**PROGRAM:**

import java.util.Arrays;  
class Product{  
 int productId;  
 String productName;  
 String category;  
 public Product(int productId, String productName, String category){  
 this.productId=productId;  
 this.productName=productName;  
 this.category=category;  
 }  
 public String toString(){  
 return "[ " +productId+", "+productName+", "+category+"]";  
 }  
 public static Product linearSearch(Product[] products,String name){  
 for(Product product: products){  
 if(product.productName.equalsIgnoreCase(name)){  
 return product;  
 }  
 }  
 return null;  
 }  
 public static Product binarySearch(Product[] products,String name){  
 int left=0,right=products.length-1;  
 while(left<=right){  
 int mid=(left+right)/2;  
 int compare=products[mid].productName.compareToIgnoreCase(name);  
 if(compare==0)  
 return products[mid];  
 else if(compare<0)  
 left=mid+1;  
 else  
 right=mid-1;  
 }  
 return null;  
 }  
 public static void main(String[] args){  
 Product[] products={new Product(1,"Planet and beauty","Hair care"),  
 new Product(2,"Shampoo","Personal care"),  
 new Product(3,"Mobile Phone","Electronics"),  
 new Product(4,"XO blue pen","Stationary")};  
 Arrays.*sort*(products, (a, b) -> a.productName.compareToIgnoreCase(b.productName));  
 Product foundLinear=*linearSearch*(products,"XO blue pen");  
 System.*out*.println("Linear Search Result:"+foundLinear);  
 Product foundBinary=*binarySearch*(products,"Planet and beauty");  
 System.*out*.println("Binary Search Result:"+foundBinary);  
 }  
  
}  
/\*Analysis:  
Binary Search is suitable for large sorted arrays which has time complexity of O(log n) whereas the time complexity of linear search is O(n) which makes it   
unsuitable for large data arrays which are unsorted \*/ /\* Big O Notation helps in analyzing upper bound of an algorithm's run time.  
It works on worst case analysis which is always O(n).  
/\* Big O Notation helps in analyzing upper bound of an algorithm's run time.  
It works on worst case analysis which is always O(n).\*/

**OUTPUT:**

