### 1 Array Traversal

public class ArrayTraversal {

public static void main(String[] args) {

int[] arr = {1, 2, 3, 4, 5};

for (int i = 0; i < arr.length; i++) {

System.out.print(arr[i] + " ");

}

}

}

### 2. Finding the Largest Element in an Array

public class LargestElement {

public static void main(String[] args) {

int[] arr = {1, 2, 3, 4, 5};

int max = arr[0];

for (int i = 1; i < arr.length; i++) {

if (arr[i] > max) {

max = arr[i];

}

}

System.out.println("Largest element is: " + max);

}

}

### 3. Finding the Smallest Element in an Array

public class SmallestElement {

public static void main(String[] args) {

int[] arr = {1, 2, 3, 4, 5};

int min = arr[0];

for (int i = 1; i < arr.length; i++) {

if (arr[i] < min) {

min = arr[i];

}

}

System.out.println("Smallest element is: " + min);

}

}

### 4. Reversing an Array

public class ReverseArray {

public static void main(String[] args) {

int[] arr = {1, 2, 3, 4, 5};

for (int i = arr.length - 1; i >= 0; i--) {

System.out.print(arr[i] + " ");

}

}

}

### 5. Searching for an Element in an Array

public class LinearSearch {

public static void main(String[] args) {

int[] arr = {1, 2, 3, 4, 5};

int target = 3;

boolean found = false;

for (int i = 0; i < arr.length; i++) {

if (arr[i] == target) {

found = true;

break;

}

}

if (found) {

System.out.println("Element found");

} else {

System.out.println("Element not found");

}

}

}

#### 5 Binary Search (requires sorted array)

import java.util.Arrays;

public class BinarySearch {

public static void main(String[] args) {

int[] arr = {1, 2, 3, 4, 5};

int target = 3;

Arrays.sort(arr); // Ensure array is sorted

int result = Arrays.binarySearch(arr, target);

if (result >= 0) {

System.out.println("Element found at index: " + result);

} else {

System.out.println("Element not found");

}

}

}

### 6. Sorting an Array

public class BubbleSort {

public static void main(String[] args) {

int[] arr = {5, 2, 9, 1, 5, 6};

for (int i = 0; i < arr.length - 1; i++) {

for (int j = 0; j < arr.length - 1 - i; j++) {

if (arr[j] > arr[j + 1]) {

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

for (int num : arr) {

System.out.print(num + " ");

}

}

}

#### 7 Selection Sort

public class SelectionSort {

public static void main(String[] args) {

int[] arr = {5, 2, 9, 1, 5, 6};

for (int i = 0; i < arr.length - 1; i++) {

int minIndex = i;

for (int j = i + 1; j < arr.length; j++) {

if (arr[j] < arr[minIndex]) {

minIndex = j;

}

}

int temp = arr[minIndex];

arr[minIndex] = arr[i];

arr[i] = temp;

}

for (int num : arr) {

System.out.print(num + " ");

}

}

}

#### 8 Insertion Sort

public class InsertionSort {

public static void main(String[] args) {

int[] arr = {5, 2, 9, 1, 5, 6};

for (int i = 1; i < arr.length; i++) {

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j--;

}

arr[j + 1] = key;

}

for (int num : arr) {

System.out.print(num + " ");

}

}

}

### 7. Merging Two Arrays

public class MergeArrays {

public static void main(String[] args) {

int[] arr1 = {1, 3, 5};

int[] arr2 = {2, 4, 6};

int[] merged = new int[arr1.length + arr2.length];

int i = 0, j = 0, k = 0;

while (i < arr1.length && j < arr2.length) {

if (arr1[i] <= arr2[j]) {

merged[k++] = arr1[i++];

} else {

merged[k++] = arr2[j++];

}

}

while (i < arr1.length) {

merged[k++] = arr1[i++];

}

while (j < arr2.length) {

merged[k++] = arr2[j++];

}

for (int num : merged) {

System.out.print(num + " ");

}

}

}

### 8. Removing Duplicates from an Array

import java.util.Arrays;

public class RemoveDuplicates {

public static void main(String[] args) {

int[] arr = {1, 2, 2, 3, 4, 4, 5};

Arrays.sort(arr); // Sorting the array

int[] temp = new int[arr.length];

int j = 0;

for (int i = 0; i < arr.length - 1; i++) {

if (arr[i] != arr[i + 1]) {

temp[j++] = arr[i];

}

}

temp[j++] = arr[arr.length - 1];

int[] result = new int[j];

for (int i = 0; i < j; i++) {

result[i] = temp[i];

}

for (int num : result) {

System.out.print(num + " ");

}

}

}

### 9. Rotating an Array

public class RotateArray {

public static void main(String[] args) {

int[] arr = {1, 2, 3, 4, 5};

int k = 2; // Number of positions to rotate

k = k % arr.length; // In case k > length of array

reverse(arr, 0, arr.length - 1);

reverse(arr, 0, k - 1);

reverse(arr, k, arr.length - 1);

**Proper Oops concept**

### Step 1: Define the Base Class (Encapsulation and Abstraction)

**// Animal.java**

**public abstract class Animal {**

**private String name;**

**private int age;**

**public Animal(String name, int age) {**

**this.name = name;**

**this.age = age;**

**}**

**// Getter and Setter methods for name and age**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public int getAge() {**

**return age;**

**}**

**public void setAge(int age) {**

**this.age = age;**

**}**

**// Abstract method to be implemented by subclasses**

**public abstract void makeSound();**

**// Common method for all animals**

**public void eat() {**

**System.out.println(name + " is eating.");**

**}**

**}**

### Step 2: Define Subclasses (Inheritance)

**// Dog.java**

**public class Dog extends Animal {**

**public Dog(String name, int age) {**

**super(name, age);**

**}**

**@Override**

**public void makeSound() {**

**System.out.println(getName() + " says: Woof!");**

**}**

**public void fetch() {**

**System.out.println(getName() + " is fetching the ball.");**

**}**

**}**

**// Cat.java**

**public class Cat extends Animal {**

**public Cat(String name, int age) {**

**super(name, age);**

**}**

**@Override**

**public void makeSound() {**

**System.out.println(getName() + " says: Meow!");**

**}**

**public void scratch() {**

**System.out.println(getName() + " is scratching the furniture.");**

**}**

**}**

### Step 3: Demonstrate Polymorphism

**// Main.java**

**public class Main {**

**public static void main(String[] args) {**

**Animal dog = new Dog("Buddy", 3);**

**Animal cat = new Cat("Whiskers", 2);**

**// Using polymorphism to call the makeSound method**

**dog.makeSound();**

**cat.makeSound();**

**// Calling common method from the base class**

**dog.eat();**

**cat.eat();**

**// Typecasting to access subclass-specific methods**

**if (dog instanceof Dog) {**

**((Dog) dog).fetch();**

**}**

**if (cat instanceof Cat) {**

**((Cat) cat).scratch();**

**}**

**}**

**}**

### Step 4: Compile and Run the Program

**javac Animal.java**

**javac Dog.java**

**javac Cat.java**

**javac Main.java**

**java Main**

**Output:-**

**Buddy says: Woof!**

**Whiskers says: Meow!**

**Buddy is eating.**

**Whiskers is eating.**

**Buddy is fetching the ball.**

**Whiskers is scratching the furniture.**

**Switch case program**

**public class SwitchCaseExample {**

**public static void main(String[] args) {**

**int day = 3;**

**String dayName;**

**switch (day) {**

**case 1:**

**dayName = "Sunday";**

**break;**

**case 2:**

**dayName = "Monday";**

**break;**

**case 3:**

**dayName = "Tuesday";**

**break;**

**case 4:**

**dayName = "Wednesday";**

**break;**

**case 5:**

**dayName = "Thursday";**

**break;**

**case 6:**

**dayName = "Friday";**

**break;**

**case 7:**

**dayName = "Saturday";**

**break;**

**default:**

**dayName = "Invalid day";**

**break;**

**}**

**System.out.println("The day is: " + dayName);**

**}**

**}**