER

Ada Pertanyaan?

- ❑ copying array,
- ❑ Passing arrays
- ❑ Pass by Value, Pass array references
- ❑ Latihan dan Algoritma terkait array

FAKULTAS
ILMU
KOMPUTER

Dasar Dasar Pemrograman 2

Acuan: Introduction to Java Programming and Data Structure, Bab 7 dan Bab 8

Sumber Slide: Liang,

Dimodifikasi untuk Fasilkom UI oleh Ade Azurat

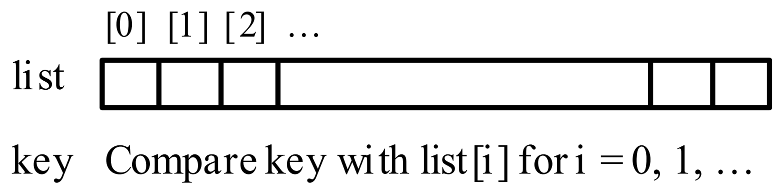
Topik Pekan 3b: Algoritma pada Single Array

Searching Arrays

Searching is the process of looking for a specific element in an array; for example, discovering whether a certain score is included in a list of scores.

Searching is a common task in computer programming. There are many algorithms and data structures devoted to searching. In this section, two commonly used approaches are discussed, *linear search* and *binary search*.

```
public class LinearSearch {
  /** The method for finding a key in the list */
  public static int linearSearch(int[] list, int key) {
    for (int i = 0; i < list.length; i++)
      if (key == list[i])
        return i;
    return -1;
  }
}
```



Linear Search

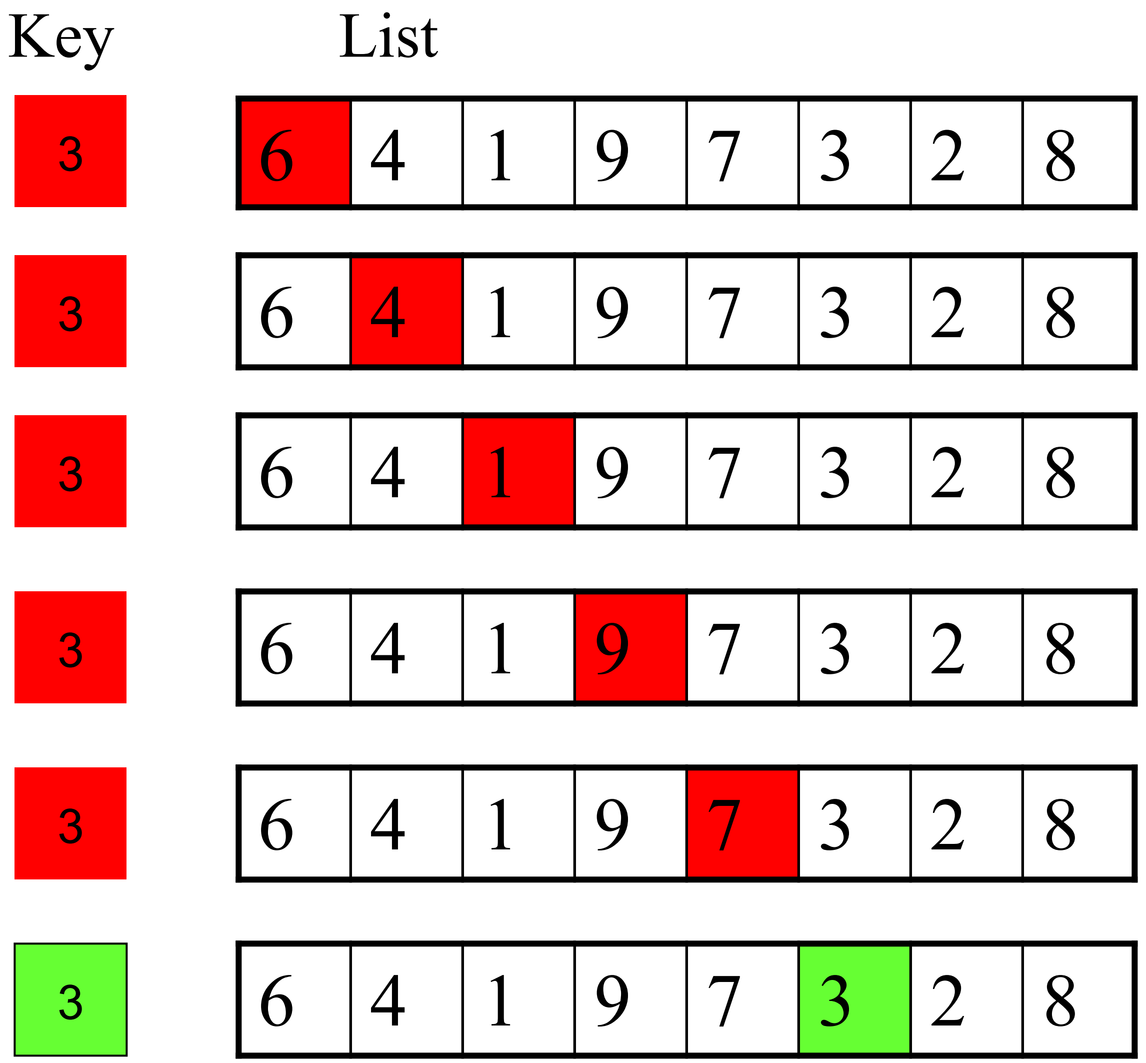
The linear search approach compares the key element, key, *sequentially* with each element in the array list.

The method continues to do so until the key matches an element in the list or the list is exhausted without a match being found.

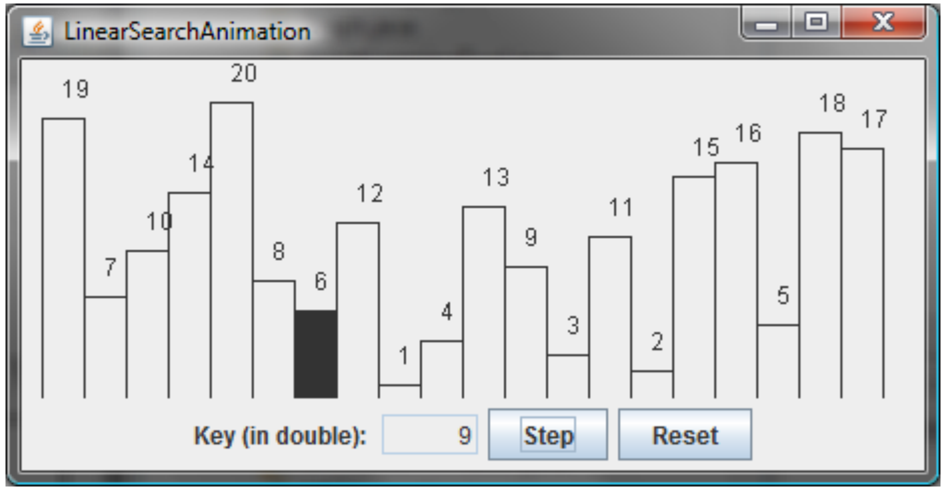
If a match is made, the linear search returns the index of the element in the array that matches the key.

If no match is found, the search returns -1.

Linear Search Animation



Linear Search Animation



From Idea to Solution

```
/** The method for finding a key in the list */
public static int linearSearch(int[] list, int key) {
  for (int i = 0; i < list.length; i++)
    if (key == list[i])
      return i;
  return -1;
}
```

Trace the method

```
int[] list = {1, 4, 4, 2, 5, -3, 6, 2};
int i = linearSearch(list, 4);  // returns 1
int j = linearSearch(list, -4); // returns -1
int k = linearSearch(list, -3); // returns 5
```

Binary Search

For binary search to work, the elements in the array must already be ordered.

Without loss of generality, assume that the array is in ascending order.

e.g., 2 4 7 10 11 45 50 59 60 66 69 70 79

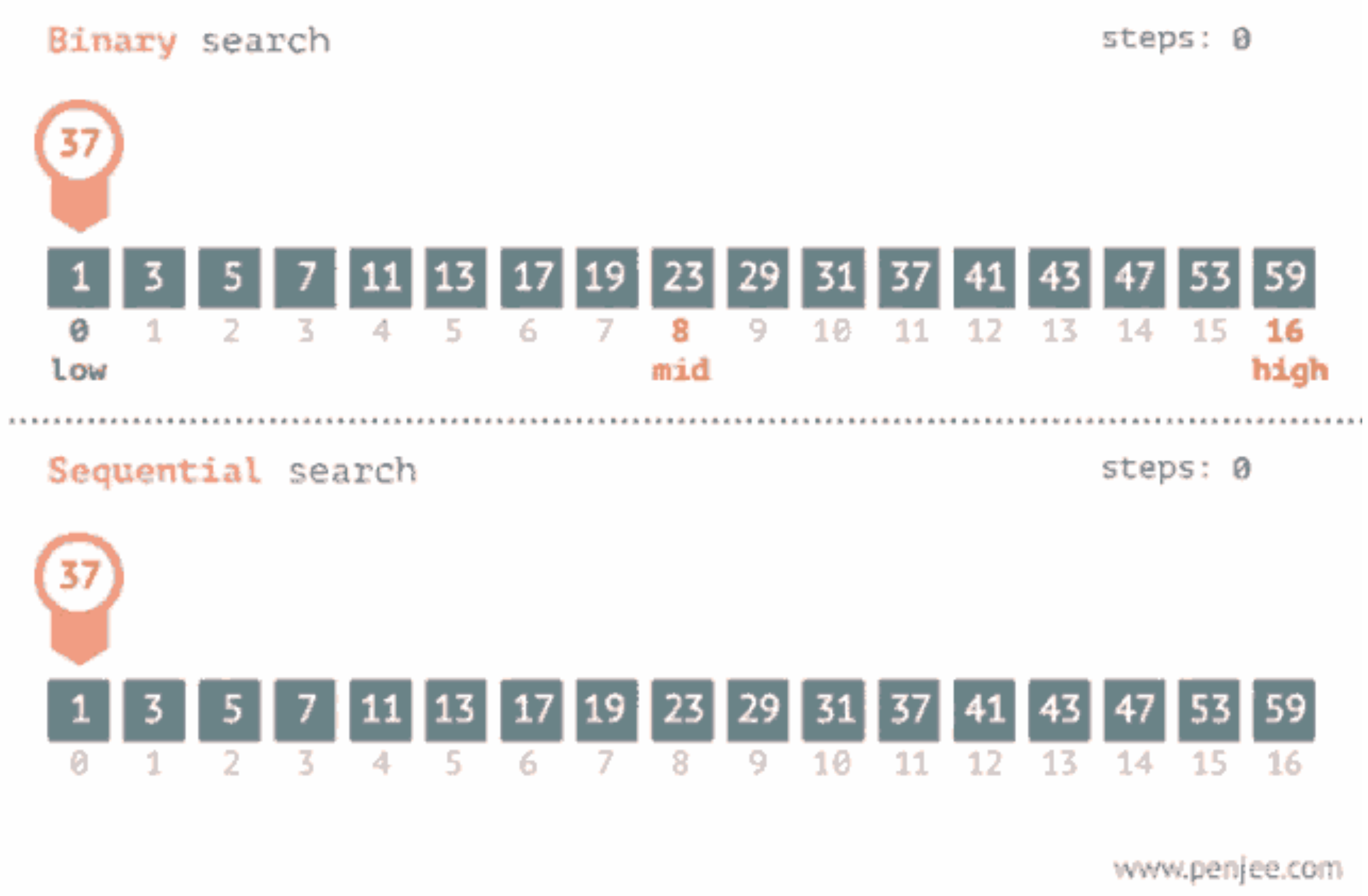
The binary search first compares the key with the element in the middle of the array.

Binary Search, cont.

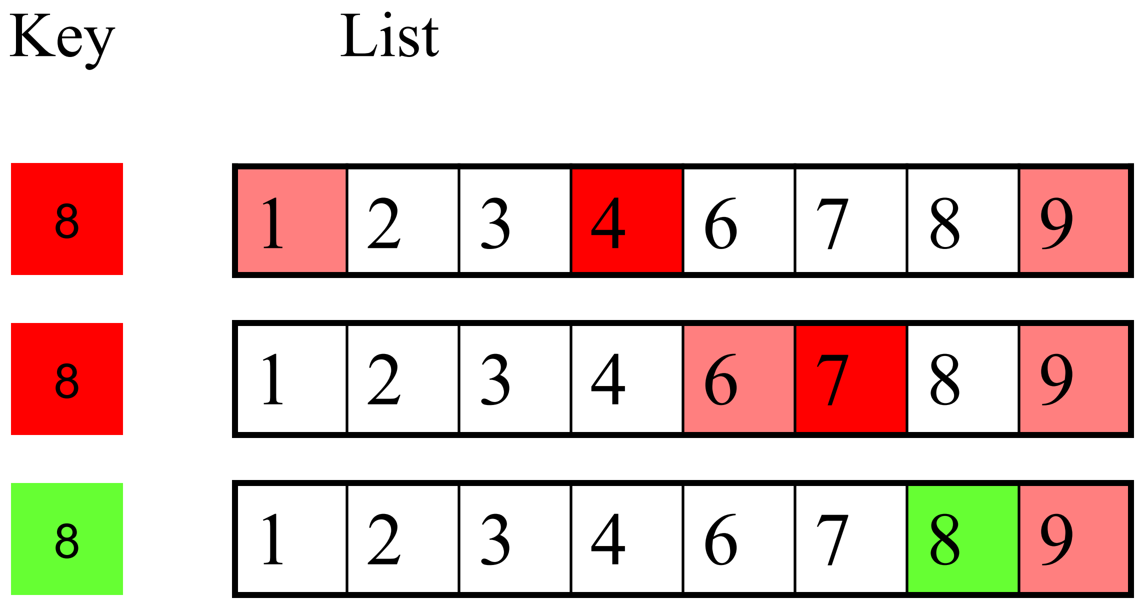
Consider the following three cases:

- If the key is less than the middle element, you only need to search the key in the first half of the array.
- If the key is equal to the middle element, the search ends with a match.
- If the key is greater than the middle element, you only need to search the key in the second half of the array.

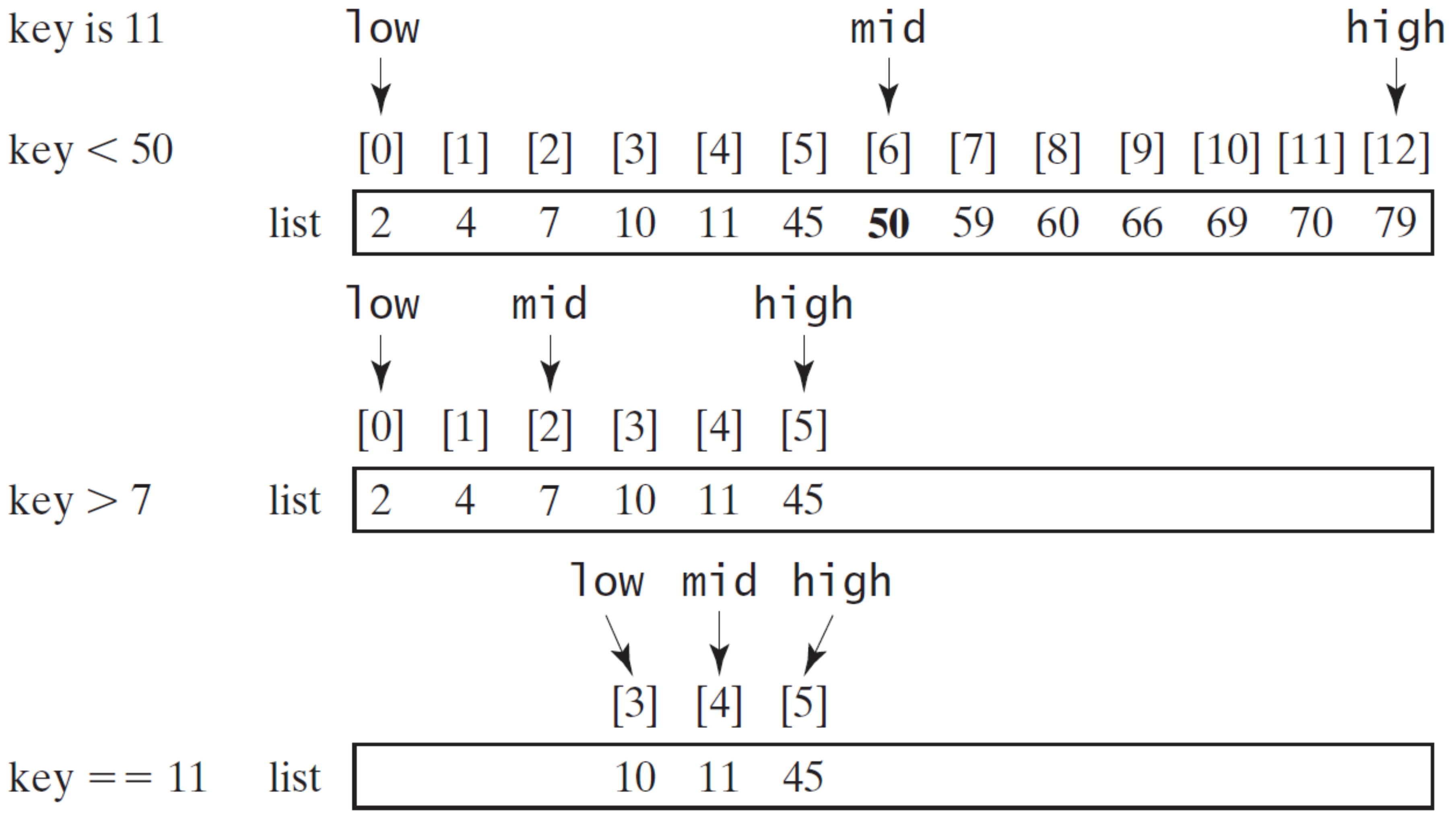
Binary Search Animation



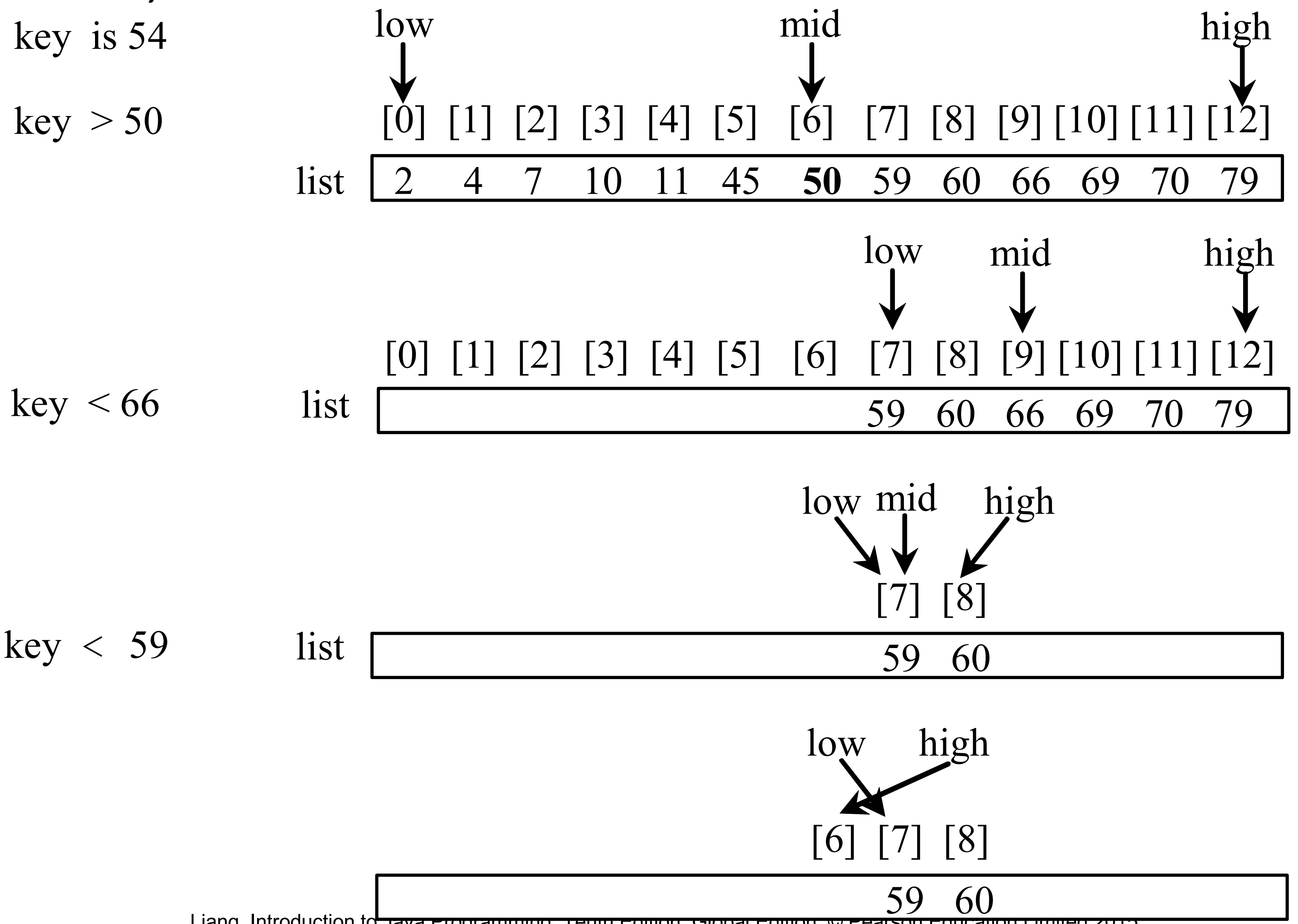
Binary Search



Binary Search, cont.



Binary Search, cont.



Binary Search, cont.

The binarySearch method returns the index of the element in the list that matches the search key if it is contained in the list. Otherwise, it returns

insertion point = - 1.

The insertion point is the point at which the key would be inserted into the list.

Masih belum paham,
mengapa perlu return demikian?
Baca Section 7.10

From Idea to Soluton

```
/** Use binary search to find the key in the list */
public static int binarySearch(int[] list, int key) {
  int low = 0;
  int high = list.length - 1;

  while (high >= low) {
    int mid = (low + high) / 2;
    if (key < list[mid])
      high = mid - 1;
    else if (key == list[mid])
      return mid;
    else
      low = mid + 1;
  }

  return -1 - low;
}
```

FACULTY OF COMPUTER SCIENCE
UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

The Arrays.binarySearch Method

Since binary search is frequently used in programming, Java provides several overloaded binarySearch methods for searching a key in an array of int, double, char, short, long, and float in the java.util.Arrays class. For example, the following code searches the keys in an array of numbers and an array of characters.

```
int[] list = {2, 4, 7, 10, 11, 45, 50, 59, 60, 66, 69, 70, 79};
System.out.println("Index is " +
  java.util.Arrays.binarySearch(list, 11));
```

Return is 4

```
char[] chars = {'a', 'c', 'g', 'x', 'y', 'z'};
System.out.println("Index is " +
  java.util.Arrays.binarySearch(chars, 't'));
```

Return is –4 (insertion point is 3, so return is -3-1)

For the binarySearch method to work, the array must be pre-sorted in increasing order.

Sorting Arrays

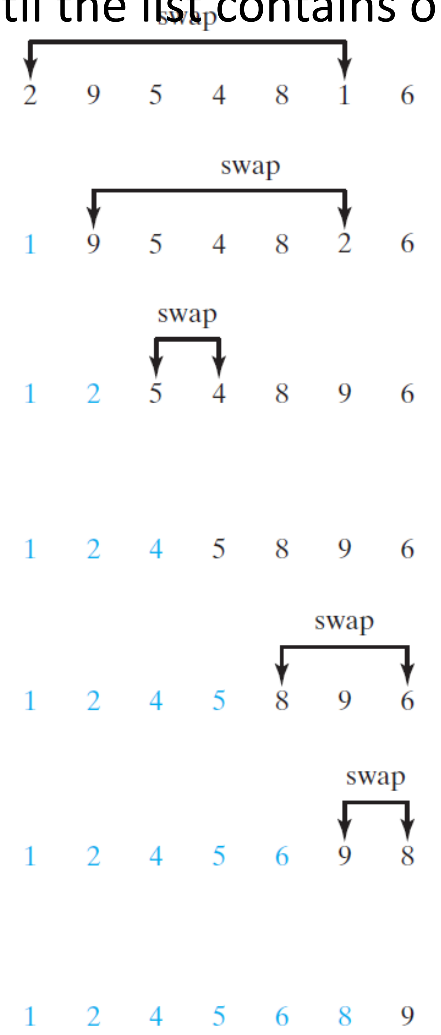
Sorting, like searching, is also a common task in computer programming.

Many different algorithms have been developed for sorting.

This section introduces a simple, intuitive sorting algorithms: *selection sort*.

Selection Sort

Selection sort finds the smallest number in the list and places it first. It then finds the smallest number remaining and places it second, and so on until the list contains only a single number.



Select 1 (the smallest) and swap it with 2 (the first) in the list.

The number 1 is now in the correct position and thus no longer needs to be considered.

Select 2 (the smallest) and swap it with 9 (the first) in the remaining list.

The number 2 is now in the correct position and thus no longer needs to be considered.

Select 4 (the smallest) and swap it with 5 (the first) in the remaining list.

The number 4 is now in the correct position and thus no longer needs to be considered.

5 is the smallest and in the right position. No swap is necessary.

The number 5 is now in the correct position and thus no longer needs to be considered.

Select 6 (the smallest) and swap it with 8 (the first) in the remaining list.

The number 6 is now in the correct position and thus no longer needs to be considered.

Select 8 (the smallest) and swap it with 9 (the first) in the remaining list.

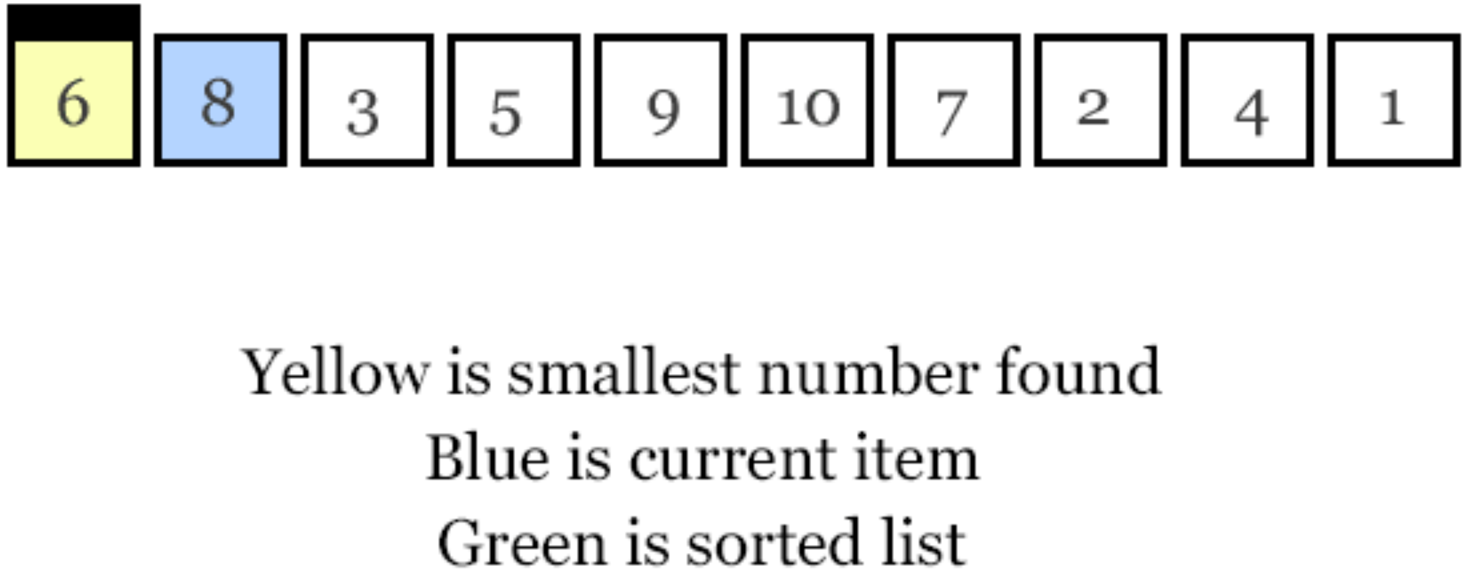
The number 8 is now in the correct position and thus no longer needs to be considered.

Since there is only one element remaining in the list, the sort is completed.

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FACULTY OF COMPUTER SCIENCE

Selection Sort Animation



Selection Sort in Action, thanks to Xybernetics for the gif

From Idea to Solution

```
for (int i = 0; i < list.length; i++) {
  select the smallest element in list[i..listSize-1];
  swap the smallest with list[i], if necessary;
  // list[i] is in its correct position.
  // The next iteration apply on list[i..listSize-1]
}
```

```
list[0] list[1] list[2] list[3] ...              list[10]

list[0] list[1] list[2] list[3] ...              list[10]

list[0] list[1] list[2] list[3] ...              list[10]

list[0] list[1] list[2] list[3] ...              list[10]

list[0] list[1] list[2] list[3] ...              list[10]

                                ...

list[0] list[1] list[2] list[3] ...              list[10]
```

FACULTY OF COMPUTER SCIENCE

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

```
for (int i = 0; i < listSize; i++) {
  select the smallest element in list[i..listSize-1];
  swap the smallest with list[i], if necessary;
  // list[i] is in its correct position.
  // The next iteration apply on list[i..listSize-1]
}
```

```
for (int i = 0; i < listSize; i++) {
  select the smallest element in list[i..listSize-1];
  swap the smallest with list[i], if necessary;
  // list[i] is in its correct position.
  // The next iteration apply on list[i..listSize-1]
}
```

Expand

```
double currentMin = list[i];
int currentMinIndex = i;
for (int j = i; j < list.length; j++) {
  if (currentMin > list[j]) {
    currentMin = list[j];
    currentMinIndex = j;
  }
}
```

```
for (int i = 0; i < listSize; i++) {
  select the smallest element in list[i..listSize-1];
  swap the smallest with list[i], if necessary;
  // list[i] is in its correct position.
  // The next iteration apply on list[i..listSize-1]
}
```

Expand

```
    if (currentMinIndex != i) {
      list[currentMinIndex] = list[i];
      list[i] = currentMin;
    }
```

UNIVERSITAS INDONESIA Veritas, Probitas, Iustitia
FACULTY OF COMPUTER SCIENCE

Wrap it in a Method

```
/** The method for sorting the numbers */
public static void selectionSort(double[] list) {
  for (int i = 0; i < list.length; i++) {
    // Find the minimum in the list[i..list.length-1]
    double currentMin = list[i];
    int currentMinIndex = i;
    for (int j = i + 1; j < list.length; j++) {
      if (currentMin > list[j]) {
        currentMin = list[j];
        currentMinIndex = j;
      }
    }

    // Swap list[i] with list[currentMinIndex] if necessary;
    if (currentMinIndex != i) {
      list[currentMinIndex] = list[i];
      list[i] = currentMin;
    }
  }
}
```

Invoke it

selectionSort(yourList)

The Arrays.sort Method

Since sorting is frequently used in programming, Java provides several overloaded sort methods for sorting an array of int, double, char, short, long, and float in the java.util.Arrays class. For example, the following code sorts an array of numbers and an array of characters.

```
double[] numbers = {6.0, 4.4, 1.9, 2.9, 3.4, 3.5};
java.util.Arrays.sort(numbers);

char[] chars = {'a', 'A', '4', 'F', 'D', 'P'};
java.util.Arrays.sort(chars);
```

Java 8 provides Arrays.parallelSort(list) that utilizes the multicore for fast sorting.

The Arrays.toString(list) Method

The Arrays.toString(list) method can be used to return a string representation for the list.

Dasar Dasar Pemrograman 2

Acuan: Introduction to Java Programming and Data Structure, Bab 7 dan Bab 8

Sumber Slide: Liang,

Dimodifikasi untuk Fasilkom UI oleh Ade Azurat

Topik: Pass Arguments to Invoke the Main Method

Main Method Is Just a Regular Method

You can call a regular method by passing actual parameters. Can you pass arguments to main?

Of course, yes.

For example, the main method in class B is invoked by a method in A, as shown below:

```
public class A {
  public static void main(String[] args) {
    String[] strings = {"New York",
      "Boston", "Atlanta"};
    B.main(strings);
  }
}
```

```
class B {
  public static void main(String[] args) {
    for (int i = 0; i < args.length; i++)
      System.out.println(args[i]);
  }
}
```

FACULTY OF COMPUTER SCIENCE

Command-Line Parameters

```
class TestMain {
  public static void main(String[] args) {
   ...
   }
}

java TestMain arg0 arg1 arg2 ... argn
```

Processing Command-Line Parameters

In the main method, get the arguments from `args[0], args[1], ..., args[n],` which corresponds to `arg0, arg1, ..., argn` in the command line.

Problem: Calculator

- Objective:
  Write a program that will perform binary operations on integers.
  The program receives three parameters: an operator and two integers.

```
java Calculator 2 + 3
java Calculator 2 - 3
java Calculator 2 / 3
java Calculator 2 . 3
```

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER

Ada Pertanyaan?

- ❑ Linear Search
- ❑ Binary Search
- ❑ Sorting Array
- ❑ Command Line Parameter

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS
ILMU
KOMPUTER

Dasar Dasar Pemrograman 2

Acuan: Introduction to Java Programming and Data Structure, Bab 7 dan Bab 8

Sumber Slide: Liang,

Dimodifikasi untuk Fasilkom UI oleh Ade Azurat

Topik Pekan 3d: MultiDimensional Array

Motivations

You can use a two-dimensional array to represent a matrix or a table.

For example, the following table that describes the distances between the cities can be represented using a two-dimensional array.

Distance Table (in miles)

	Chicago	Boston	New York	Atlanta	Miami	Dallas	Houston
Chicago	0	983	787	714	1375	967	1087
Boston	983	0	214	1102	1763	1723	1842
New York	787	214	0	888	1549	1548	1627
Atlanta	714	1102	888	0	661	781	810
Miami	1375	1763	1549	661	0	1426	1187
Dallas	967	1723	1548	781	1426	0	239
Houston	1087	1842	1627	810	1187	239	0

Motivations

You can use a two-dimensional array to represent a matrix or a table.

The following matrix escribes the distances between the cities.

It is represented using a two-dimensional array.

```
double[][] distances = {
  {0, 983, 787, 714, 1375, 967, 1087},
  {983, 0, 214, 1102, 1763, 1723, 1842},
  {787, 214, 0, 888, 1549, 1548, 1627},
  {714, 1102, 888, 0, 661, 781, 810},
  {1375, 1763, 1549, 661, 0, 1426, 1187},
  {967, 1723, 1548, 781, 1426, 0, 239},
  {1087, 1842, 1627, 810, 1187, 239, 0},
};
```

Declare/Create Two-dimensional Arrays

```
// Declare array ref var
dataType[][] refVar;

// Create array and assign its reference to variable
refVar = new dataType[10][10];

// Combine declaration and creation in one statement
dataType[][] refVar = new dataType[10][10];

// Alternative syntax
dataType refVar[][] = new dataType[10][10];
```

Declaring Variables of Two-dimensional Arrays and Creating Two-dimensional Arrays

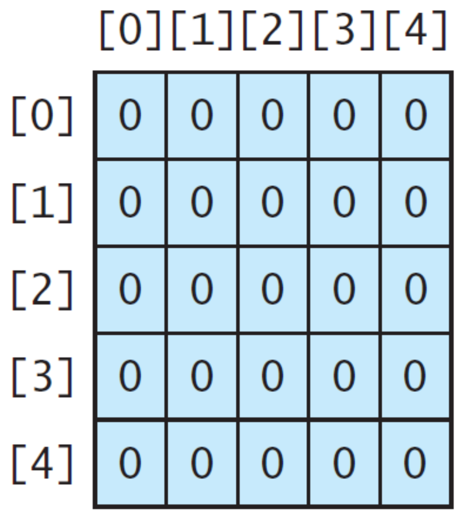
```
int matrix[][] = new int[10][10];
```

or

```
int[][] matrix = new int[10][10];
matrix[0][0] = 3;

for (int i = 0; i < matrix.length; i++)
  for (int j = 0; j < matrix[i].length; j++)
    matrix[i][j] = (int)(Math.random() * 1000);
```

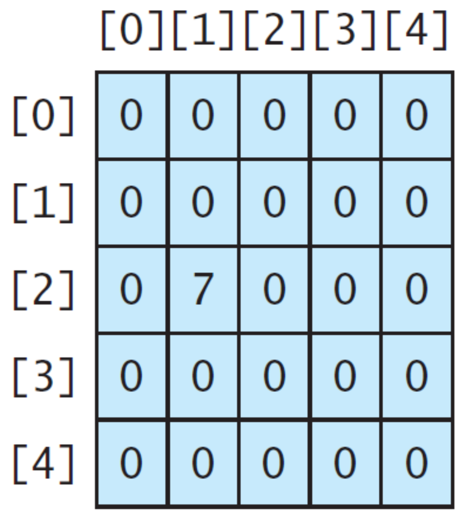
Style seperti ini lebih disukai!

Two-dimensional Array Illustration



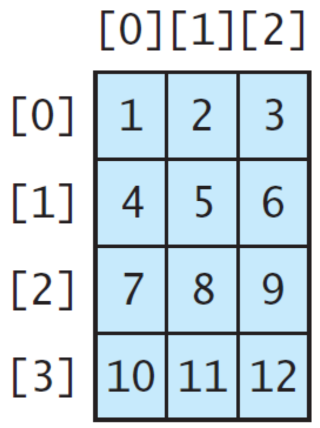
```
matrix = new int[5][5];
```

(a)



```
matrix[2][1] = 7;
```

(b)



```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

(c)

matrix.length? 5

matrix[0].length? 5

array.length? 4

array[0].length? 3

Declaring, Creating, and Initializing Using Shorthand Notations

You can also use an array initializer to declare, create and initialize a two-dimensional array. For example,

```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

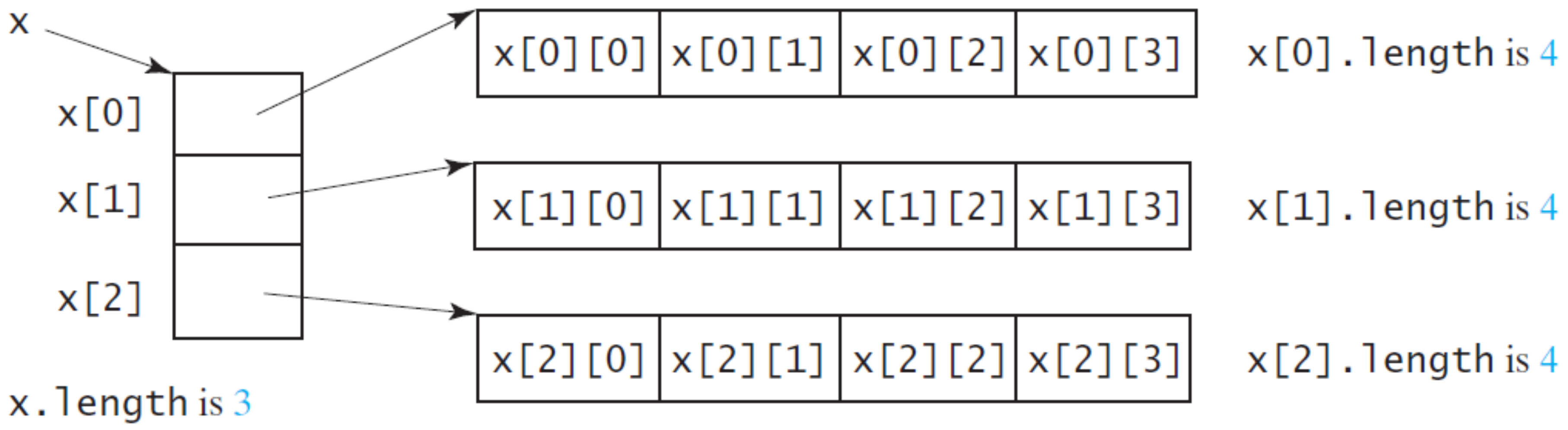
Same as

```
int[][] array = new int[4][3];
array[0][0] = 1; array[0][1] = 2; array[0][2] = 3;
array[1][0] = 4; array[1][1] = 5; array[1][2] = 6;
array[2][0] = 7; array[2][1] = 8; array[2][2] = 9;
array[3][0] = 10; array[3][1] = 11; array[3][2] = 12;
```

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FACULTY OF COMPUTER SCIENCE

Lengths of Two-dimensional Arrays

```
int[][] x = new int[3][4];
```



Lengths of Two-dimensional Arrays, cont.

```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

array.length
array[0].length
array[1].length
array[2].length
array[3].length

array[4].length ArrayIndexOutOfBoundsException

Ragged Arrays

Each row in a two-dimensional array is itself an array.

So, the rows can have different lengths.

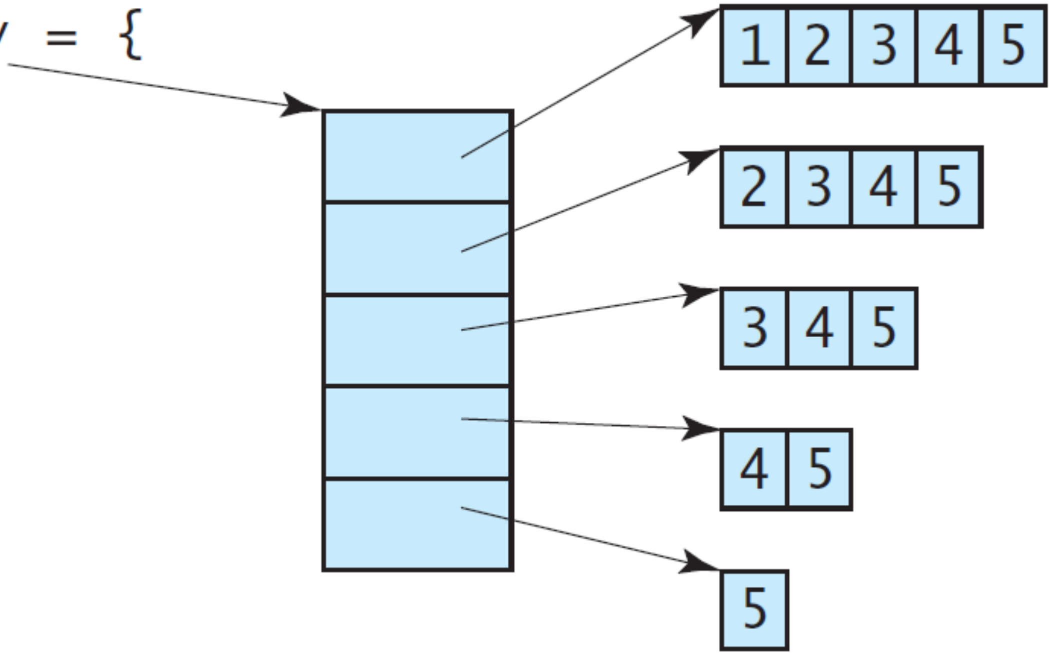
Such an array is known as *a ragged array*. For example:

```
int[][] matrix = {
  {1, 2, 3, 4, 5},
  {2, 3, 4, 5},
  {3, 4, 5},
  {4, 5},
  {5}
};
```

```
matrix.length is 5
matrix[0].length is 5
matrix[1].length is 4
matrix[2].length is 3
matrix[3].length is 2
matrix[4].length is 1
```

Ragged Arrays, cont.

```
int[][] triangleArray = {
  {1, 2, 3, 4, 5},
  {2, 3, 4, 5},
  {3, 4, 5},
  {4, 5},
  {5}
};
```



UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS
ILMU
KOMPUTER

Ada Pertanyaan?

- ❑ Mengapa perlu multidimensi array?
- ❑ Membuat dan menginisiasi multidimensi array
- ❑ Mengetahui ukuran array dua dimensi
- ❑ *Ragged Array*

Processing Two-Dimensional Arrays

See the examples in the text.

1. (Initializing arrays with input values)
2. (Printing arrays)
3. (Summing all elements)
4. (Summing all elements by column)
5. (Which row has the largest sum)
6. (Finding the smallest index of the largest element)
7. (*Random shuffling*)

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FACULTY OF
COMPUTER
SCIENCE

Initializing arrays with input values

```
java.util.Scanner input = new Scanner(System.in);
System.out.println("Enter " + matrix.length + " rows and " +
  matrix[0].length + " columns: ");
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length; column++) {
    matrix[row][column] = input.nextInt();
  }
}
```

FACULTY OF COMPUTER SCIENCE

Initializing arrays with random values

```
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length; column++) {
    matrix[row][column] = (int)(Math.random() * 100);
  }
}
```

Printing arrays

```
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length; column++) {
    System.out.print(matrix[row][column] + " ");
  }
  System.out.println();
}
```

Summing all elements

```
int total = 0;
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length; column++){
    total += matrix[row][column];
  }
}
```

FACULTY OF COMPUTER SCIENCE

Summing elements by column

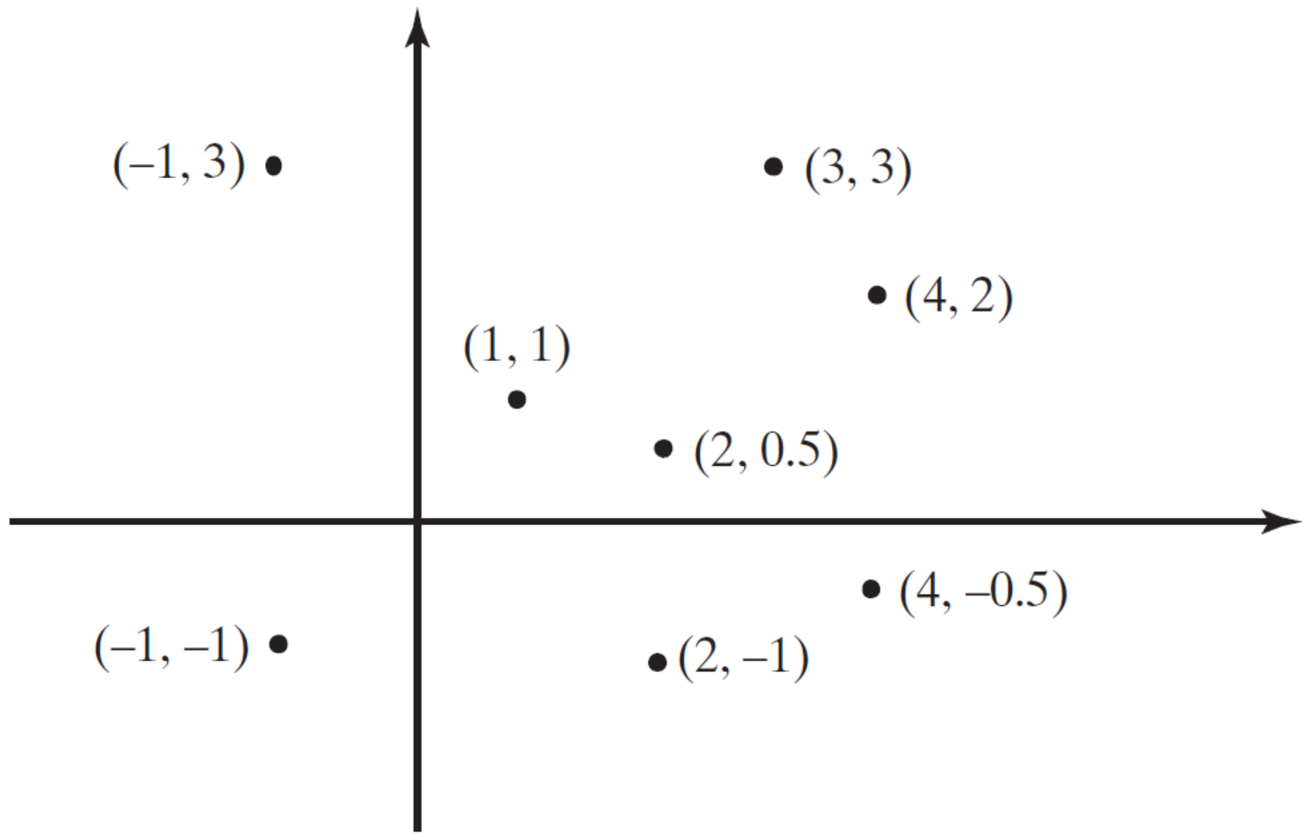
```
for (int column = 0; column < matrix[0].length; column++) {
  int total = 0;
  for (int row = 0; row < matrix.length; row++)
      total += matrix[row][column];
  System.out.println("Sum for column " + column + " is " + total);
}
```

Random shuffling

```
for (int i = 0; i < matrix.length; i++) {
  for (int j = 0; j < matrix[i].length; j++) {
    int i1 = (int)(Math.random() * matrix.length);
    int j1 = (int)(Math.random() * matrix[i].length);
    // Swap matrix[i][j] with matrix[i1][j1]
    int temp = matrix[i][j];
    matrix[i][j] = matrix[i1][j1];
    matrix[i1][j1] = temp;
  }
}
```

Problem: Finding Two Points Nearest to Each Other

https://replit.com/@AdeAzurat/DDP2-Pekan05-Array#FindNearestPoints.java



	x	y
0	–1	3
1	–1	–1
2	1	1
3	2	0.5
4	2	–1
5	3	3
6	4	2
7	4	–0.5

Problem: Grading Multiple-Choice Test

https://replit.com/@AdeAzurat/DDP2-Pekan05-Array#GradeExam.java

Students' answer

	0	1	2	3	4	5	6	7	8	9
Student 0	A	B	A	C	C	D	E	E	A	D
Student 1	D	B	A	B	C	A	E	E	A	D
Student 2	E	D	D	A	C	B	E	E	A	D
Student 3	C	B	A	E	D	C	E	E	A	D
Student 4	A	B	D	C	C	D	E	E	A	D
Student 5	B	B	E	C	C	D	E	E	A	D
Student 6	B	B	A	C	C	D	E	E	A	D
Student 7	E	B	E	C	C	D	E	E	A	D

Objective:
write a program that grades multiple-choice test.

Key to the Questions:

	0	1	2	3	4	5	6	7	8	9
Key	D	B	D	C	C	D	A	E	A	D

Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

Every row contains the numbers 1 to 9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

Every column contains the numbers 1 to 9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

Every 3×3 box contains the numbers 1 to 9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FACULTY OF COMPUTER SCIENCE

Checking Whether a Solution Is Correct

https://replit.com/@AdeAzurat/DDP2-Pekan05-Array#CheckSudokuSolution.java

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6							
			4	1	9			5
				8			7	9

5	3	*4*	*6*	7	*8*	*9*	*1*	*2*
6	*7*	*2*	1	9	5	*3*	*4*	*8*
1	9	8	*3*	*4*	*2*	*5*	6	*7*
8	*5*	*9*	*7*	6	*1*	*4*	*2*	3
4	*2*	*6*	8	*5*	3	*7*	*9*	1
7	*1*	*3*	*9*	2	*4*	*8*	*5*	6
9	6	*1*	*5*	*3*	*7*	*2*	*8*	*4*
2	*8*	*7*	4	1	9	*6*	*3*	5
3	*4*	*5*	*2*	8	*6*	*1*	7	9

Untuk memudahkan pengisian data, bisa gunakan input redirection "<", contoh:

```
> javac CheckSudokuSolution.java
> java  CheckSudokuSolution < SudokuSol1.txt
```

FACULTY OF COMPUTER SCIENCE

UNIVERSITAS INDONESIA

Veritas, Probitas, Iustitia

Multidimensional Arrays

Occasionally, you will need to represent n-dimensional data structures. In Java, you can create n-dimensional arrays for any integer n.

The way to declare two-dimensional array variables and create two-dimensional arrays can be generalized to declare n-dimensional array variables and create n-dimensional arrays for n >= 3.

FACULTY OF
COMPUTER
SCIENCE

Problem: Calculating Total Scores

https://replit.com/@AdeAzurat/DDP2-Pekan05-Array#TotalScore.java

Objective: Write a program that calculates the total score for students in a class.

- Suppose the scores are stored in a three-dimensional array named scores.
- The first index in scores refers to a student, the second refers to an exam, and the third refers to the part of the exam.
- Suppose there are 7 students, 5 exams, and each exam has two parts--the multiple-choice part and the programming part.
- So, scores[i][j][0] represents the score on the multiple-choice part for the i's student on the j's exam.

Your program displays the total score for each student.

Multidimensional Arrays (Array 3 Dimensi)

```
double[][][] scores ={
 {{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},
 {{4.5, 21.5}, {9.0, 22.5}, {15, 34.5}, {12, 20.5}, {14, 9.5}},
 {{6.5, 30.5}, {9.4, 10.5}, {11, 33.5}, {11, 23.5}, {10, 2.5}},
 {{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},
 {{8.5, 26.5}, {9.4, 52.5}, {13, 36.5}, {13, 24.5}, {16, 2.5}},
 {{9.5, 20.5}, {9.4, 42.5}, {13, 31.5}, {12, 20.5}, {16, 6.5}}};
```

Nilai mhs ke-5
score[5]

Nilai mhs ke-5 ujian ke-4
score[5][4]

Nilai Multiple choice
score[5][4][0] = 16

Nilai essay
score[5][4][1] = 6.5

```
Which student              Which exam        Multiple-choice or essay

                    scores[ i ] [ j ] [ k ]
```

Problem: Weather Information

https://replit.com/@AdeAzurat/DDP2-Pekan05-Array#Weather.java

Suppose a meteorology station records the temperature and humidity at each hour of every day and stores the data for the past ten days in a text file named weather.txt. Each line of the file consists of four numbers that indicate the day, hour, temperature, and humidity. Your task is to write a program that calculates the average daily temperature and humidity for the 10 days.

```
1 1 76.4 0.92
1 2 77.7 0.93
...
10 23 97.7 0.71
10 24 98.7 0.74
```

(a)

```
10 24 98.7 0.74
1 2 77.7 0.93
...
10 23 97.7 0.71
1 1 76.4 0.92
```

(b)

Problem: Guessing Birthday Using Array

https://replit.com/@AdeAzurat/DDP2-Pekan05-Array#GuessBirthdayUsingArray.java

Previous example: `GuessBirthday.java`, gives a program that guesses a birthday.

The program can be simplified by storing the numbers in five sets in a three-dimensional array, and it prompts the user for the answers using a loop.

FAKULTAS ILMU KOMPUTER

Ada Pertanyaan?

- ❑ Berbagai Algoritma Array
- ❑ Bisa rekonstruksi semua latihan
- ❑ Paham semua istilah terkait array
- ❑ Bisa mendeklarasikan dan menginsiasi
- ❑ Passing array ke method dan menerima return nya.

- ➢ Harap lihat lagi daftar lengkap objective pekan ini!
- ➢ Selamat berlatih:

FAKULTAS ILMU KOMPUTER

UNIVERSITAS INDONESIA

Selamat Berlatih!

Perhatikan lagi List Objective yang perlu dikuasai pekan ini.
Baca buku acuan dan berlatih!
Bila masih belum yakin tanyakan ke dosen, tutor atau Kak Burhan.

Semangat !