1.write a java code for sorting an array, and find number of occurrences of a particular number in java

Source code:

import java.util.Arrays;

public class SortArray {

public static void main(String[] args) {

// Example array

int[] array = {5, 2, 8, 3, 1, 4};

// Sorting the array

Arrays.sort(array);

// Printing the sorted array

System.out.println("Sorted Array: " + Arrays.toString(array));

}

}

public class CountOccurrences {

public static void main(String[] args) {

// Example array

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

// The number to find occurrences of

int numberToFind = 2;

// Method call to find the occurrences

int occurrences = findOccurrences(array, numberToFind);

// Printing the result

System.out.println("Number of occurrences of " + numberToFind + ": " + occurrences);

}

public static int findOccurrences(int[] array, int numberToFind) {

int count = 0;

for (int num : array) {

if (num == numberToFind) {

count++;

}

}

return count;

}

}

2)write a java code to find the maximum element in array.

Source code:

public class FindMaxElement {

public static void main(String[] args) {

// Example array

int[] array = {7, 3, 6, 2, 5};

// Finding the maximum element

int maxElement = findMax(array);

// Printing the maximum element

System.out.println("Maximum Element: " + maxElement);

}

public static int findMax(int[] array) {

// Assume the first element is the max to start

int max = array[0];

// Iterate through the array to find the maximum element

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

if (array[i] > max) {

max = array[i];

}

}

return max;

}

}

Polymorphism

1)Create a base class Animal with a method makeSound(). Create two derived classes Dog and Cat that

override the makeSound() method. Illustrate runtime polymorphism by creating objects of both derived

classes and invoking the method.

Source code:

public class Animal {

// Method to be overridden by derived classes

public void makeSound() {

System.out.println("Animal makes a sound");

}

}

public class Dog extends Animal {

// Overriding the makeSound method

@Override

public void makeSound() {

System.out.println("Dog barks");

}

}

public class Cat extends Animal {

// Overriding the makeSound method

@Override

public void makeSound() {

System.out.println("Cat meows");

}

}

public class Main {

public static void main(String[] args) {

// Creating objects of the derived classes

Animal myDog = new Dog();

Animal myCat = new Cat();

// Runtime polymorphism in action

// The makeSound method of the object's actual class is called

myDog.makeSound(); // Outputs: Dog barks

myCat.makeSound(); // Outputs: Cat meows

}

}

2)Create a Java class MathOperations with overloaded methods for addition (add) that can accept two

integers, three doubles, and two strings, respectively. Demonstrate the usage of these methods in the

main method by adding two integers, three doubles, and concatenating two strings.

Source code:

public class MathOperations {

// Overloaded method for adding two integers

public int add(int a, int b) {

return a + b;

}

// Overloaded method for adding three doubles

public double add(double a, double b, double c) {

return a + b + c;

}

// Overloaded method for concatenating two strings

public String add(String a, String b) {

return a + b;

}

public static void main(String[] args) {

// Creating an instance of MathOperations

MathOperations operations = new MathOperations();

// Adding two integers

int sumInt = operations.add(5, 10);

System.out.println("Sum of two integers: " + sumInt);

// Adding three doubles

double sumDouble = operations.add(1.5, 2.5, 3.5);

System.out.println("Sum of three doubles: " + sumDouble);

// Concatenating two strings

String concatenatedString = operations.add("Hello, ", "World!");

System.out.println("Concatenated string: " + concatenatedString);

}

}