Pasos para realizar mergesort:
MergeSort(arr[], l, r)
if r > 1

1. Encuentra el punto medio para dividir el array en dos mitades:
   middle m = (1+r) / 2
2. Llama a mergeSort para la primera mitad:
   Llama a mergeSort(arr, l, m)
3. Llama a mergeSort para la segunda mitad:
   Llama a mergeSort(arr, m+1, r)
4. Une las dos mitades ordenadas en el paso 2 y 3:
   Llama a merge(arr, l, m, r).

Implementación en Java.
Para empezar, creamos una clase la cual contendrá 3 métodos, una que se encargará de dividir en dos mitades el vector recursivamente, el segundo se encargará de unir los sub-vectores y el tercero es un método de apoyo que nos imprimirá el vector ordenado. Entonces, así quedaría:

```
public class MergeSort{
    public void sort() {

    }

    public void merge() {

    }

    public void printArray() {

    }
}
```

Empecemos con el método sort(), ésta función tiene la tarea de dividir en dos mitades el vector que se pasa por parametro hasta que el vector se quede de tamaño 1.

```
public void sort(int arr[], int left, int right){
    if(left < right){
      //Encuentra el punto medio del vector.
      int middle = (left + right) / 2;

      //Divide la primera y segunda mitad (llamada recursiva).
      sort(arr, left, middle);
      sort(arr, midddle+1, right);

      //Une las mitades.
      merge(arr, left, middle, right);
    }
}
```

El siguiente método a realizar es el método merge() el cuál recibirá por parámetro el vector, la posición de la izquierda, la posición del medio y la posición final(right).

```
public void merge(int arr[], int left, int middle, int right) {
  //Encuentra el tamaño de los sub-vectores para unirlos.
  int n1 = middle - left + 1;
  int n2 = right - middle;

  //Vectores temporales.
  int leftArray[] = new int [n1];
  int rightArray[] = new int [n2];

  //Copia los datos a los arrays temporales.
  for (int i=0; i < n1; i++) {
    leftArray[i] = arr[left+i];
  }
  for (int j=0; i < n2; j++) {
    rightArray[j] = arr[middle + j + 1];
  }
  /* Une los vectorestemporales. */

  //Índices inicial del primer y segundo sub-vector.
  int i = 0, j = 0;

  //Índice inicial del sub-vector arr[].
  int k = left;

  //Ordenamiento.
  while (i < n1 && j < n2) {
    if (leftArray[i] <= rightArray[j]) {
      arr[k] = leftArray[i];
      i++;
    } else {
        arr[k] = rightArray[j];
        j++;
    }
    k++;
  }//Fin del while.

  /* Si quedan elementos por ordenar */
  //Copiar los elementos restantes de leftArray[].
  while (i < n1) {
    arr[k] = leftArray[i];
    i++;
    k++;
  }
  //Copiar los elementos restantes de rightArray[].
  while (j < n2) {
    arr[k] = rightArray[j];
    j++;
    k++;
  }
}
```