Code:

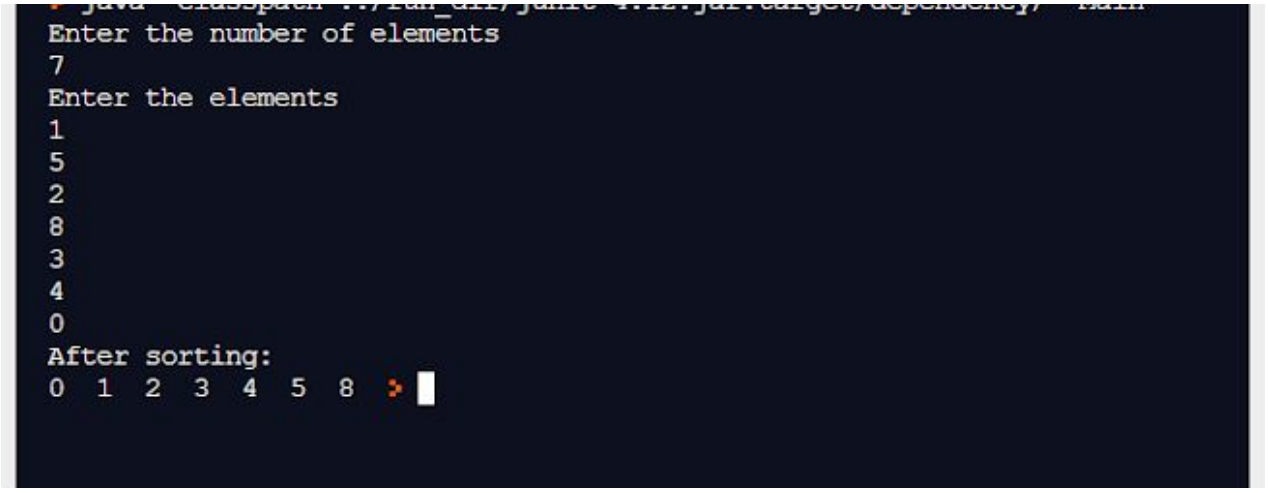
```
import java.util.*;
public class Main
{
   public static void insertionSort(int arr[],int size)
   {
      int value,index;

      for (int i = 1; i < size; i++)
      {
         value = arr[i];
         index = i;

         while (index > 0 && arr[index-1] > value)
         {
            arr[index] = arr[index-1];
            index--;
         }
         arr[index] = value;
      }
   }
         public static void main(String[] args)
         {
                  Scanner s=new Scanner(System.in);
                  System.out.println("Enter the number of elements");
                  int size=s.nextInt();
                  int[] arr=new int[size];
                  System.out.println("Enter the elements");
                  for(int i=0;i<size;i++)
```

```
            {
               arr[i]=s.nextInt();
            }
            insertionSort(arr,size);
            System.out.println("After sorting:");
            for(int i=0;i<size;i++)
            {
               System.out.print(arr[i]+"  ");
            }
        }
}
```

Output:



Time and space complexity:

```
insertionSort (int arr[], int size)
{
    int value, index
    for (int i = 1; i < size; i++)        n-1
    {
        value = arr[i];
        index = i;
        while (index > 0 && arr[index-1] > value)
        {
            arr[index] = arr[index-1];
            index--;
        }
        arr[index] = value;
    }
}
```

Time Complexity:

1) Best case:
   If size of array is n then if it is already sorted it will require only 1 pass
   ∴ time complexity = O(n)

2) Worst case:
   If the array is arranged in descending order initially Total no of comparisions → $(n-1)+(n-2)+\cdots+3+2+1$
   $= \frac{(n-1)n}{2} = \frac{n^2-n}{2} \approx O(n^2)$

3) Space complexity: arr[] → n words
   i, size, index, value → 1 word each
   ⇒ $n+4 \approx O(n)$