Implementing Binary Search in Java

Binary Search - Java Implementation

```
//Sorted names and score
String []names = {"Edward","James","John","May","Peter"};
int []testScore = {67,99,57,88,78};

Scanner in = new Scanner(System.in);
System.out.print("Enter name:");
String name = in.nextLine();

int z = binarySearchPos(name, names);
if(z!=-1)
{
        System.out.println("Score of "+name+ " is "+testScore[z]);
}
else
{
        System.out.println("Name not found");
}
```

Binary Search - Java Implementation

```
public static int binarySearchPos(String name, String [] a)
{
        int first=0,last=a.length-1;
        int mid=0;
        while(first<=last)
        {
                mid = first + (last-first)/2; //it is an integer division here.
                if(name.equals(a[mid]))    //found!!
                {
                        return mid; //return immediately, the loop stops here
                }
                else if(name.compareTo(a[mid])<0)
                {
                        last = mid - 1;
                }
                else
                {
                        first = mid + 1;
                }
        }

        return -1;
}
```

int	compareTo(String anotherString) Compares two strings lexicographically.

Binary Search - Java Implementation

- String compareTo() method
  - Compare two Strings.
  - Tells is which String comes first in dictionary order.
  - Example
    - string1.compareTo(string2);
    - Returns 0 if string1 is exactly equals to string2
    - Returns negative if string1 comes before string2 in dictionary order
    - Returns positive if string1 comes after string2 in dictionary order.

Practice

```
String sOne = "apple";
String sTwo = "Pear";

if(sOne.compareTo(sTwo)<0){
        System.out.println(sOne+" comes before "+ sTwo);
}else{
        System.out.println(sOne+" comes aftere "+ sTwo);

}
```

Binary Search - Java Implementation

name -> "May"
a-> {"Edward","James","John","May","Peter"}

```
public static int binarySearchPos(String name, String [] a)
{
        int first=0,last=a.length-1;
        int mid=0;
        while(first<=last)
        {
                mid = first + (last-first)/2; //it is an integer division here.
                if(name.equals(a[mid]))    //found!!
                {
                        return mid; //return immediately, the loop stops here
                }
                else if(name.compareTo(a[mid])<0)
                {
                        last = mid - 1;
                }
                else
                {
                        first = mid + 1;
                }
        }

        return -1;
}
```

int	compareTo(String anotherString) Compares two strings lexicographically.

Binary Search - Java Implementation

```
public static int binarySearchPos(String name, String [] a)
{
        int first=0,last=a.length-1;
        int mid=0;
        while(first<=last)
        {
                mid = first + (last-first)/2; //it is an integer division here.
                if(name.equals(a[mid]))    //found!!
                {
                        return mid; //return immediately, the loop stops here
                }
                else if(name.compareTo(a[mid])<0)
                {
                        last = mid - 1;
                }
                else
                {
                        first = mid + 1;
                }
        }

        return -1;
}
```

first=0
last=4
mid=0

name -> "May"
a-> {"Edward","James","John","May","Peter"}

first

last

mid

int	compareTo(String anotherString) Compares two strings lexicographically.

Binary Search - Java Implementation

```
public static int binarySearchPos(String name, String [] a)
{
        int first=0,last=a.length-1;
        int mid=0;
        while(first<=last) true
        {
                mid = first + (last-first)/2; //it is an integer division here.
                if(name.equals(a[mid]))    //found!!
                {
                        return mid; //return immediately, the loop stops here
                }
                else if(name.compareTo(a[mid])<0)
                {
                        last = mid - 1;
                }
                else
                {
                        first = mid + 1;
                }
        }

        return -1;
}
```

first=0
last=4
mid=0

name -> "May"
a-> {"Edward","James","John","May","Peter"}

first

last

mid

int	compareTo(String anotherString) Compares two strings lexicographically.

Binary Search - Java Implementation

```
public static int binarySearchPos(String name, String [] a)
{
        int first=0,last=a.length-1;
        int mid=0;
        while(first<=last)
        {
                mid = first + (last-first)/2; //it is an integer division here.
                if(name.equals(a[mid]))    //found!!
                {
                        return mid; //return immediately, the loop stops here
                }
                else if(name.compareTo(a[mid])<0)
                {
                        last = mid - 1;
                }
                else
                {
                        first = mid + 1;
                }
        }

        return -1;
}
```

first=0
last=4
mid=2

name -> "May"
a-> {"Edward","James","John","May","Peter"}

int	compareTo(String anotherString) Compares two strings lexicographically.

Binary Search - Java Implementation

```
public static int binarySearchPos(String name, String [] a)
{
        int first=0,last=a.length-1;
        int mid=0;
        while(first<=last)
        {
                mid = first + (last-first)/2; //it is an integer division here.
                if(name.equals(a[mid]))    //found!!
                {       false
                        return mid; //return immediately, the loop stops here
                }
                else if(name.compareTo(a[mid])<0)
                {
                        last = mid - 1;
                }
                else
                {
                        first = mid + 1;
                }
        }

        return -1;
}
```

name -> "May"
a-> {"Edward","James","John","May","Peter"}

first mid last

first=0
last=4
mid=2

int	compareTo(String anotherString) Compares two strings lexicographically.

Binary Search - Java Implementation

```
public static int binarySearchPos(String name, String [] a)
{
        int first=0,last=a.length-1;
        int mid=0;
        while(first<=last)
        {
                mid = first + (last-first)/2; //it is an integer division here.
                if(name.equals(a[mid]))    //found!!
                {
                        return mid; //return immediately, the loop stops here
                }
                else if(name.compareTo(a[mid])<0)
                {
                        last = mid - 1;
                }
                else
                {
                        first = mid + 1;
                }
        }

        return -1;
}
```

name -> "May"
a-> {"Edward","James","John","May","Peter"}

first mid last

first=0
last=4
mid=2

false

int	compareTo(String anotherString) Compares two strings lexicographically.

Binary Search - Java Implementation

```
public static int binarySearchPos(String name, String [] a)
{
        int first=0,last=a.length-1;
        int mid=0;
        while(first<=last)
        {
                mid = first + (last-first)/2; //it is an integer division here.
                if(name.equals(a[mid]))    //found!!
                {
                        return mid; //return immediately, the loop stops here
                }
                else if(name.compareTo(a[mid])<0)
                {
                        last = mid - 1;
                }
                else
                {
                        first = mid + 1;
                }
        }

        return -1;
}
```

name -> "May"
a-> {"Edward","James","John","May","Peter"}

mid first last

first=3
last=4
mid=2

int	compareTo(String anotherString) Compares two strings lexicographically.

Binary Search - Java Implementation

```
public static int binarySearchPos(String name, String [] a)
{
        int first=0,last=a.length-1;
        int mid=0;
        while(first<=last)      true
        {
                mid = first + (last-first)/2; //it is an integer division here.
                if(name.equals(a[mid]))    //found!!
                {
                        return mid; //return immediately, the loop stops here
                }
                else if(name.compareTo(a[mid])<0)
                {
                        last = mid - 1;
                }
                else
                {
                        first = mid + 1;
                }
        }

        return -1;
}
```

name -> "May"
a-> {"Edward","James","John","May","Peter"}

mid first last

first=3
last=4
mid=2

int	compareTo(String anotherString) Compares two strings lexicographically.

Binary Search - Java Implementation

```
public static int binarySearchPos(String name, String [] a)
{
        int first=0,last=a.length-1;
        int mid=0;
        while(first<=last)
        {
                mid = first + (last-first)/2; //it is an integer division here.
                if(name.equals(a[mid]))
                {
                        return mid; //return immediately, the loop stops here
                }
                else if(name.compareTo(a[mid])<0)
                {
                        last = mid - 1;
                }
                else
                {
                        first = mid + 1;
                }
        }

        return -1;
}
```

first=3
last=4
mid=3

name -> "May"
a-> {"Edward","James","John","May","Peter"}

first last mid

int	compareTo(String anotherString) Compares two strings lexicographically.

Binary Search - Java Implementation

```
public static int binarySearchPos(String name, String [] a)
{
        int first=0,last=a.length-1;
        int mid=0;
        while(first<=last)
        {
                mid = first + (last-first)/2;
                if(name.equals(a[mid]))   //found!!
                {       true
                        return mid; //return immediately, the loop stops here
                }
                else if(name.compareTo(a[mid])<0)
                {
                        last = mid - 1;
                }
                else
                {
                        first = mid + 1;
                }
        }

        return -1;
}
```

name -> "May"
a-> {"Edward","James","John","May","Peter"}

first last mid

first=3
last=4
mid=3

int	compareTo(String anotherString) Compares two strings lexicographically.

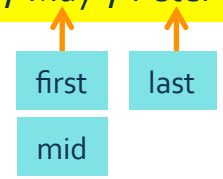
Binary Search - Java Implementation

```
public static int binarySearchPos(String name, String [] a)
{
        int first=0,last=a.length-1;
        int mid=0;
        while(first<=last)
        {
                mid = first + (last-first)/2; //it is an integer division here.
                if(name.equals(a[mid]))
                {
                        return mid; //return immediately, the loop stops here
                }
                else if(name.compareTo(a[mid])<0)
                {
                        last = mid - 1;
                }
                else
                {
                        first = mid + 1;
                }
        }

        return -1;
}
```

first=3
last=4
mid=3

name -> "May"
a-> {"Edward","James","John","May","Peter"}



int	compareTo(String anotherString) Compares two strings lexicographically.

Practice

```
public static int binarySearchPos(String name, String [] a)
{
        int first=0,last=a.length-1;
        int mid=0;
        while(first<=last)
        {
                mid = first + (last-first)/2; //it is an integer division here.
                if(name.equals(a[mid]))
                {
                        return mid; //return immediately, the loop stops here
                }
                else if(name.compareTo(a[mid])<0)
                {
                        last = mid - 1;
                }
                else
                {
                        first = mid + 1;
                }
        }

        return -1;
}
```

Practice

```
//Sorted names and score
String []names = {"Edward","James","John","May","Peter"};
int []testScore = {67,99,57,88,78};

Scanner in = new Scanner(System.in);
System.out.print("Enter name:");
String name = in.nextLine();

int z = binarySearchPos(name, names);
if(z!=-1)
{
        System.out.println("Score of "+name+ " is "+testScore[z]);
}
else
{
        System.out.println("Name not found");
}
```