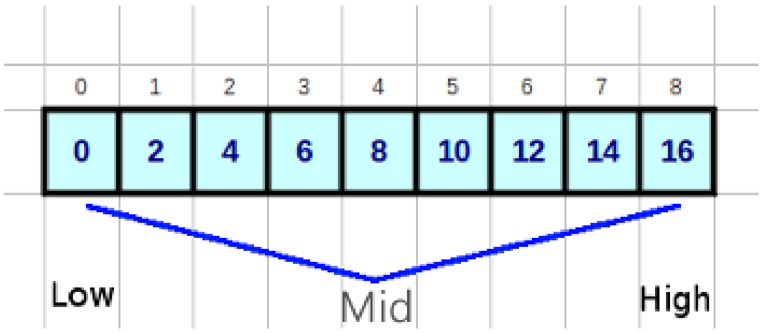
Lab – Binary Search for an Array of ints

See Canvas for due date

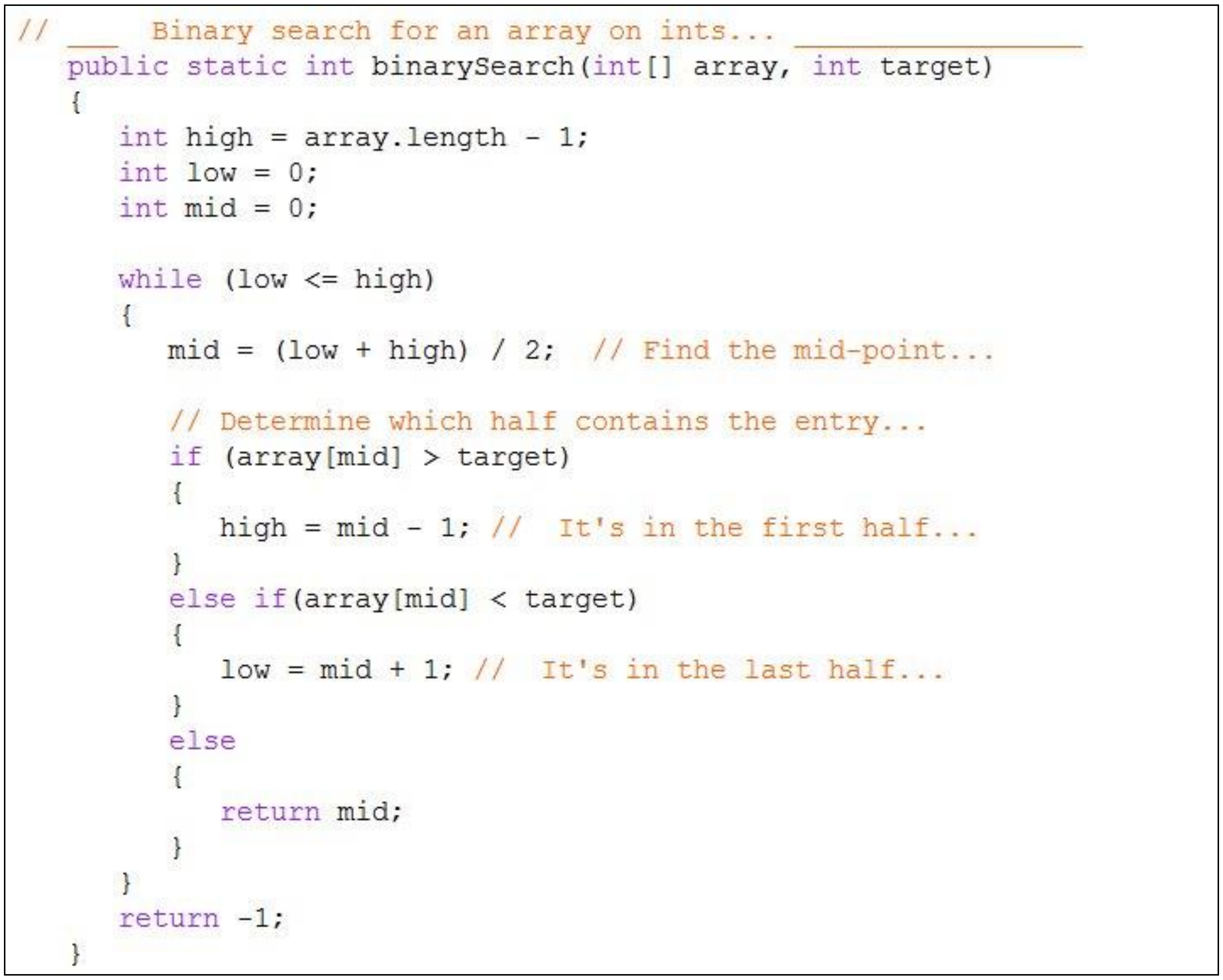


One standard method to search for a given value within a ***sorted*** array is called 'Binary Search'.

Binary Search works by continually dividing an array in half and comparing the value at each midpoint with a target value. If the midpoint value is less than the target, the first half of the array is disregarded and the process is repeated over the second half of the array. If the midpoint value is greater than the target the second half of the array is disregarded and the process is repeated over the first half.

If the target value is found within the array, the method returns the index of the element where the value was found. If the value is not found in the array, -1 is returned.

Binary Search as an image:



```
// ___  Binary search for an array on ints... ________________
   public static int binarySearch(int[] array, int target)
   {
      int high = array.length - 1;
      int low = 0;
      int mid = 0;

      while (low <= high)
      {
         mid = (low + high) / 2;  // Find the mid-point...

         // Determine which half contains the entry...
         if (array[mid] > target)
         {
            high = mid - 1; //  It's in the first half...
         }
         else if(array[mid] < target)
         {
            low = mid + 1; //  It's in the last half...
         }
         else
         {
            return mid;
         }
      }
      return -1;
   }
```

Complete the following lab instructions to create a utility class and a binary search method within that class. Note your utility class will receive more methods as the quarter goes on.

---

1. If you haven't already done so, create a new .java class file named SortSearchUtil.java. Note this utility class will not contain a main method – it's just a number of helper methods.
2. Create a method within the SortSearchUtil class named 'binarySearch'. This method should be defined as public, static and should return a value of type int. The method should also accept incoming parameters of int[] array – the array to search – and a single int – the target value to search for.

3. Write an array looping structure that will divide the array in halves and examine the array's value at the midpoint with respect to the target. If a match is found, the current element's index is returned. If no element's value matches the target value, -1 is returned.
4. Copy and paste the following code into another .java file named 'SearchTester.java'. This code contains a 'main' method that will use your utility class to perform a test of your binarySearch method:

```
public class SearchTester
{
   public static void main(String[] args)
   {
      int[] intAra = {4, 11, 18, 27, 83, 96, 101, 113};

      int found = SortSearchUtil.binarySearch(intAra, 11);

      if (found > -1)
      {
        System.out.println("Value 11 was found at index " + found);
      }
      else
      {
         System.out.println("Value 11 was not found in the array.  Sad...");
      }
   }
 }
```

5. Run your SearchTester main method to confirm the value '11' was found at index 1.

6. Print your code listings and turn in your solution on paper in class.