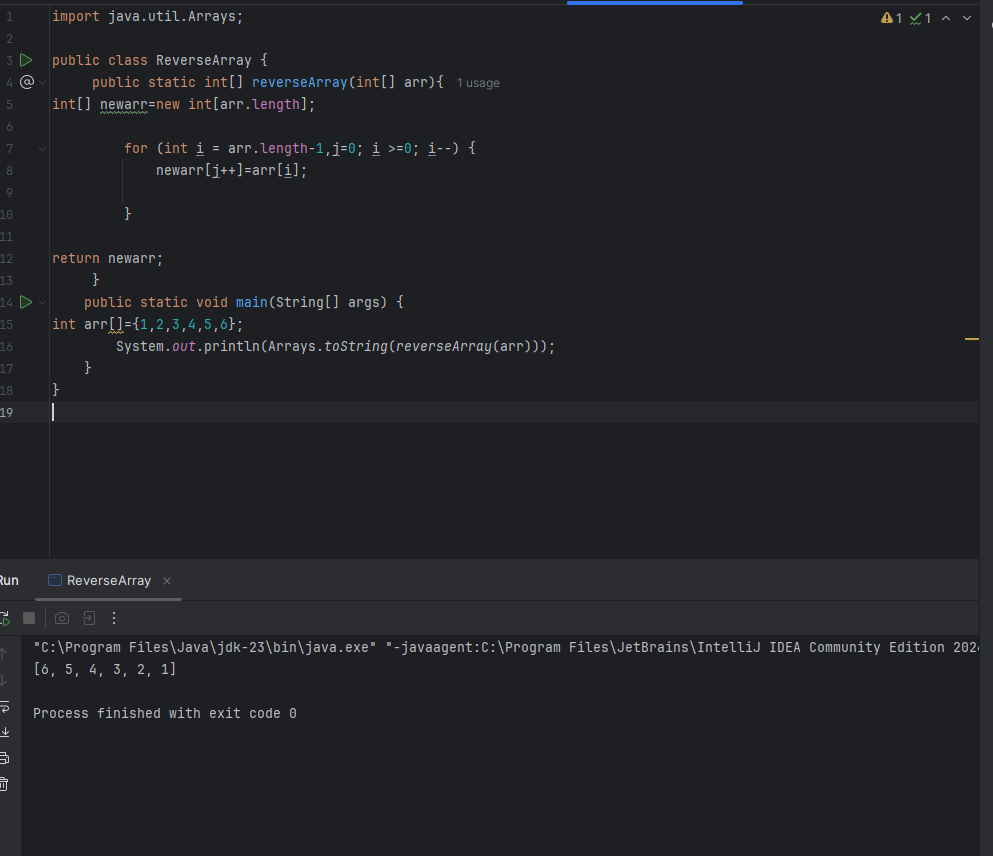
## **Basic Practice Task**

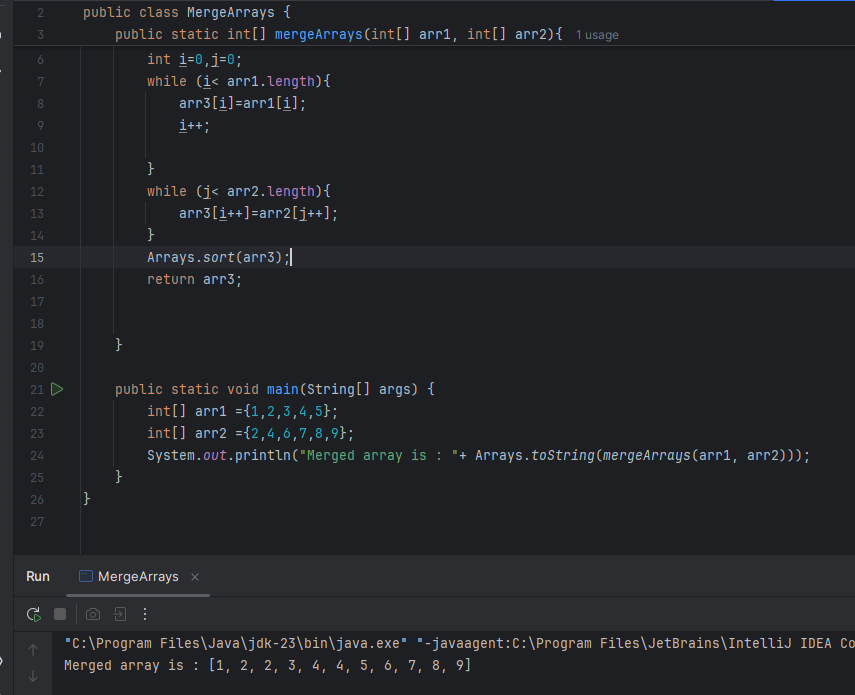
1. Implement a program to **reverse an array in place**.

import java.util.Arrays;  
  
public class ReverseArray {  
 public static int[] reverseArray(int[] arr){  
int[] newarr=new int[arr.length];  
  
 for (int i = arr.length-1,j=0; i >=0; i--) {  
 newarr[j++]=arr[i];  
  
 }  
  
return newarr;  
 }  
 public static void main(String[] args) {  
int arr[]={1,2,3,4,5,6};  
 System.*out*.println(Arrays.*toString*(*reverseArray*(arr)));  
 }  
}



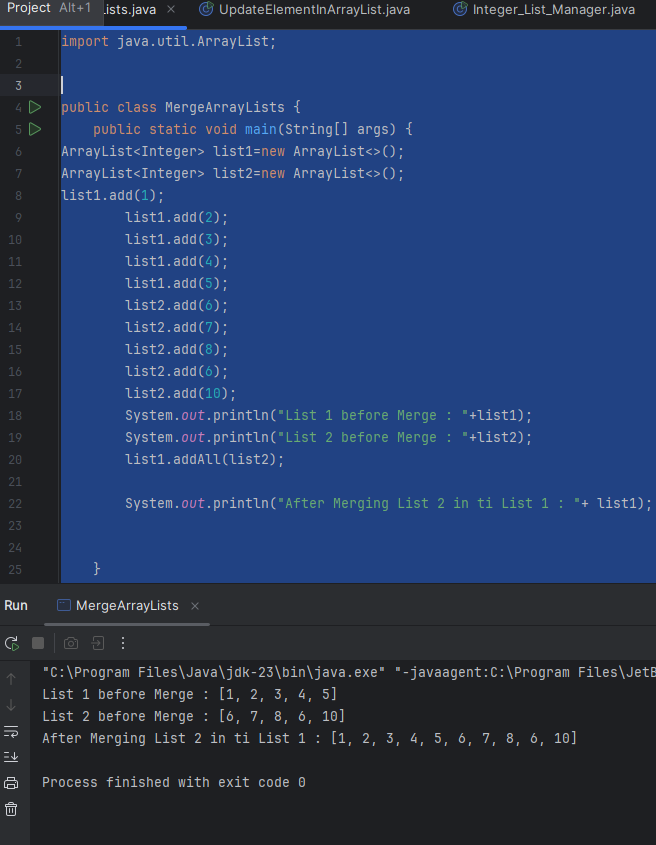
1. Implement a **merge function** for two sorted arrays.

import java.util.Arrays;  
public class MergeArrays {  
 public static int[] mergeArrays(int[] arr1, int[] arr2){  
 int[] arr3=new int[arr1.length+arr2.length];  
  
 int i=0,j=0;  
 while (i< arr1.length){  
 arr3[i]=arr1[i];  
 i++;  
  
 }  
 while (j< arr2.length){  
 arr3[i++]=arr2[j++];  
 }  
 Arrays.*sort*(arr3);  
 return arr3;  
  
  
 }  
  
 public static void main(String[] args) {  
 int[] arr1 ={1,2,3,4,5};  
 int[] arr2 ={2,4,6,7,8,9};  
 System.*out*.println("Merged array is : "+ Arrays.*toString*(*mergeArrays*(arr1, arr2)));  
 }  
}



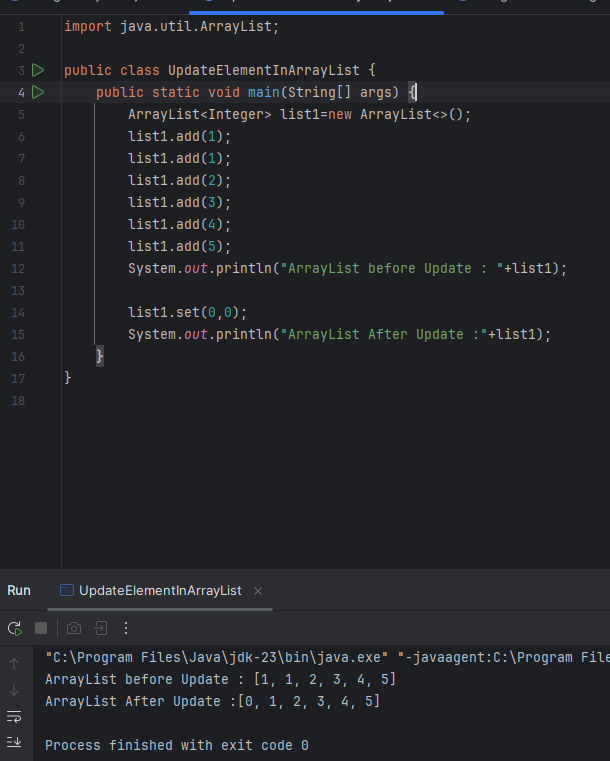
1. **Merge two ArrayList** into a single list.

import java.util.ArrayList;  
  
  
public class MergeArrayLists {  
 public static void main(String[] args) {  
ArrayList<Integer> list1=new ArrayList<>();  
ArrayList<Integer> list2=new ArrayList<>();  
list1.add(1);  
 list1.add(2);  
 list1.add(3);  
 list1.add(4);  
 list1.add(5);  
 list2.add(6);  
 list2.add(7);  
 list2.add(8);  
 list2.add(6);  
 list2.add(10);  
 System.*out*.println("List 1 before Merge : "+list1);  
 System.*out*.println("List 2 before Merge : "+list2);  
 list1.addAll(list2);  
  
 System.*out*.println("After Merging List 2 in ti List 1 : "+ list1);  
  
  
 }  
}



1. **Update an element** in an ArrayList at a specific index.

import java.util.ArrayList;  
  
public class UpdateElementInArrayList {  
 public static void main(String[] args) {  
 ArrayList<Integer> list1=new ArrayList<>();  
 list1.add(1);  
 list1.add(1);  
 list1.add(2);  
 list1.add(3);  
 list1.add(4);  
 list1.add(5);  
 System.*out*.println("ArrayList before Update : "+list1);  
  
 list1.set(0,0);  
 System.*out*.println("ArrayList After Update :"+list1);  
 }  
}



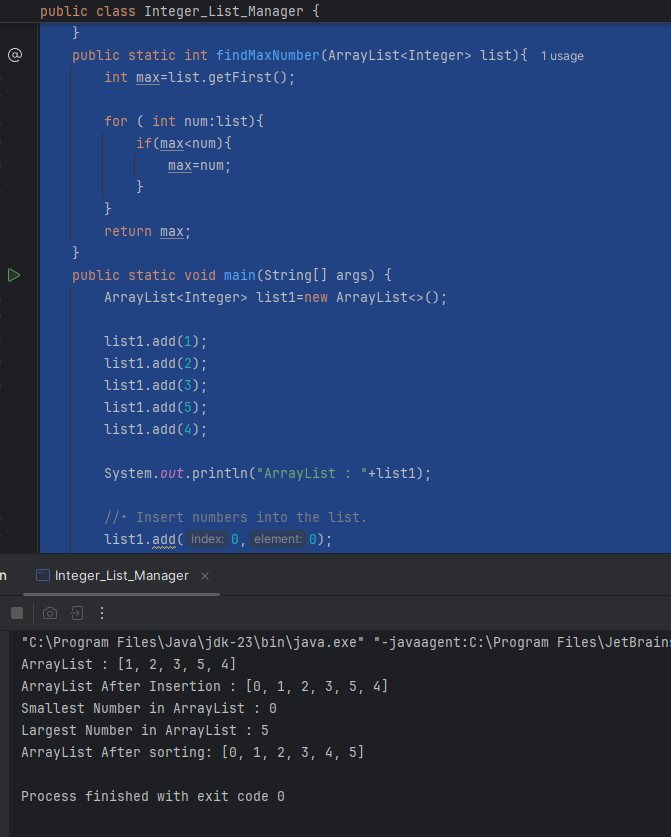
## **Scenario-Based Case Studies**

### **Case Study 1: Number List Operations**

**Task:** Implement an **Integer List Manager** with ArrayList<Integer>.  
Features:

* Insert numbers into the list.
* Find the largest and smallest number.
* Sort the list in ascending order.

import java.util.ArrayList;  
import java.util.Collections;  
  
public class Integer\_List\_Manager {  
 public static int findSmallNumber(ArrayList<Integer> list){  
 int minimum=list.get(0);  
  
 for ( int num:list){  
 if(minimum>num){  
 minimum=num;  
 }  
 }  
 return minimum;  
 }  
 public static int findMaxNumber(ArrayList<Integer> list){  
 int max=list.getFirst();  
  
 for ( int num:list){  
 if(max<num){  
 max=num;  
 }  
 }  
 return max;  
 }  
 public static void main(String[] args) {  
 ArrayList<Integer> list1=new ArrayList<>();  
  
 list1.add(1);  
 list1.add(2);  
 list1.add(3);  
 list1.add(5);  
 list1.add(4);  
  
 System.*out*.println("ArrayList : "+list1);  
  
 //• Insert numbers into the list.  
 list1.add(0,0);  
  
 System.*out*.println("ArrayList After Insertion : "+list1);  
 // • Find the largest and smallest number.  
  
 System.*out*.println("Smallest Number in ArrayList : "+*findSmallNumber*(list1));  
 System.*out*.println("Largest Number in ArrayList : "+*findMaxNumber*(list1));  
  
 //• Sort the list in ascending order.  
 Collections.*sort*(list1);  
 System.*out*.println("ArrayList After sorting: "+list1);  
 }  
}



## **Case Study 2: Student Records System**

**Scenario:**

A **school** maintains student roll numbers using an array-based system. The system should allow the modification and retrieval of records.

**Tasks:**

1. **Insert a new student's roll number** at a specified position.
2. **Search for a roll number** to check if a student is enrolled.
3. **Delete a student's roll number** if they transfer to another school.
4. **Count how many students are currently enrolled.**
5. **Find the student with the highest roll number** (assuming roll numbers are unique and increasing

import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Scanner;  
  
public class StudentRecordsSystem {  
 private ArrayList<Integer> studentRollNumbers;  
  
 public StudentRecordsSystem() {  
 studentRollNumbers = new ArrayList<>();  
 }  
  
 public void insertRollNumber(int rollNumber, int position) {  
 if (position < 0 || position > studentRollNumbers.size()) {  
 System.*out*.println("Invalid position.");  
 return;  
 }  
 studentRollNumbers.add(position, rollNumber);  
 }  
  
 public boolean searchRollNumber(int rollNumber) {  
 return studentRollNumbers.contains(rollNumber);  
 }  
  
 public boolean deleteRollNumber(int rollNumber) {  
 return studentRollNumbers.remove(Integer.*valueOf*(rollNumber));  
 }  
  
 public int countEnrolledStudents() {  
 return studentRollNumbers.size();  
 }  
  
 public int findHighestRollNumber() {  
 if (studentRollNumbers.isEmpty()) {  
 System.*out*.println("No students enrolled.");  
 return -1;  
 }  
 return Collections.*max*(studentRollNumbers);  
 }  
  
 public void displayAllStudents() {  
 System.*out*.println("Enrolled Students: " + studentRollNumbers);  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 StudentRecordsSystem recordsSystem = new StudentRecordsSystem();  
 int choice;  
  
 do {  
 System.*out*.println("\nStudent Records System Menu:");  
 System.*out*.println("1. Insert Roll Number");  
 System.*out*.println("2. Search Roll Number");  
 System.*out*.println("3. Delete Roll Number");  
 System.*out*.println("4. Count Enrolled Students");  
 System.*out*.println("5. Find Highest Roll Number");  
 System.*out*.println("6. Display All Students");  
 System.*out*.println("0. Exit");  
 System.*out*.print("Enter your choice: ");  
 choice = scanner.nextInt();  
  
 switch (choice) {  
 case 1:  
 System.*out*.print("Enter roll number: ");  
 int rollNumber = scanner.nextInt();  
 System.*out*.print("Enter position: ");  
 int position = scanner.nextInt();  
 recordsSystem.insertRollNumber(rollNumber, position);  
 break;  
 case 2:  
 System.*out*.print("Enter roll number to search: ");  
 rollNumber = scanner.nextInt();  
 System.*out*.println("Enrolled: " + recordsSystem.searchRollNumber(rollNumber));  
 break;  
 case 3:  
 System.*out*.print("Enter roll number to delete: ");  
 rollNumber = scanner.nextInt();  
 System.*out*.println("Deleted: " + recordsSystem.deleteRollNumber(rollNumber));  
 break;  
 case 4:  
 System.*out*.println("Number of enrolled students: " + recordsSystem.countEnrolledStudents());  
 break;  
 case 5:  
 System.*out*.println("Highest Roll Number: " + recordsSystem.findHighestRollNumber());  
 break;  
 case 6:  
 recordsSystem.displayAllStudents();  
 break;  
 case 0:  
 System.*out*.println("Exiting...");  
 break;  
 default:  
 System.*out*.println("Invalid choice, please try again.");  
 }  
 } while (choice != 0);  
 scanner.close();  
 }  
}

