```java
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Mearge {
    public static void main(String[] args) {
        // Scanner for reading input from the user
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter integers (type 'done' to
finish):");

        // List to hold records read from user input
        List<Integer> records = new ArrayList<>();

        // Read records from the user
        while (scanner.hasNext()) {
            String input = scanner.next();
            if (input.equalsIgnoreCase("done")) {
                break;
            }

            try {
                // Parse the input as an integer and add to
the list
                int record = Integer.parseInt(input);
                records.add(record);
            } catch (NumberFormatException e) {
                System.out.println("Invalid input. Please
enter a valid integer or type 'done' to finish.");
            }
        }

        // Perform merge sort on the list of records
        mergeSort(records, 0, records.size() - 1);

        // Output the sorted records
        System.out.println("Sorted integers:");
        for (Integer record : records) {
```

```
            System.out.println(record);
        }
    }

    // Merge sort function
    private static void mergeSort(List<Integer> records, int
left, int right) {
        if (left < right) {
            int mid = (left + right) / 2;

            // Recursively sort the left and right halves
            mergeSort(records, left, mid);
            mergeSort(records, mid + 1, right);

            // Merge the two sorted halves
            merge(records, left, mid, right);
        }
    }

    // Function to merge two sorted halves
    private static void merge(List<Integer> records, int
left, int mid, int right) {
        // Create temporary arrays for the two halves
        List<Integer> leftArray = new
ArrayList<>(records.subList(left, mid + 1));
        List<Integer> rightArray = new
ArrayList<>(records.subList(mid + 1, right + 1));

        int leftIndex = 0, rightIndex = 0;
        int mergeIndex = left;

        // Merge the left and right halves in sorted order
        while (leftIndex < leftArray.size() && rightIndex <
rightArray.size()) {
            if (leftArray.get(leftIndex) <=
rightArray.get(rightIndex)) {
                records.set(mergeIndex,
leftArray.get(leftIndex));
```

```
                leftIndex++;
            } else {
                records.set(mergeIndex,
rightArray.get(rightIndex));
                rightIndex++;
            }
            mergeIndex++;
        }

        // Copy any remaining elements from the left half
        while (leftIndex < leftArray.size()) {
            records.set(mergeIndex,
leftArray.get(leftIndex));
            leftIndex++;
            mergeIndex++;
        }

        // Copy any remaining elements from the right half
        while (rightIndex < rightArray.size()) {
            records.set(mergeIndex,
rightArray.get(rightIndex));
            rightIndex++;
            mergeIndex++;
        }
    }
}
```