|  |
| --- |
| Write a JAVA program to display default value of all primitive data type of JAVA |
| public class DefaultValues {  // Declare fields for each primitive data type  byte defaultByte;  short defaultShort;  int defaultInt;  long defaultLong;  float defaultFloat;  double defaultDouble;  char defaultChar;  boolean defaultBoolean;  public static void main(String[] args) {  // Create an instance of the DefaultValues class  DefaultValues defaults = new DefaultValues();  // Print the default values of each field  System.out.println("Default byte: " + defaults.defaultByte);  System.out.println("Default short: " + defaults.defaultShort);  System.out.println("Default int: " + defaults.defaultInt);  System.out.println("Default long: " + defaults.defaultLong);  System.out.println("Default float: " + defaults.defaultFloat);  System.out.println("Default double: " + defaults.defaultDouble);  System.out.println("Default char: '" + defaults.defaultChar + "'");  System.out.println("Default boolean: " + defaults.defaultBoolean);  }  } |
| Output:  Default byte: 0  Default short: 0  Default int: 0  Default long: 0  Default float: 0.0  Default double: 0.0  Default char: ' |

Program1:

Program 2:

|  |
| --- |
| Write a java program that display the roots of a quadratic equation ax2+bx+c=0. Calculate the  discriminate D and basing on value of D, describe the nature of root. |
| import java.util.Scanner;  public class QuadraticEquation {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  // Input coefficients  System.out.println("Enter coefficient a: ");  double a = sc.nextDouble();  System.out.println("Enter coefficient b: ");  double b = sc.nextDouble();  System.out.println("Enter coefficient c: ");  double c = sc.nextDouble();  // Calculate the discriminant  double D = b \* b - 4 \* a \* c;  System.out.println("The discriminant (D) is: " + D);  // Determine the nature of the roots  if (D > 0) {  // Two distinct real roots  double root1 = (-b + Math.sqrt(D)) / (2 \* a);  double root2 = (-b - Math.sqrt(D)) / (2 \* a);  System.out.println("The equation has two distinct real roots:");  System.out.println("Root 1: " + root1);  System.out.println("Root 2: " + root2);  } else if (D == 0) {  // One real root (double root)  double root = -b / (2 \* a);  System.out.println("The equation has twp equal real roots: " + root);  } else {  // Complex roots  double realPart = -b / (2 \* a);  double imaginaryPart = Math.sqrt(-D) / (2 \* a);  System.out.println("The equation has complex roots:");  System.out.println("Root 1: " + realPart + " + " + imaginaryPart + "i");  System.out.println("Root 2: " + realPart + " - " + imaginaryPart + "i");  }  sc.close();  }  } |
| Output1:  Enter coefficient a:  1  Enter coefficient b:  -5  Enter coefficient c:  6  The discriminant (D) is: 1.0  The equation has two distinct real roots:  Root 1: 3.0  Root 2: 2.0 |
| Output2:  Enter coefficient a:  1  Enter coefficient b:  -4  Enter coefficient c:  4  The discriminant (D) is: 0.0  The equation has twp equal real roots: 2.0 |
| Output3:  Enter coefficient a:  3  Enter coefficient b:  5  Enter coefficient c:  6  The discriminant (D) is: -47.0  The equation has complex roots:  Root 1: -0.8333333333333334 + 1.1426091000668406i  Root 2: -0.8333333333333334 - 1.1426091000668406i |

Program 3:

|  |
| --- |
| import java.util.Scanner;  public class BinarySearch {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  // Input the size of the array  System.out.println("Enter the number of elements: ");  int n = sc.nextInt();  // Input the elements of the array  int[] array = new int[n];  System.out.println("Enter the elements (sorted): ");  for (int i = 0; i < n; i++) {  array[i] = sc.nextInt();  }  // Input the element to be searched  System.out.println("Enter the element to search: ");  int key = sc.nextInt();  // Perform binary search  int result = binarySearch(array, key);  // Display the result  if (result == -1) {  System.out.println("Element not found in the array.");  } else {  System.out.println("Element found at index: " + result);  }  sc.close();  }  // Method to perform binary search  public static int binarySearch(int[] array, int key) {  int left = 0;  int right = array.length - 1;  while (left <= right) {  int mid = left + (right - left) / 2;  // Check if key is present at mid  if (array[mid] == key) {  return mid;  }  // If key is greater, ignore the left half  if (array[mid] < key) {  left = mid + 1;  }  // If key is smaller, ignore the right half  else {  right = mid - 1;  }  }  // Key not found  return -1;  }  } |
| Output1:  Enter the number of elements:  6  Enter the elements (sorted):  20 30 40 50 60 70  Enter the element to search:  50  Element found at index: 3 |
| Output2:  Enter the number of elements:  5  Enter the elements (sorted):  -1 0 2 4 8  Enter the element to search:  10  Element not found in the array. |

Program4:

|  |
| --- |
| import java.util.Scanner;  public class BubbleSort {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  // Input the size of the array  System.out.println("Enter the number of elements: ");  int n = sc.nextInt();  // Input the elements of the array  int[] array = new int[n];  System.out.println("Enter the elements: ");  for (int i = 0; i < n; i++) {  array[i] = sc.nextInt();  }  // Perform bubble sort  bubbleSort(array);  // Display the sorted array  System.out.println("Sorted array: ");  for (int i : array) {  System.out.print(i + " ");  }  }  // Method to perform bubble sort  public static void bubbleSort(int[] array) {  int n = array.length;  for (int i = 0; i < n - 1; i++) {  for (int j = 0; j < n - 1 - i; j++) {  if (array[j] > array[j + 1]) {  // Swap array[j] and array[j + 1]  int temp = array[j];  array[j] = array[j + 1];  array[j + 1] = temp;  }  }  }  }  } |
| Output:  Enter the number of elements:  5  Enter the elements:  1 -5 4 2 89  Sorted array:  -5 1 2 4 89 |