CHAPTER 6

Defining Classes II and Arrays

Shin-Jie Lee (李信杰)
Associate Professor
Computer and Network Center
Department of Computer Science and Information Engineering
National Cheng Kung University

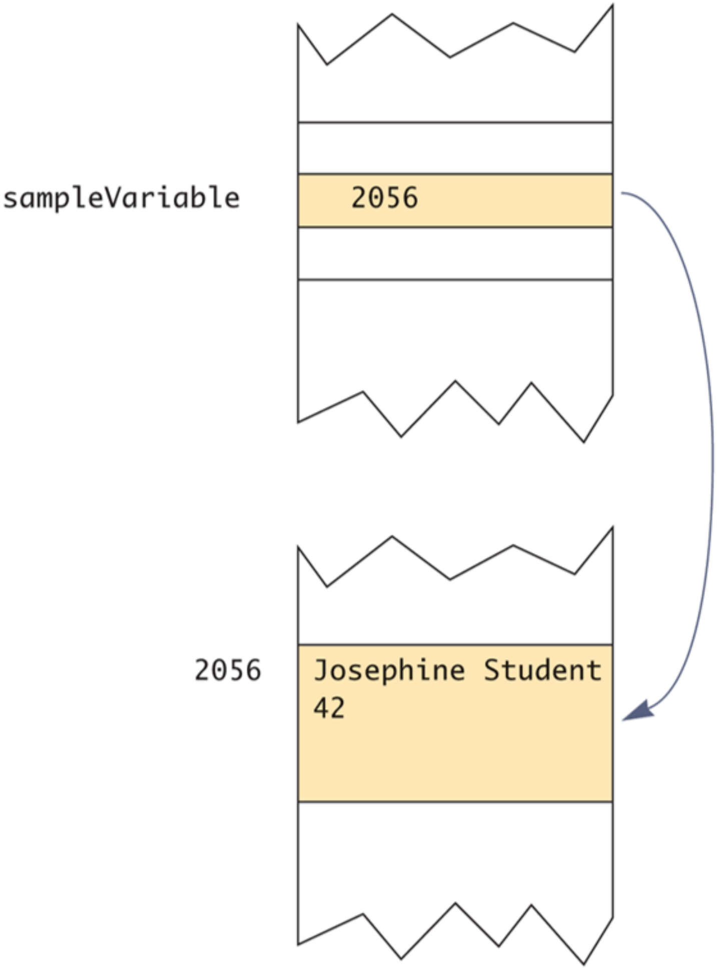
References

- When the variable is **a class type**, **only the memory address (or *reference*)** where its object is located is stored in the memory location assigned to the variable

A variable stores a reference (Class Type)

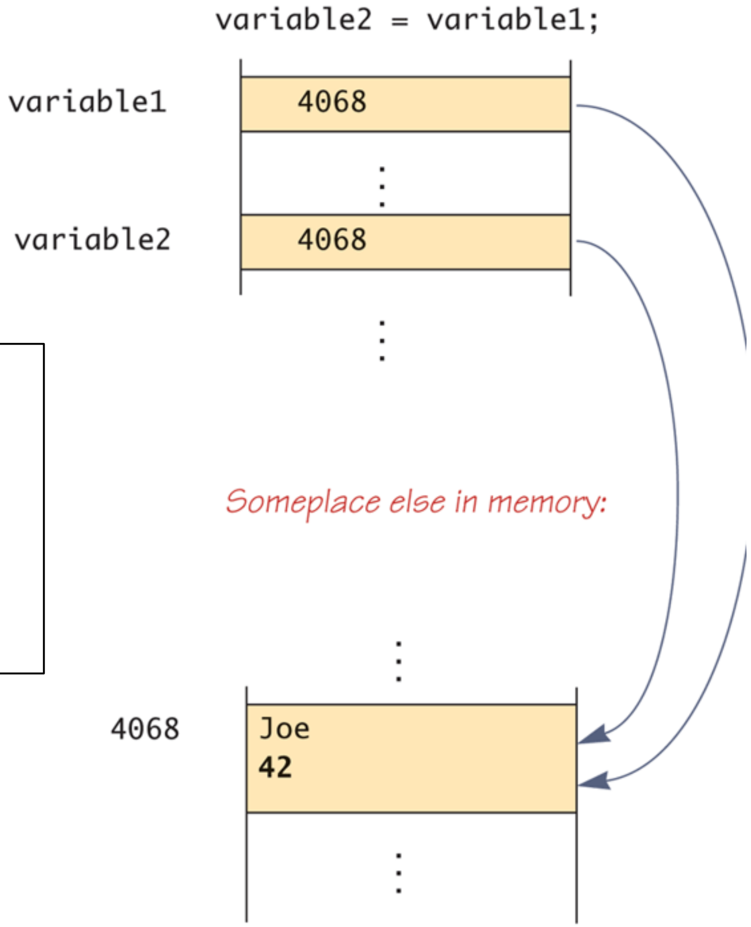
```java
public class Toy {
  private String name;
  private int number;

  public Toy(String name, int number){
    this.name = name;
    this.number = number;
  }
}
```

```java
public class Store {
    public static void main(String[] args){
        Toy sampleVariable =
                        new Toy(“Josephine”,42);
    }
}
```



Two variables can contain the same reference

```
public class Store {
    public static void main(String[] args){
        Toy variable1 = new Toy(“Joe”,42);
        Toy variable2 = variable1;
    }
}
```



Lab

```
public class Cat {

    int age = 1;

    public static void main(String[] args)
    {
        Cat cat1 = new Cat();
        Cat cat2 = cat1;

        cat1.age = 2;
        System.out.println(cat2.age);

     }

}
```

Call-by-reference

- **Class type parameters** appear to behave differently from primitive type parameters
  - ***call-by-reference***

Lab

```
public class Toy
{
    private String name;
    private int number;

    public Toy(String name, int number)
    {
        this.name = name;
        this.number = number;
    }

    public String toString( )
    {
        return (name + " " + number);
    }


    public void set(String newName, int newNumber)
    {
        name = newName;
        number = newNumber;
    }
}
```

Lab

```java
public class Store
{
    public static void main(String[] args)
    {
        Toy toy = new Toy ("Robot Dog", 10);
        System.out.println(toy);

        change(toy);
        System.out.println(toy);
     }

    public static void change(Toy toyobj)
    {
         toyobj.set("Robot Cat",20);
    }

}
```

The Constant `null`

- `null` is a special constant that may be assigned to a **variable of any class type**

```
YourClass yourObject = null;
```

- **It is used to indicate that the variable has no "real value“**

- A method cannot be invoked using a variable that is initialized to `null`
  - `Null Point Exception`

Lab

```
public class Welcome {

  public static void main(String[] args) {

    Welcome wc = new Welcome();
    wc.showMessage();

    Welcome wc2 = null;
    wc2.showMessage();
  }

  public void showMessage(){
    System.out.println("Hi!");
  }
}
```

Creating and Accessing Arrays

- ❑ An array that behaves like this collection of variables, all of type `double`, can be created using one statement as follows:

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

or

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

Creating and Accessing Arrays

- The individual variables that together make up the array are called ***indexed variables***
  - *starting with* `0`

```
score[0], score[1], score[2], score[3], score[4]
```

Creating and Accessing Arrays

- ❑ The number of indexed variables in an array is called the ***length* or *size* of the array**

```
double[] score = new double[5];
System.out.println(score.length);
```

Declaring and Creating an Array

- ❑ An array is declared and created in almost the same way that objects are declared and created:

```
char[] line = new char[80];

double[] reading = new double[count];

Person[] specimen = new Person[100];
```

Initializing Arrays

- An array can be initialized when it is declared

```
int[] age = {2, 12, 1};
```

or

```
double[] score = new double[100];
for (int i = 0; i < score.length; i++)
     score[i] = 42.0;
```

Lab

```
public class ArrayTest {
  public static void main(String[] args) {
  double[] reading = new double[100];
  for (int i = 0; i < reading.length; i++){
    reading[i] = 42.0;
  }

  System.out.println(reading[38]);


  int[] age = {12, 24, 36};
    System.out.println(age.length);
    System.out.println(age[2]);
  }
}
```

Pitfall: Arrays with a Class Base Type

```
Date[] holidayList = new Date[20];
```

- **It does not create 20 objects of the class `Date`**
- Each of these indexed variables are automatically initialized to **`null`**

Pitfall: Arrays with a Class Base Type

- Like any other object, each of the indexed variables requires a separate invocation of a constructor using **`new`**

```
 holidayList[0] = new Date();
                 . . .
holidayList[19] = new Date();
```

or

```
for (int i = 0; i < holidayList.length; i++)
   holidayList[i] = new Date();
```

Lab

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

    String[] names = new String[3];
    System.out.println(names[0]);

    names[0] = "Apple";
    System.out.println(names[0]);

  }
}
```

Array Parameters

- An array can be used as an argument

```
String[] names = new String[10];
myMethod(names);
```

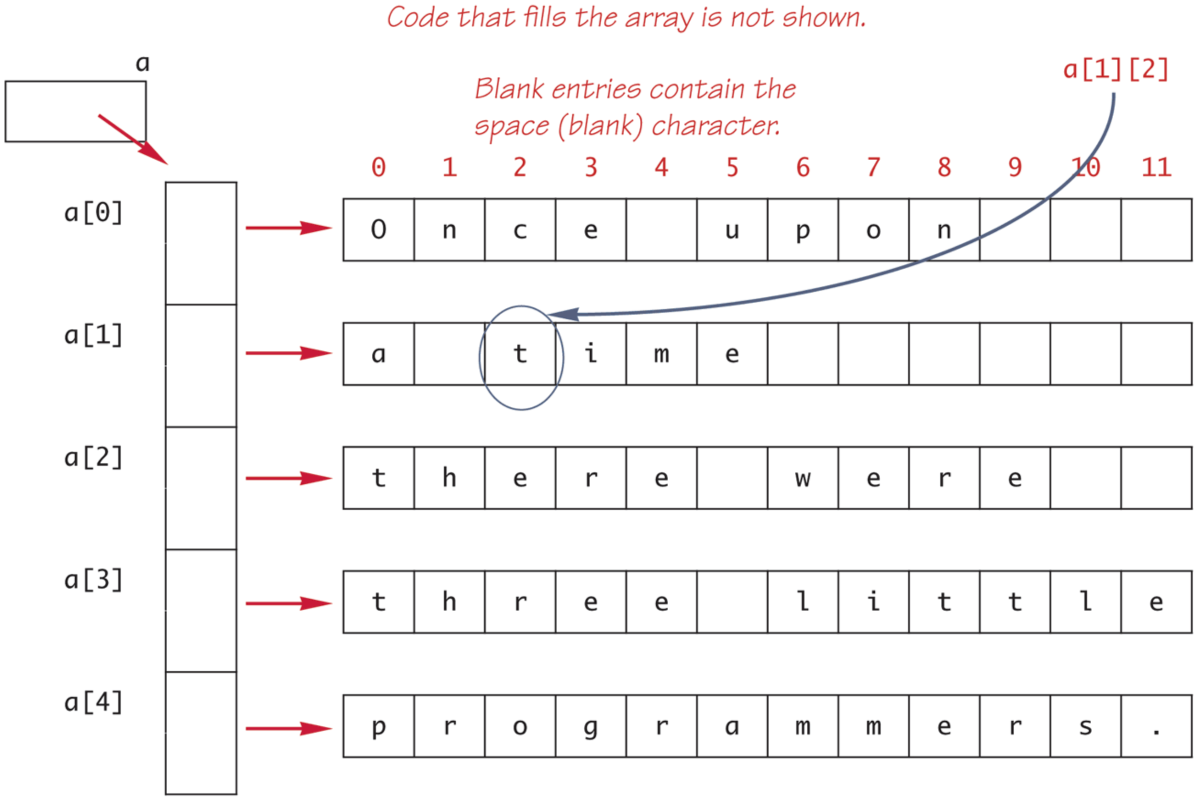
Lab

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

    String[] names = new String[3];
    System.out.println(names[0]);
    names[0] = "Apple";
    System.out.println(names[0]);

    showMessage(names);
  }

  public static void showMessage(String[] message){
    System.out.println(message[0]);
  }
}
```

Multidimensional Arrays

Display 6.17 **Two-Dimensional Array as an Array of Arrays**

```
char[][] a = new char[5][12];
```



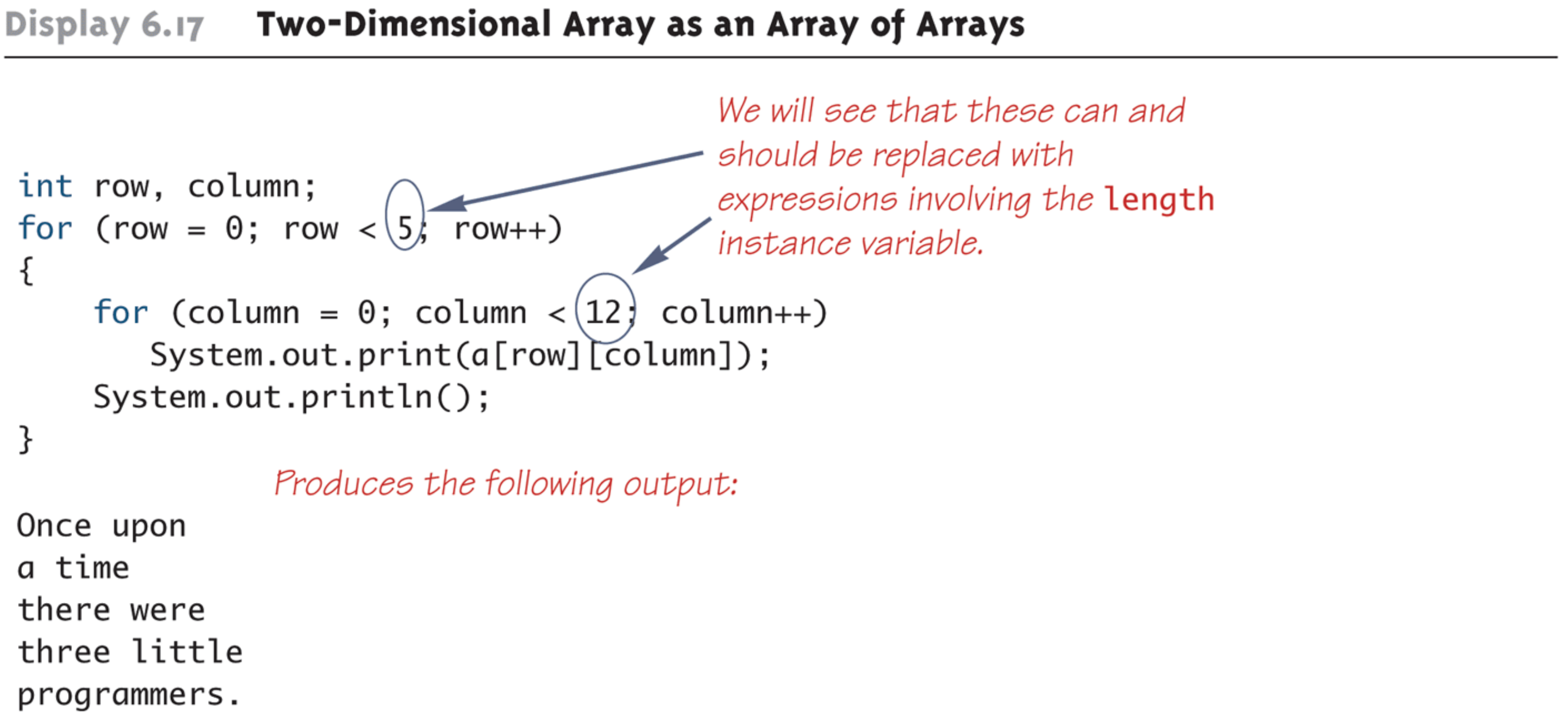
Code that fills the array is not shown.

Blank entries contain the space (blank) character.

	0	1	2	3	4	5	6	7	8	9	10	11
a[0]	O	n	c	e		u	p	o	n			
a[1]	a		t	i	m	e						
a[2]	t	h	e	r	e		w	e	r	e		
a[3]	t	h	r	e	e		l	i	t	t	l	e
a[4]	p	r	o	g	r	a	m	m	e	r	s	.

(continued)

Two-Dimensional Array as an Array of Arrays (Part 2 of 2)



Display 6.17 **Two-Dimensional Array as an Array of Arrays**

```
int row, column;
for (row = 0; row < 5; row++)
{
    for (column = 0; column < 12; column++)
       System.out.print(a[row][column]);
    System.out.println();
}
```

We will see that these can and should be replaced with expressions involving the `length` *instance variable.*

Produces the following output:

```
Once upon
a time
there were
three little
programmers.
```

Using the `length` Instance Variable

Given

```
char[][] page = new char[30][100];
```

Then

`page.length` is equal to 30

`page[0].length` is equal to 100

Lab

```
public class ArrayTest3 {

  public static void main(String[] args){

    int[][] seat = new int[100][10];

    for(int i=0;i<seat.length;i++){
      for(int j=0;j<seat[i].length;j++){
        seat[i][j] = i*j;
      }
    }

    System.out.println(seat[5][3]);
  }
}
```

ArrayList

- An **ArrayList** is a dynamic data structure, meaning items can be added and removed from the list.

- You can then create a new ArrayList object:
  - **ArrayList listTest = new ArrayList( );**

- Add elements to it with the add method:
  - **listTest.add("first item");**

Lab

```
import java.util.ArrayList;

public class ArrayListTest {

  public static void main(String[] args) {
        ArrayList<String> names = new ArrayList<String>();
        names.add("Apple");
        names.add("Orange");
        names.add("pear");

        System.out.println(names.get(1));

  }

}
```

Lab

```
import java.util.ArrayList;

public class ArrayListDemo {

  public static void main(String[] args) {

   ArrayList<String> names = new ArrayList<String>();
   names.add("A");
   names.add("B");
   names.add("C");
   names.remove(1);
   System.out.println(names.get(1));
  }

}
```

Lab

```java
import java.util.ArrayList;

public class ArrayListDemo2 {

  public static void main(String[] args) {

    ArrayList<String> names = new ArrayList<String>();
    names.add("A");
    names.add("B");
    names.add("C");

    for(int i=0;i<3;i++){
      names.remove(i);
    }
    System.out.println(names.size());
  }

}
```

Lab

```
public class Sum
{
   public static void main( String[] args )
   {
      int[] a;
      a = new int[3];

      for ( int i = 0; i < a.length; i++ ){
         a[i] = i + 2;
      }

      int result = 0;
      for ( int i = 0; i < a.length; i++ ){
         result += a[i];
      }

      System.out.println( "Result is:" + result );
   }
}
```

Lab

```java
public class SumTest
{
   public static void main( String[] args )
   {
      int[] a = { 99, 22, 11, 3, 11, 55, 44, 88, 2, -3 };

      int result = 0;

      for ( int i = 0; i < a.length; i++ )
      {
         if ( a[ i ] > 30 )
            result += a[ i ];
      }

      System.out.printf( "Result is: %d\n", result );
   }
}
```

Lab

- ❑ Which statement below initializes array items to contain 3 rows and 2 columns?

a. int[][] items = { { 2, 4 }, { 6, 8 }, { 10, 12 } };.

b. int[][] items = { { 2, 6, 10 }, { 4, 8, 12 } };.

c. int[][] items = { 2, 4 }, { 6, 8 }, { 10, 12 };.

d. int[][] items = { 2, 6, 10 }, { 4, 8, 12 };.

Reference

- “Absolute Java”. Walter Savitch and Kenrick Mock. Addison-Wesley; 5 edition. 2012
- “Java How to Program”. Paul Deitel and Harvey Deitel. Prentice Hall; 9 edition. 2011.
- “A Programmers Guide To Java SCJP Certification: A Comprehensive Primer 3rd Edition”. Khalid Mughal, Rolf Rasmussen. Addison-Wesley Professional. 2008