JAVA Textbook

Chapter 6

Using Arrays in Java Programs

Objectives

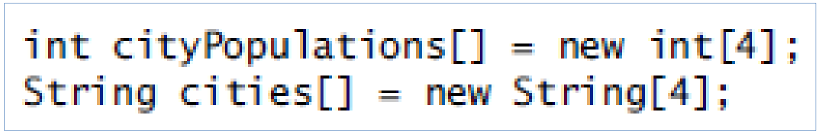
In this chapter, you will learn about:

- Using arrays in Java.
- Search an array for a matching value in Java.
- Using parallel arrays in Java.

Array Basics

- Array: a group of data items, with each
  - having the same data type;
  - being referenced using the same variable name;
  - and being stored in consecutive memory locations.
- Subscript:
  - Used to reference individual elements in an array.
  - A subscript is the position number of a value within an array.
  - In Java, subscript values begin with *0* and end with *n–1*, where *n* is the number of items stored in the array

Declaring Arrays in Java



```
int cityPopulations[] = new int[4];
String cities[] = new String[4];
```

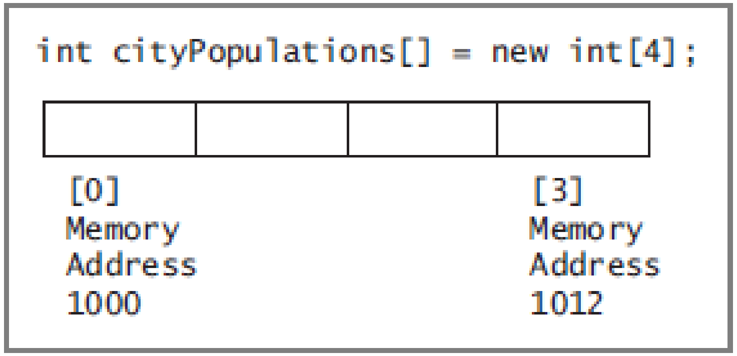
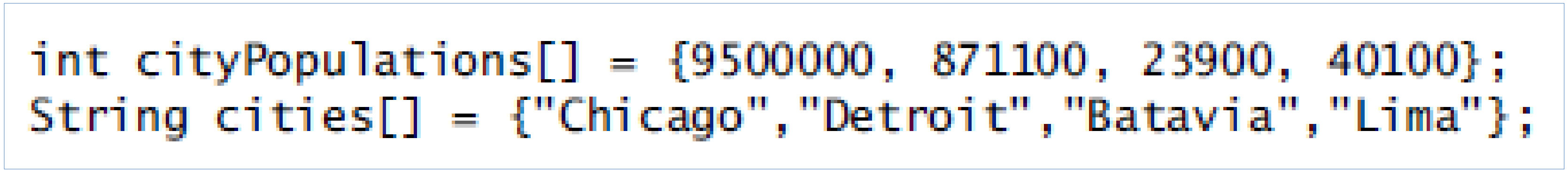


Figure 6-1 Memory allocation for `cityPopulations` array

- It is an error to place anything within the pair of square brackets that follows the array name.
  - The integer that specifies the array size belongs in the second pair of brackets.
- The *new* operator is used to allocate enough memory for the array elements, based on the data type and the array size specified.
  - Data type *int* - 4 bytes each item; data type *double* - 8 bytes each item

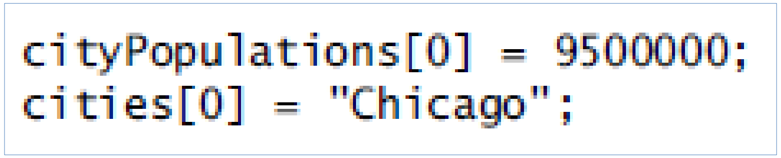
Initializing Arrays in Java



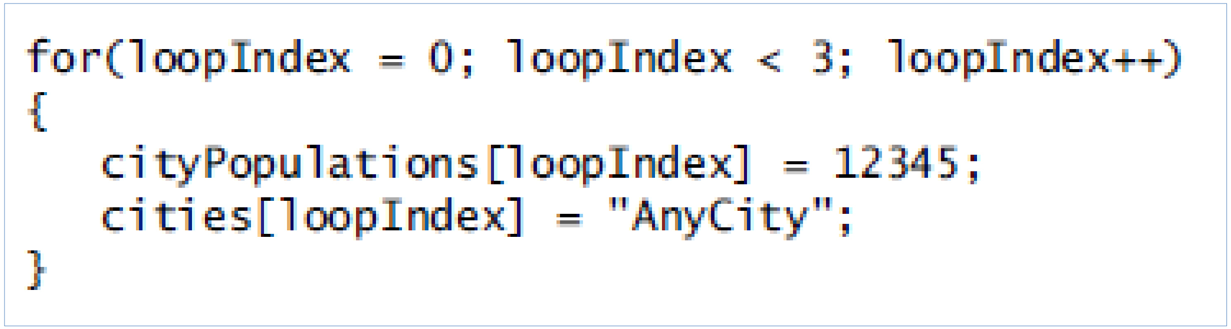
```
int cityPopulations[] = {9500000, 871100, 23900, 40100};
String cities[] = {"Chicago","Detroit","Batavia","Lima"};
```

- Array elements are automatically initialized to 0 for numeric data types, and to the value *null* for references to objects (such as String objects).
  - The keyword *null* is a special value in Java, and it is the zero value for references.
- Arrays can also be initialized with specific values of choice when being declared.

Assigning Values to Array Elements



```
cityPopulations[0] = 9500000;
cities[0] = "Chicago";
```

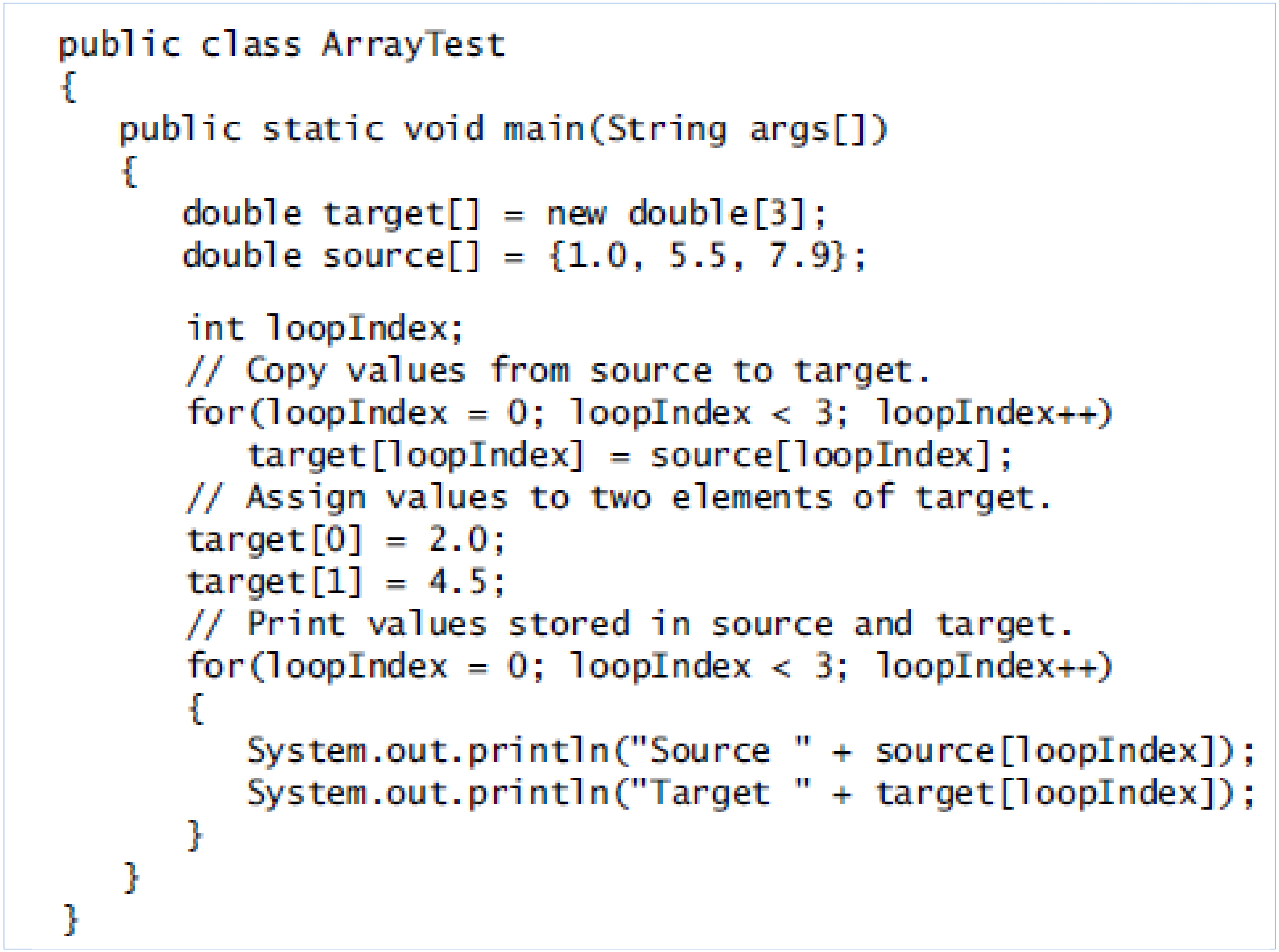


```
for(loopIndex = 0; loopIndex < 3; loopIndex++)
{
   cityPopulations[loopIndex] = 12345;
   cities[loopIndex] = "AnyCity";
}
```

- Assignment statements can be used to provide values for array elements after an array is declared.
- A loop is often used to assign values to the elements in an array

Accessing Array Elements

- Need to access individual locations in an array to work with individual array elements.



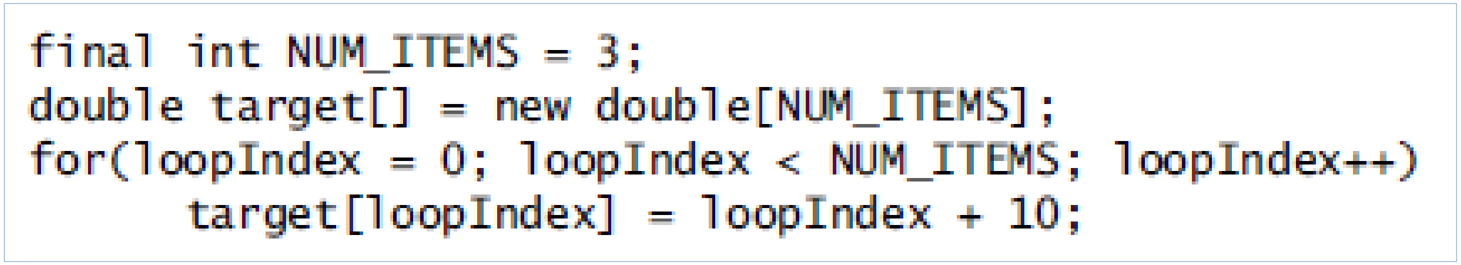
```
public class ArrayTest
{
   public static void main(String args[])
   {
      double target[] = new double[3];
      double source[] = {1.0, 5.5, 7.9};

      int loopIndex;
      // Copy values from source to target.
      for(loopIndex = 0; loopIndex < 3; loopIndex++)
         target[loopIndex] = source[loopIndex];
      // Assign values to two elements of target.
      target[0] = 2.0;
      target[1] = 4.5;
      // Print values stored in source and target.
      for(loopIndex = 0; loopIndex < 3; loopIndex++)
      {
         System.out.println("Source " + source[loopIndex]);
         System.out.println("Target " + target[loopIndex]);
      }
   }
}
```

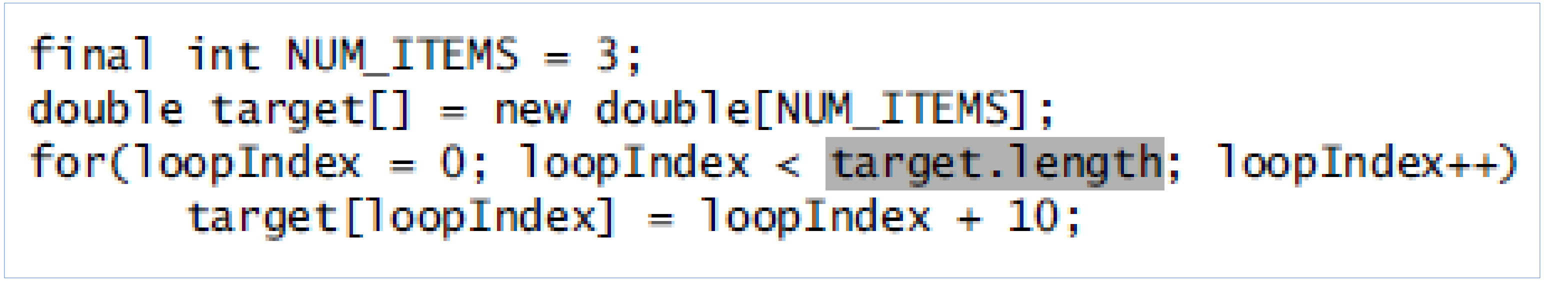
Staying Within the Bounds

- Need to ensure that the subscript values used to access array elements are within the legal bounds.
  - Greater than or equal to 0, and less than the length of the array.
- The Java interpreter checks to make sure that a subscript is in the bounds.
  - It throws an exception if not in the bounds.

Using Constants with Arrays



```
final int NUM_ITEMS = 3;
double target[] = new double[NUM_ITEMS];
for(loopIndex = 0; loopIndex < NUM_ITEMS; loopIndex++)
      target[loopIndex] = loopIndex + 10;
```



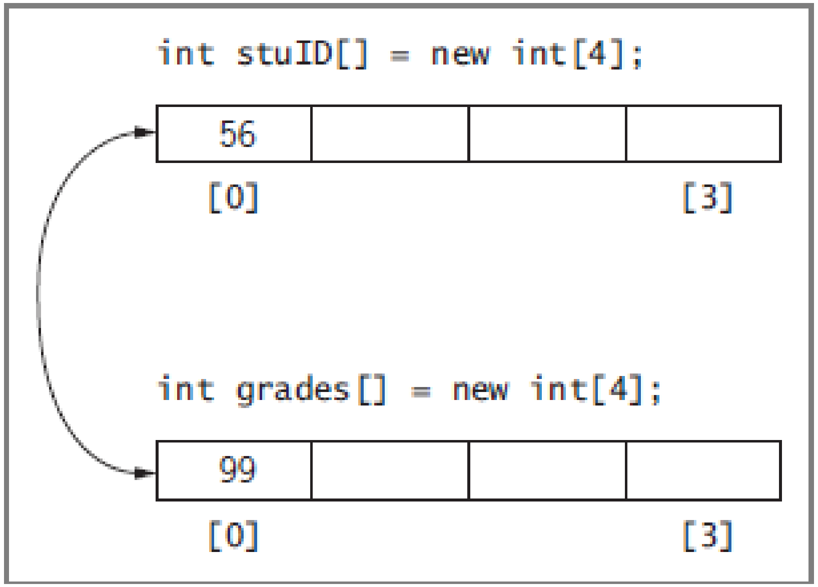
```
final int NUM_ITEMS = 3;
double target[] = new double[NUM_ITEMS];
for(loopIndex = 0; loopIndex < target.length; loopIndex++)
      target[loopIndex] = loopIndex + 10;
```

- Use a named constant to specify size of an array to help stay within the bounds.
- In Java, after an array (e.g., *target*) is declared, its size is automatically stored in a field (e.g., *target.length*).

Searching an Array for an Exact Match

- See example code in Fig. 6-4 on page 105 of the textbook.

Parallel Arrays



```
int stuID[] = new int[4];
int grades[] = new int[4];
```

Figure 6-5 Parallel arrays

- Parallel arrays are used to store values and to maintain a relationship between the items stored in the arrays.
  - This relationship is established by using the same subscript value when accessing each array.
- See example code in Fig. 6-6 on page 110 of the

Thank You!