|  |  |
| --- | --- |
| Ex.No.**1**  **01.10.2022** | **Insertion Sort** |

|  |
| --- |
| **AIM:** |

To write and execute Java program to perform Insertion Sort.

|  |
| --- |
| **Pseudocode:** |

**1 Insertion-sort(A)**

**2 For j 🡨 2 to n**

**3 do key 🡨 A[j]**

**4 Insert A[j] into the sorted sequence A[1….j -1]**

**5 i 🡨j-1**

**6 While i>0 and A[i] > key**

**7 do A[i+1] 🡨 A[i]**

**8 i 🡨 i-1**

**9 A[i+1] 🡨 key**

**Insertion sort – sorts the element in place**

|  |
| --- |
| **Explanation:** |

**In line number 1 :-**There is a Linear array A.

**In line No. 2:-** There is a for loop Starting from 2 to n.

**In line No. 3:-** put the value of A of j into key.

**In line no. 4:-** Insert into array A of j into the sorted sequence A of 1, A of 2 and so on till A of

Minus one.

**In line no. 5:-** I become j minus one.

**In line no. 6:-** There is a while loop and the condition is I greater than zero and there is and

Operator followed by A of I value is greater than key value.

**In line no. 7:-** in line no. 5 if both the condition will true then the compiler goes to line no 7

And do A of I plus 1 element to A of i.

**In line no.8 :-** and I become I minus 1.

**In line no. 9:-** when while loop condition fail then compiler goes to line no. 9 and put the\

Value of key into A of I plus 1.

|  |
| --- |
| **Example:** |

We take an unsorted array for our example.

Unsorted Array

Insertion sort compares the first two elements.

Insertion Sort

It finds that both 14 and 33 are already in ascending order. For now, 14 is in sorted sub-list.

Insertion Sort

Insertion sort moves ahead and compares 33 with 27.

Insertion Sort

And finds that 33 is not in the correct position.

Insertion Sort

It swaps 33 with 27. It also checks with all the elements of sorted sub-list. Here we see that the sorted sub-list has only one element 14, and 27 is greater than 14. Hence, the sorted sub-list remains sorted after swapping.

Insertion Sort

By now we have 14 and 27 in the sorted sub-list. Next, it compares 33 with 10.

Insertion Sort

These values are not in a sorted order.

Insertion Sort

So we swap them.

Insertion Sort

However, swapping makes 27 and 10 unsorted.

Insertion Sort

Hence, we swap them too.

Insertion Sort

Again we find 14 and 10 in an unsorted order.

Insertion Sort

We swap them again. By the end of third iteration, we have a sorted sub-list of 4 items.

Insertion Sort

This process goes on until all the unsorted values are covered in a sorted sub-list.

|  |
| --- |
| **Program Code:** |

import java.util.Scanner;

class Insertion {

    private static int size=5;

    private int[] a=new int[size];

    public void read() {

        Scanner sc=new Scanner(System.in);

        for(int i=0;i<size;i++) {

            System.out.print("Enter Elements of an Array :-");

            a[i]=sc.nextInt();

        }

    }

    public void shorts() {

        int j=0;

        int temp=0;

        for(int i=1;i<size;i++) {

            temp=a[i];

            j=i-1;

            while(j>=0 && a[j]>temp) {

                a[j+1]=a[j];

                j--;

            }

            a[++j]=temp;

        }

    }

    public void print() {

        for(int i=0;i<size;i++) {

            System.out.println(a[i]);

        }

    }

}

public class InsertionShort {

    public static void main(String[] args) {

        Insertion i=new Insertion();

        i.read();

        i.shorts();

        i.print();

    }

}

|  |
| --- |
| **Output Screenshots:** |



|  |
| --- |
| **RESULT:** |

Thus, the programs for the given problem statements has been executed and the results are verified successfully.