```
//1. Sort an ArrayList of integers in ascending and descending order.
package Sorting_And_Searching_Collections;
import java.util.ArrayList;
import java.util.Collections;
public class Challenge1 {
        public static void main(String[] args) {
                ArrayList<Integer> integers = new ArrayList<>();
                integers.add(5);
                integers.add(3);
                integers.add(8);
                integers.add(21);
                integers.add(6);
                integers.add(2);
    Collections.sort(integers);
    System.out.println("Ascending: " + integers);
    Collections.sort(integers, Collections.reverseOrder());
    System.out.println("Descending: " + integers);
        }
}
```

```
//2. Use Collections.binarySearch() to find an element in a sorted list.
package Sorting_And_Searching_Collections;
import java.util.ArrayList;
import java.util.Collections;
public class Challenge2 {
        public static void main(String[] args) {
                 ArrayList<Integer> list = new ArrayList<>();
             Collections.addAll(list, 40, 10, 70, 80, 50);
             Collections.sort(list);
             int index = Collections.binarySearch(list, 70);
             if (index != -1 && index < list.size())
                System.out.println("The element is at index: " + index);
             else
                System.out.println("Element not found in the list.");
        }
}
```

```
//3. Sort a list of custom objects like Employees by name using Comparator.
package Sorting_And_Searching_Collections;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
class Employee {

  private int id;
  private String name;
```

```
       public Employee(String name, int id) {
              this.name = name;
              this.id = id;
       }
       public String getName() {
              return name;
       }
       public int getId() {
              return id;
       }

       public String toString() {
    return "Employee-"+name+" (id: " + id+")";
  }

}
public class Challenge3 {
       public static void main(String[] args) {
              List<Employee> employees = new ArrayList<>();
    employees.add(new Employee("John", 1));
    employees.add(new Employee("Alice", 2));
    employees.add(new Employee("Bob", 3));
    employees.add(new Employee("David", 4));
    System.out.println("Before sorting: "+employees);
    Collections.sort(employees, new Comparator<Employee>() {
      public int compare(Employee e1, Employee e2) {
        return e1.getName().compareToIgnoreCase(e2.getName());
      }
    });
    System.out.println("\nAfter sorting by name:");
    for (Employee emp : employees) {
      System.out.println(emp);
    }
       }
}


//4. You have a list of products with prices. Sort them by price and then search for a product within
a specific price range.
package Sorting_And_Searching_Collections;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
class Product {

  private String productName;
  private double price;
```

```
        public Product(String productName, double price) {
                this.productName = productName;
                this.price = price;
        }
        public String getProductName() {
                return productName;
        }
        public double getPrice() {
                return price;
        }
        public String toString() {
    return productName+" - " + price;
  }

}
public class Challenge4 {
        public static void main(String[] args) {
                List<Product> products = new ArrayList<>();
                products.add(new Product("Shirt", 500.0));
                products.add(new Product("Car", 400000.0));
                products.add(new Product("Mobile", 29000.0));
                products.add(new Product("Bike", 100000.0));
    System.out.println("Products Before sorting:\n"+products);
    products.sort(Comparator.comparingDouble(Product::getPrice));
    System.out.println("Product after sorting:\n"+products);
    System.out.println("\nProducts in the price range 1000.0 - 30000.0:\n");
    for (Product p : products) {
      if (p.getPrice() >= 1000.0 && p.getPrice() <= 30000.0) {
        System.out.println(p);
      }
    }
        }
}
```

```
//5. Build a leaderboard system that keeps players sorted by scores (highest first). Allow
searching for a specific player's rank.
package Sorting_And_Searching_Collections;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
class Player {
  private String name;
  private int score;
  public Player(String name, int score) {
    this.name = name;
    this.score = score;
  }
```

```
   public String getName() {
       return name;
   }
   public int getScore() {
       return score;
   }
   public String toString() {
       return name + " - " + score;
   }
}
public class Challenge5 {
             public static void main(String[] args) {
                           List<Player> leaderboard = new ArrayList<>();
                     leaderboard.add(new Player("Alice", 67));
                     leaderboard.add(new Player("Bob", 75));
                     leaderboard.add(new Player("Charlie", 64));
                     leaderboard.add(new Player("David", 62));
                     leaderboard.add(new Player("Eve", 97));
                     leaderboard.sort(Comparator.comparingInt(Player::getScore).reversed());
                     System.out.println("Leaderboard:");
                     int rank = 1;
                     for (Player p : leaderboard) {
                         System.out.println(rank + ". " + p);
                         rank++;
                     }
                     String searchName = "Charlie";
                     int playerRank = getPlayerRank(leaderboard, searchName);
                     if (playerRank != -1) {
                         System.out.println("\n" + searchName + "'s rank is: " + playerRank);
                     } else {
                         System.out.println("\nPlayer " + searchName + " not found.");
                     }
                 }
                 public static int getPlayerRank(List<Player> leaderboard, String name) {
                     for (int i = 0; i < leaderboard.size(); i++) {
                         if (leaderboard.get(i).getName().equalsIgnoreCase(name)) {
                             return i + 1;
                         }
                     }
                                       return -1;
             }
}
```