The process of looking through a data structure for a particular element or elements is called *searching*.

If we search through an array by traversing its elements one by one in the order in which they occur in the array, then we are said to be conducting a *sequential search*. (You might also see it referred to as a *linear search*.) In the following code, for example, a sequential search is made for an Item with a myN value of 3:

  public static void main( String[] args )   
  {   
    Item[] array =    
      {    
        new Item( 1 ),    
        new Item( 5 ),    
        new Item( 3 ),    
        new Item( 4 ),    
        new Item( 7 ),    
        new Item( 10 ),   
      };   
  
    Item.displayArray( array );   
  
    boolean found = false;   
    int i = 0;   
    while ( !found && i < array.length )   
    {   
      if ( array[ i ].getN() == 3 )   
      {   
        found = true;   
        System.out.println( "Found 3 at index " + i );   
      }   
  
      i++;   
    }   
  }

[Show program details »](https://www.eimacs.com/eimacs/mainpage?cid=162149&epid=E2203093181)

public class Item   
{   
  private int myN;   
  
  public Item( int n )   
  {   
    myN = n;   
  }   
  
  public String toString()   
  {   
    return "Item: " + myN;   
  }   
  
  public int getN()    
  {   
    return myN;   
  }   
  
  public static Item[] makeItemArray( int len )   
  {   
    Item[] a = new Item[ len ];   
    int i;   
    for ( i = 0 ; i < len ; i++ )   
      a[ i ] = new Item( i );   
    return a;   
  }   
  
  public static void displayArray( Item[] array )   
  {   
    for ( Item item : array )   
      System.out.println( item );   
  }   
}   
  
public class MainClass   
{   
  public static void main( String[] args )   
  {   
    Item[] array =    
      {    
        new Item( 1 ),    
        new Item( 5 ),    
        new Item( 3 ),    
        new Item( 4 ),    
        new Item( 7 ),    
        new Item( 10 ),   
      };   
  
    Item.displayArray( array );   
  
    boolean found = false;   
    int i = 0;   
    while ( !found && i < array.length )   
    {   
      if ( array[ i ].getN() == 3 )   
      {   
        found = true;   
        System.out.println( "Found 3 at index " + i );   
      }   
  
      i++;   
    }   
  }    
}

Item: 1   
Item: 5   
Item: 3   
Item: 4   
Item: 7   
Item: 10   
Found 3 at index 2

Typically, a sequential search stops once the sought-for element is found. The use of while together with a boolean whose value changes once the element is found can be very effective. An alternative approach is to use a for-loop that is terminated by a return statement once the item is found. This approach is used in the following code:

  public static int sequentialSearch( Item[] array, int target )   
  {   
    for ( int i = 0 ; i < array.length ; i++ )   
    {       
      if ( array[ i ].getN() == target )   
        return i;   
    }   
  
    return -1; // not found   
  }   
  
  public static void main( String[] args )   
  {   
    Item[] array =    
      {    
        new Item( 1 ),    
        new Item( 0 ),    
        new Item( 4 ),    
        new Item( 5 ),    
        new Item( 3 ),    
        new Item( 0 ),   
      };   
  
    Item.displayArray( array );   
  
    int i = sequentialSearch( array, 3 );   
    System.out.println( "Found 3 at index " + i );   
  }

[Show program details »](https://www.eimacs.com/eimacs/mainpage?cid=162149&epid=E2203093181)

Item: 1   
Item: 0   
Item: 4   
Item: 5   
Item: 3   
Item: 0   
Found 3 at index 4

**Exercise 172**

Instead of returning the *index* of the target, a search method might return the target itself. To the definition of Item below, add a class method sequentialSearch that takes as its arguments an Item[] array and an int intTarget, and returns the first Item in array whose myN instance variable is equal to the intTarget. If the method fails to find such an item, it should return null.

public class Item   
{   
  private int myN;   
  
  public Item( int n )   
  {   
    myN = n;   
  }   
  
  public String toString()   
  {   
    return "Item: " + myN;   
  }   
  
  public int getN()    
  {   
    return myN;   
  }   
  
  public static Item[] makeItemArray( int len )   
  {   
    Item[] a = new Item[ len ];   
    int i;   
    for ( i = 0 ; i < len ; i++ )   
      a[ i ] = new Item( i );   
    return a;   
  }   
  
  public static void displayArray( Item[] array )   
  {   
    for ( Item item : array )   
      System.out.println( item );   
  }

public static Item sequentialSearch( Item[] array, int intTarget )

{

    for ( Item item : array )

     {

       if ( item.getN() == intTarget )

         return item;

     }

     return null;

}

}   
  
public class MainClass   
{   
  public static void main( String[] args )   
  {   
    Item[] array =    
      {    
         new Item( 0 ), new Item( 0 ),    
         new Item( 3 ), new Item( 3 ),    
         new Item( 1 ), new Item( 0 ),    
         new Item( 0 ), new Item( 3 )   
      };   
  
    System.out.println( "Found " + Item.sequentialSearch( array,  ) );   
  }   
}

Found Item: 3

**Exercise 173**

Implement a new class method, findItem, that takes two arguments, Item[] array and int target, where it is known that the Items in the input array are sorted into ascending order according to the values of their myN instance variables. The method should conduct a sequential search for an Item having a myN instance variable with value target. If such an Item is present in the input array, findItem should return the Item; if not, it should return a null reference. The method should take advantage of the fact that array is sorted to improve efficiency in the case when there is no Item in array with a myN instance variable having the desired value.

public class Item   
{   
  private int myN;   
  
  public Item( int n )   
  {   
    myN = n;   
  }   
  
  public String toString()   
  {   
    return "Item: " + myN;   
  }   
  
  public int getN()    
  {   
    return myN;   
  }   
  
  public static Item[] makeItemArray( int len )   
  {   
    Item[] a = new Item[ len ];   
    int i;   
    for ( i = 0 ; i < len ; i++ )   
      a[ i ] = new Item( i );   
    return a;   
  }   
    
  public static void displayArray( Item[] array )   
  {   
    for ( Item item : array )   
      System.out.println( item );   
  }

public static Item findItem( Item[] array, int target )

{

     int i = 0;

     while ( i < array.length && array[ i ].getN() <= target )

     {

       if ( array[ i ].getN() == target )

         return array[ i ];

       i++;

     }

     return null;

   }

}﻿

}   
  
public class MainClass   
{   
  public static void main( String[] args )   
  {   
    Item[] array =   
      {    
         new Item( 0 ), new Item( 3 ),    
         new Item( 3 ), new Item( 5 ),    
         new Item( 5 ), new Item( 5 ),    
         new Item( 7 ), new Item( 10 )    
      };   
  
    System.out.println( "Found " + Item.findItem( array,  ) );   
  }   
}

Found null