# **Collections Utility Methods in Java**

## **Introduction**

This Java program demonstrates the use of various utility functions from the Collections class to perform operations on an ArrayList. The program takes user input for a list of integers and performs sorting, shuffling, reversing, and binary searching using Collections methods.

Additionally, exception handling mechanisms are implemented to handle invalid inputs and empty lists.

## **Functionality Overview**

1. **Sorting the List**
   1. Method Used: Collections.sort(list);
   2. Sorts the list in **ascending order**.
2. **Shuffling the List**
   1. Method Used: Collections.shuffle(list);
   2. Randomly rearranges the elements of the list.
3. **Reversing the List**
   1. Method Used: Collections.reverse(list);
   2. Reverses the order of elements in the list.
4. **Binary Search on the List**
   1. Method Used: Collections.binarySearch(list, searchElement);
   2. The list is **sorted before searching**, as binary search requires a sorted list.
   3. Returns the **index** of the element if found; otherwise, returns a negative value.

## **Exception Handling & Potential Risks**

### **1. Handling Non-Integer Input**

* **Risk**: The user might enter a non-integer value, causing scanner.nextInt() to throw an InputMismatchException.
* **Solution**: Before reading input, scanner.hasNextInt() is used to validate if the next input is an integer. If not, an IllegalArgumentException is thrown.

### **2. Handling an Empty List**

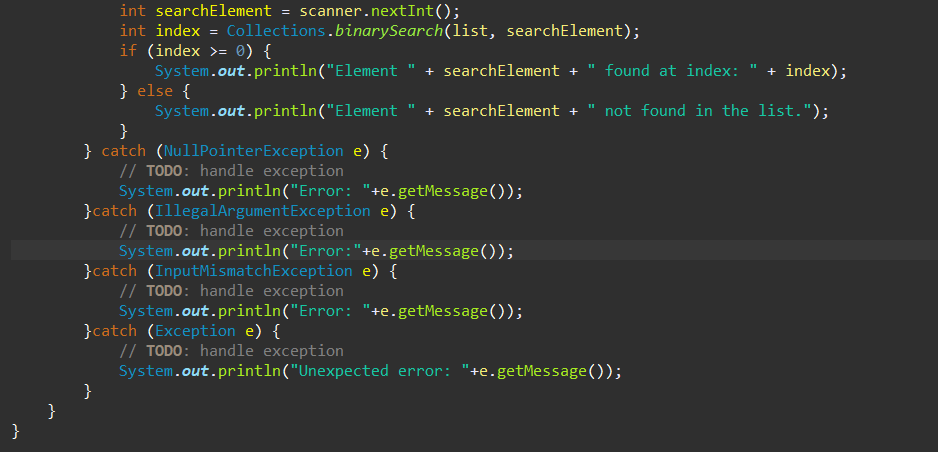
* **Risk**: If the user enters 0 elements, operations like sorting, shuffling, and binary search are not valid.
* **Solution**: A NullPointerException is thrown when the list is empty.

### **3. Handling Invalid Binary Search Input**

* **Risk**: The user might enter a non-integer value while searching for an element.
* **Solution**: Similar to list input validation, scanner.hasNextInt() is used before reading the search element.

### **4. Handling Unexpected Errors**

* **Risk**: Any unforeseen runtime error might crash the program.
* **Solution**: A generic Exception catch block is added to display unexpected errors gracefully.



## **Conclusion**

This program efficiently demonstrates Collections utility functions while ensuring robustness through proper exception handling. The implemented checks prevent runtime errors caused by invalid input, an empty list, or unexpected failures.

The use of Collections methods enhances efficiency, making the program easy to maintain and extend for additional operations.