```
 import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Scanner;
import java.util.regex.*;

public class AdvancedArrayListOps {

    private List<String> stringList;

    public AdvancedArrayListOps() {
        stringList = new ArrayList<>();
    }

    public void append(String s){
        stringList.add(s);
    }

    public void insert(int i,String s){
        if(i>=0 && i<=stringList.size()){
            stringList.add(i,s);
        }
        else{
            System.out.println("Invalid index");
        }
    }

    public boolean search(String s){
        return stringList.contains(s);
    }

    public void startswith(String s){
        List<String> result = new ArrayList<>();
        for(String str : stringList){
            if(str.toLowerCase().startsWith(s.toLowerCase())){
                result.add(str);
            }
        }
        Collections.sort(result);
        System.out.println("Strings starting with '" + s + "': " + result);
    }

    public void sortstring(boolean ascending){
        if(ascending){
            Collections.sort(stringList);
        }
        else{
            stringList.sort(Collections.reverseOrder());
        }
    }

    public void regexsearch(String regex){
        Pattern p = Pattern.compile(regex, Pattern.CASE_INSENSITIVE);
        System.out.print("Regex matches: ");
        for(String str : stringList){
            Matcher m = p.matcher(str);
            if(m.find()){
                System.out.print(str + " ");
            }
        }
        System.out.println();
    }

    public void removeDuplicates(){
        LinkedHashSet<String> set = new LinkedHashSet<>(stringList);
        stringList.clear();
        stringList.addAll(set);
    }

    public void partialMatch(String substring){
        System.out.print("Strings containing '" + substring + "': ");
        for(String str : stringList){
            if(str.toLowerCase().contains(substring.toLowerCase())){
                System.out.print(str + " ");
            }
        }
        System.out.println();
    }

    public void sortByLength(){
        stringList.sort(Comparator.comparingInt(String::length));
    }

    public void printlist(){
        System.out.println("List: " + stringList);
    }

    public static void main(String[] args){
        AdvancedArrayListOps ops = new AdvancedArrayListOps();
        Scanner scanner = new Scanner(System.in);

        int choice;
        do {
            System.out.println("\n--- MENU ---");
            System.out.println("1. Append");
            System.out.println("2. Insert at index");
            System.out.println("3. Search");
            System.out.println("4. Partial Match Search");
```

```
System.out.println("5. Regex Search");
System.out.println("6. Starts With");
System.out.println("7. Sort Ascending");
System.out.println("8. Sort Descending");
System.out.println("9. Remove Duplicates");
System.out.println("10. Sort by Length");
System.out.println("11. Print List");
System.out.println("0. Exit");
System.out.print("Enter choice: ");
choice = scanner.nextInt();
scanner.nextLine();

switch(choice){
    case 1:
        System.out.print("Enter string to append: ");
        ops.append(scanner.nextLine());
        break;
    case 2:
        System.out.print("Enter index: ");
        int index = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter string: ");
        ops.insert(index, scanner.nextLine());
        break;
    case 3:
        System.out.print("Enter string to search: ");
        System.out.println("Found? " + ops.search(scanner.nextLine()));
        break;
    case 4:
        System.out.print("Enter substring: ");
        ops.partialMatch(scanner.nextLine());
        break;
    case 5:
        System.out.print("Enter regex pattern: ");
        ops.regexsearch(scanner.nextLine());
        break;
    case 6:
        System.out.print("Enter starting letter: ");
        ops.startswith(scanner.nextLine());
        break;
    case 7:
        ops.sortstring(true);
        System.out.println("Sorted ascending.");
        break;
    case 8:
        ops.sortstring(false);
        System.out.println("Sorted descending.");
        break;
    case 9:
        ops.removeDuplicates();
```

```
                    System.out.println("Duplicates removed.");
                    break;
                case 10:
                    ops.sortByLength();
                    System.out.println("Sorted by length.");
                    break;
                case 11:
                    ops.printlist();
                    break;
                case 0:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice!");
            }
        } while(choice != 0);

        scanner.close();
    }
}
```