**1. Explain Big O Notation and Scenarios**

**Big O notation** is a mathematical notation used to describe the asymptotic behaviour of an algorithm’s time or space complexity — i.e. how the algorithm scales as the input size (n) grows.

**Best, Average, Worst Case Scenarios for Search:**

| **Algorithm** | **Best Case** | **Average Case** | **Worst Case** |
| --- | --- | --- | --- |
| **Linear Search** | O(1) (found at first index) | O(n) | O(n) (not found or last index) |
| **Binary Search** | O(1) (middle index) | O(log n) | O(log n) (not found) |

Big O helps choose the most efficient algorithm — smaller Big O ⇒ faster as n grows.

**2.**

**Product.java:**

public class Product {  
 private int productId;  
 private String productName;  
 private String Category;  
  
 public Product(int productId,String productName,String Category){  
 this.productId=productId;  
 this.productName=productName;  
 this.Category=Category;  
 }  
 public int getProductId(){  
 return productId;  
 }  
 public String getProductName(){  
 return productName;  
 }  
 public String getCategory() {  
 return Category;  
 }  
}

**searchAlgo.java:**

import java.util.\*;  
  
public class searchAlgo {  
 public static Product linearSearch(Product[] product,int reqId){  
 for(Product p: product){  
 if(reqId==p.getProductId()){  
 return p;  
 }  
 }  
 return null;  
 }  
 public static Product binarySearch(Product[] product,int reqId){  
 int left=0;  
 int right= product.length-1;  
 while(left<=right){  
 int mid=left+(right-left)/2;  
 int midId=product[mid].getProductId();  
 if(midId==reqId) return product[mid];  
 if(midId<reqId) left=mid+1;  
 else right=mid-1;  
 }  
 return null;  
 }  
 public static void sortByProductId(Product[] products) {  
 Arrays.*sort*(products, Comparator.*comparingInt*(Product::getProductId));  
 }  
}  
class Main {  
 public static void main(String[] args) {  
 Product[] products = {  
 new Product(103, "Laptop", "Electronics"),  
 new Product(101, "Phone", "Electronics"),  
 new Product(105, "Tablet", "Electronics"),  
 new Product(102, "Headphones", "Accessories"),  
 new Product(104, "Monitor", "Electronics")  
 };  
 searchAlgo s=new searchAlgo();  
 Product found = s.*linearSearch*(products, 105);  
 System.*out*.println("Linear Search Found: " + found);  
 s.*sortByProductId*(products);  
 Product foundBinary = s.*binarySearch*(products, 105);  
 System.*out*.println("Binary Search Found: " + foundBinary);  
 }  
}

**3.** Use **binary search** for fast lookups on a large, sorted list of products. For unsorted or very small lists, **linear search** is simpler.