OBJECT ORIENTED PROGRAMMING

Affefah Qureshi

Department of Computer Science

Iqra University, Islamabad Campus.

ARRAYS

- An *array is a group of like-typed variables that are referred to by a common name*.
- *Arrays of* any type can be created and may have one or more dimensions.
- A specific element in an array is accessed by its index.
- Arrays offer a convenient means of grouping related information.
- *If you are familiar with C/C++, be careful. Arrays in Java work differently than they do in those languages*.
- Normally, an array is a collection of similar type of elements which have a contiguous memory location.

ONE-DIMENSIONAL ARRAYS

- A *one-dimensional array is, essentially, a list of like-typed variables*.

  type var-name[ ]; (or)

  dataType[] arr; (or)

  dataType []arr;

- Here, *type declares the base type of the array*

  int month_days[];

ONE-DIMENSIONAL ARRAYS

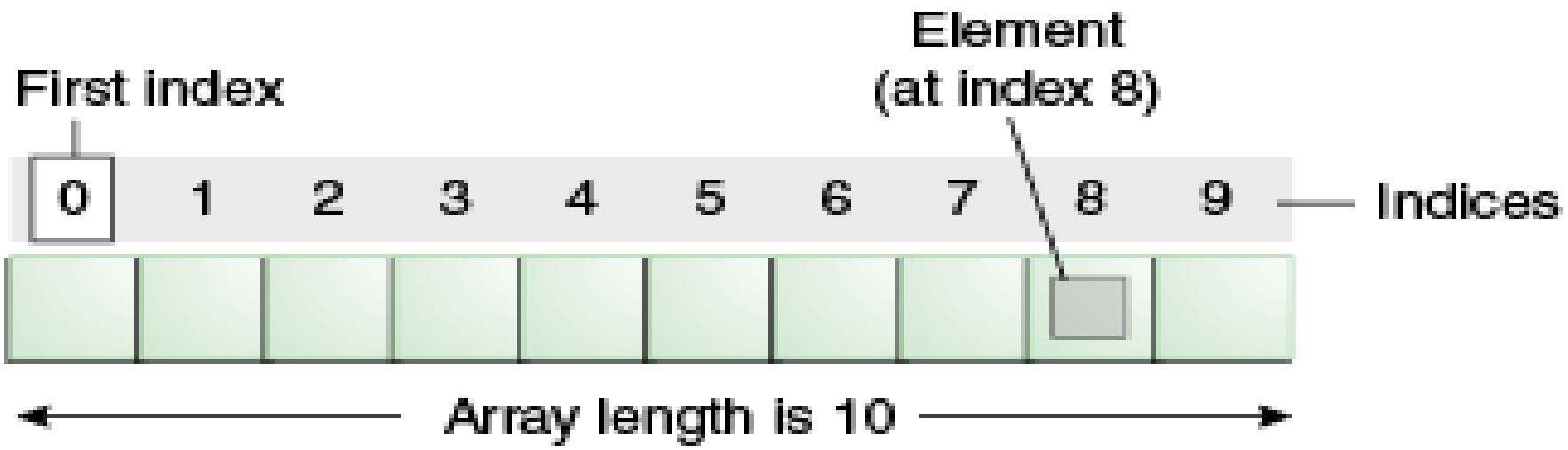
```
int month_days[];
```

- This declaration establishes the fact that month_days is an array variable, no array actually exists. In fact, the value of month_days is set to null, which represents an array with no value.
- To link month_days with an actual, physical array of integers, you must allocate one using new and assign it to month_days.
- new is a special operator that allocates memory.

array-var = new type[size];

```
month_days = new int[12];
```

MYARRAY = NEW INT[10];



EXAMPLE

```
int month_days[];
month_days = new int[12];
```

- After this statement executes, month_days will refer to an array of 12 integers.
- Further, all elements in the array will be initialized to zero.
- Obtaining an array is a two-step process.
  - First, you must declare a variable of the desired array type.
  - Second, you must allocate the memory that will hold the array, using new, and assign it to the array variable.
- Thus, in Java all arrays are dynamically allocated.
- Once you have allocated an array, you can access a specific element in the array by specifying its index within square brackets.
- All array indexes start at zero.

```
month_days[1] = 28;
```

```
class Testarray{
public static void main(String args[]){
        int a[]=new int[5];//declaration and instantiation

        a[0]=10;//initialization
        a[1]=20;
        a[2]=70;
        a[3]=40;
        a[4]=50;

        //traversing array
        for(int i=0;i<a.length;i++)//length is the of array
                System.out.println(a[i]);
}
}
```

INITIALIZATION

- Arrays can be initialized when they are declared.
- The process is much the same as that used to initialize the simple types.
- An array initializer is a list of comma-separated expressions surrounded by curly braces. The commas separate the values of the array elements.
- The array will automatically be created large enough to hold the number of elements you specify in the array initializer.
- There is no need to use new.
    - int a[]={1,2,3,4,5};

```
//Print month name along with no of days
class MonthArray {
 public static void main(String args[]) {
    int month_days[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
    String month_names[] = {"Jan", "Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};

     for(int i=0;i<12;i++)
     {
              System.out.println(month_names[i] + " is having " + month_days[i] + " days.");
     }
 }
}
```

```
C:\Users\Gpcet>java MonthArray
Jan is having 31 days.
Feb is having 28 days.
Mar is having 31 days.
Apr is having 30 days.
May is having 31 days.
Jun is having 30 days.
Jul is having 31 days.
Aug is having 31 days.
Sep is having 30 days.
Oct is having 31 days.
Nov is having 30 days.
Dec is having 31 days.

C:\Users\Gpcet>
```

FOR-EACH LOOP FOR JAVA ARRAY

- We can also print the Java array using for-each loop.
- The Java for-each loop prints the array elements one by one.
- It holds an array element in a variable, then executes the body of the loop.

```
for(data_type variable : array){
        //body of the loop
}
```

```
public class PrintArray {
    public static void main(String [] args){
        String[] array = { "hi", "hello", "java"};

        for(String str : array) {
            System.out.println(str);
        }
    }
}
```

HOME TASK

- Print array elements
- Print array elements in reverse order
- Print odd elements
- Merge two arrays into third array
- Copy even elements to even array and odd elements to odd array from the original array.
- Sort array elements