ITEC1620
Object-Based Programming

Lecture 17

Arrays

Historical Perspective

- What was the first use of computers?

- Computers process data

Data Tables

- A table is a large collection of data logically/sequentially ordered
- Want easy access to each data element
  - Note: each data element has same datatype

Accessing Data Tables

- Starting with a table index, would like to access the data element

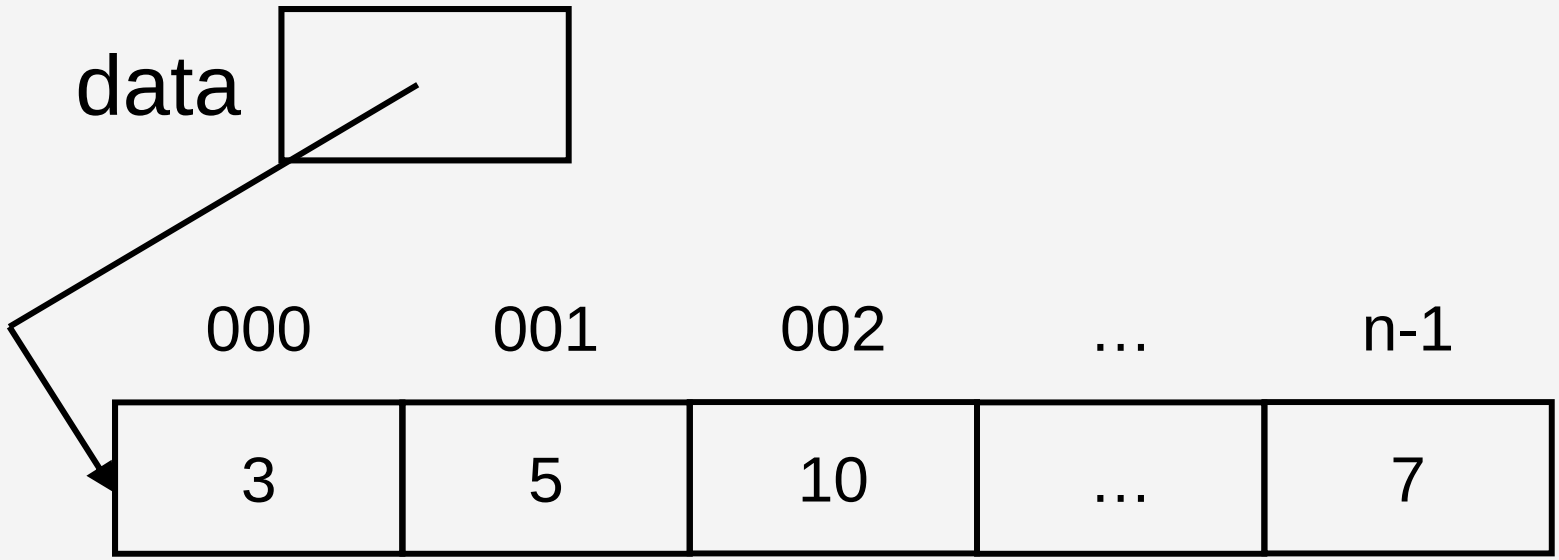
```
public int accessTableItem (int index)
{
        if (index == 1)
                return data1;
        else if (index == 2)
                return data2;
        else if (index == 3)
                return data3;
    …
}
```

Accessing Data Tables II

- What are the problems with the previous access method?
  - if – else if's are time consuming
    - ➢How much time to access n^{th} element?
  - Must know number of elements to set up access method
    - ➢Different access method needed for tables of different size

Accessing Data Tables III

- Better access method – use offsets to directly access the data block/table



Accessing Data Tables IV

- Data block/table is referred to by a reference
- Data elements are 0-indexed
  - Table of n elements will have offsets numbered from 0 to n-1

Blocks of Data

- Where have we seen blocks of data before?
  - Objects
- In JAVA, arrays are implemented as objects

Array Syntax

- Array syntax has conventions which predate JAVA
- JAVA adopts these conventions
  - Syntax is more consistent with C/C++
  - Syntax is inconsistent within JAVA

Array Syntax II

- C/C++ syntax to access i^{th} element of an array referred to by data

  data[i];

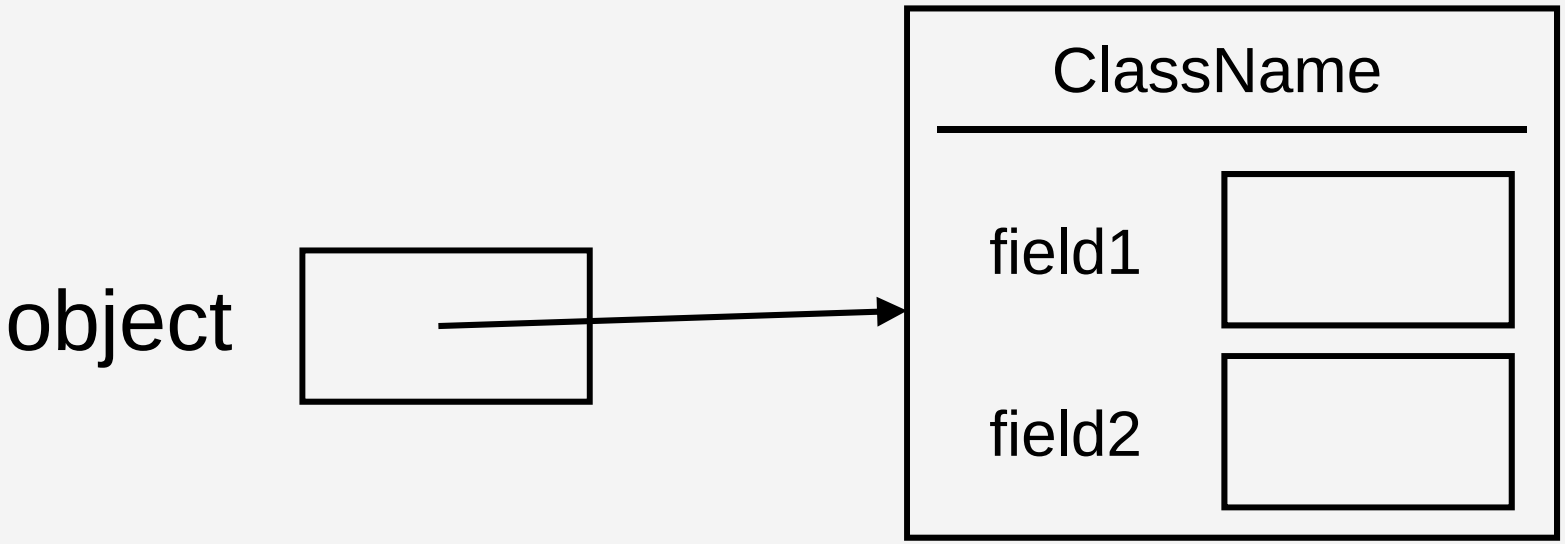
- JAVA adopts this convention

Array Syntax III

- Object syntax

  object.field1;

- Start at object, follow the arrow to field1

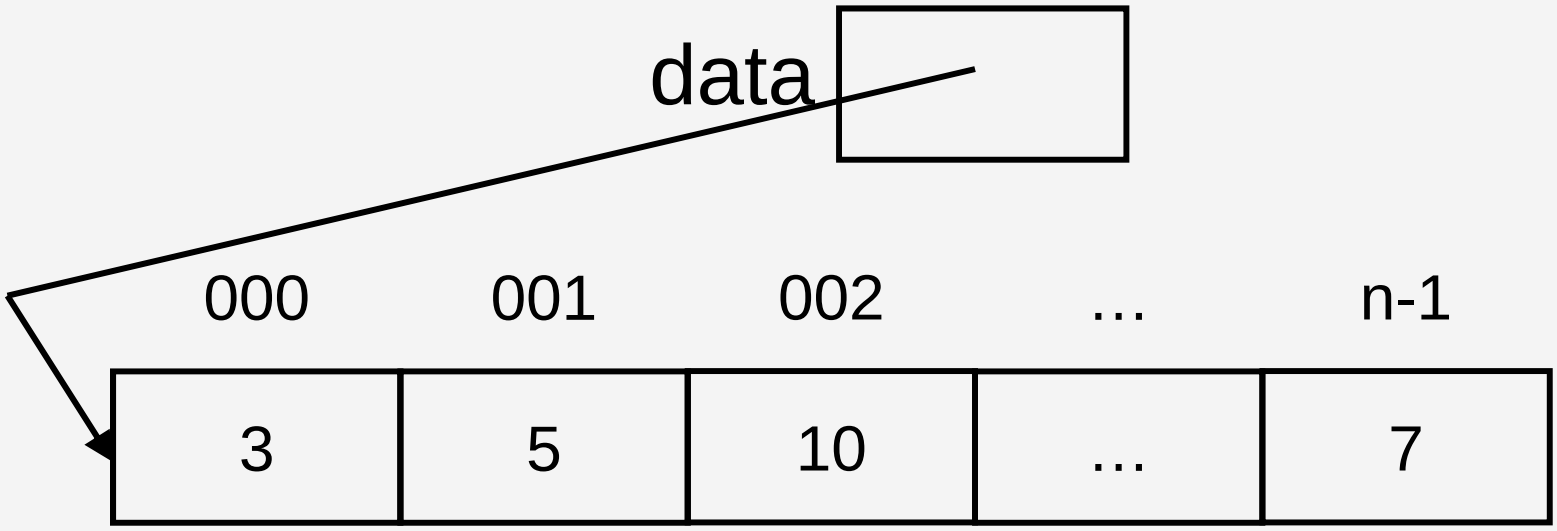


Array Syntax IV

- Array syntax

  data[2];

- Start from data, and offset to element



Array Syntax V

- Declaration of arrays

  <datatype> [] arrayName;

  int[] data;

- Declarations reserve a single space of memory
- [] specifies space has array reference
- <datatype> specifies what it is a reference to

Array Construction

arrayName = new <datatype> [n];

data = new int [10];

- new requests space from OS
- n specifies how much space
- Construction returns a reference
  - Reference (arrow) is stored in arrayName

Array Construction II

- New array gets object default values
  - 0, false, null – numbers, booleans, objects
- Can specify non-default values

new <datatype> [] {value0, value1, …}

data = new int[] {1, 1, 2, 3, 5};

- n calculated from number of values given

Using Arrays

- Arrays are often used with for loops
  - for – known number of times through loop
  - Arrays – known number of elements
    - ➢All arrays have a final public field called "length"

```
for (int i = 0; i < arrayName.length; i++)
   // arrayName[i]
```

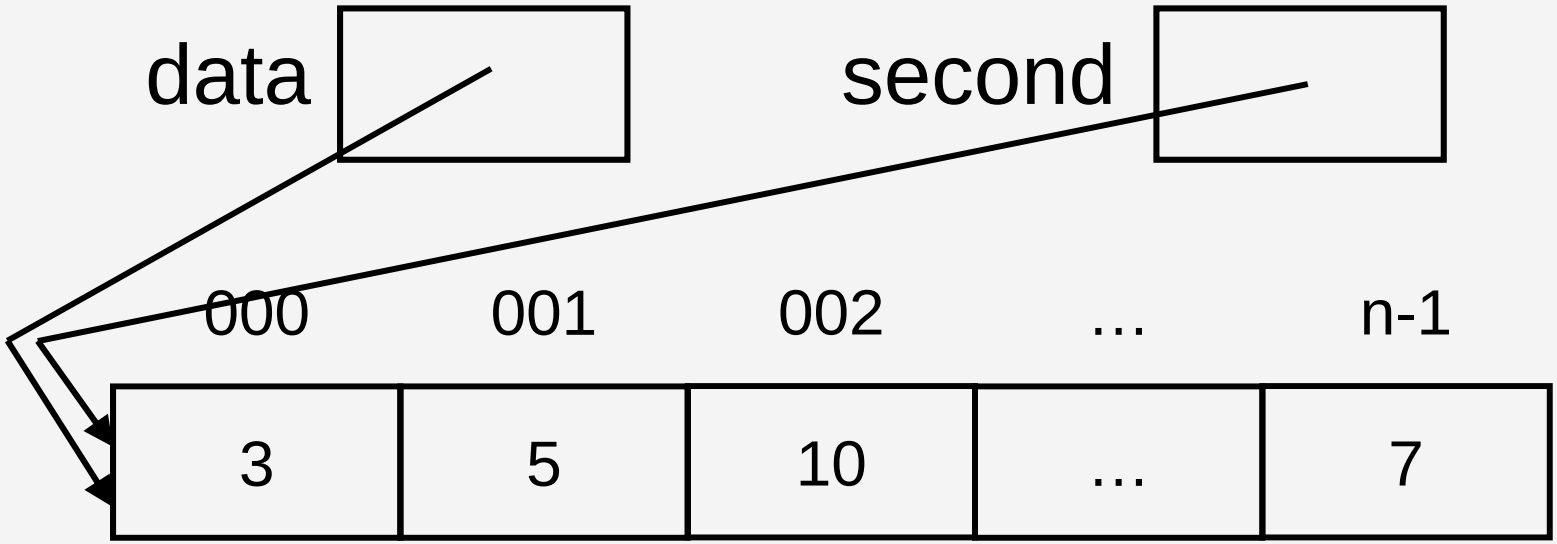
Using Arrays II

- Arrays have elements 0 to length-1
- An array index less than 0 or greater than length-1 will cause a run-time error
- Earlier programming languages (e.g. C/C++) allow alteration of unintended memory locations
- Error checking takes time – JAVA is slow

Using Arrays III

- Arrays are objects
  - Objects can have multiple references

```
int[] second = data;
```



Using Arrays IV

- Arrays can be passed as parameters into and out of methods
  - Array identifier (array head) is passed by copy
  - Array data (table) is passed by reference

Questions?

Problem Types

- Update/process an existing array
  - Assume array exists
  - Write code to update/process it
- Create a new array from an existing array
  - Do not modify original array

Problem Types II

- Partially filled arrays
  - Some methods may require an extra parameter that represents the actual element count
- “length” will not be useful
  - Need a “count” of the actual number of elements

Example

- Write a code fragment that calculates the average of the values for a fully populated array of ints

```
// int[] ar;

double sum = 0.0;
for (int i = 0; i < ar.length; i++)
{
   sum += ar[i];
}
double average = sum / ar.length;
```

Example II

- Write a code fragment that determines the range (the difference between the largest and smallest element) for a fully populated array of ints

```
// int[] ar;
```

Example III

- Write a code fragment that creates a fully populated array of ints containing only the even numbers from a partially populated array of ints

```
// int[] originalArray;
// int count;
// int[] newArray;
```

Readings and Assignments

- Text sections (5th or 6th edition)
  - 7.1, 7.2, 7.6
- Text sections (7th edition)
  - 8.1, 8.2, 8.6
- Arrays Program (Tutorial)