CS 2050
Computer Science II

Thyago Mota

METROPOLITAN STATE UNIVERSITY OF DENVER
LIVES TRANSFORMED

Agenda

- Sorting Objects:
  - Comparable<T> Interface (review)
  - Merge Sort w/ Linked List

```java
public class Student {

    private int    id;
    private String name;

    public Student(int id, String name) {
        this.id = id;
        this.name = name;
    }

    public int getId() { return id; }

    public String getName() { return name; }

    @Override
    public String toString() {
        return "(" + id + ", '" + name + "')";
    }
}
```

```java
public class Student implements Comparable<Student> {

    private int    id;
    private String name;

    public Student(int id, String name) {
        this.id = id;
        this.name = name;
    }

    public int getId() { return id; }

    public String getName() { return name; }

    @Override
    public String toString() {
        return "(" + id + ", '" + name + "')";
    }

    @Override
    public int compareTo(Student other) {
        return name.compareTo(other.name);
    }
}
```

In our “Java Guide - Interfaces Section” there Is an explanation about the **`Comparable<T>`** interface. Please read it!

```java
public class SortingObjectsSol {

    private static final String NAMES_FILENAME = "names.txt";

    public static void main(String[] args) throws FileNotFoundException {

        LinkedList<Student> students = new LinkedList<>();
        Scanner in = new Scanner(new FileInputStream(NAMES_FILENAME));
        int id = 1;
        while (in.hasNextLine()) {
            String name = in.nextLine();
            Student student = new Student(id, name);
            students.append(student);
            id++;
        }
        in.close();
        System.out.println(students);
    }
}
```

```
(1, 'Florentina') (2, 'Omar') (3, 'Brittaney') (4, 'Celestine') (5, 'Joelle') (6, 'Dulcie') (7,
'Magaret') (8, 'Sheldon') (9, 'Mercedes') (10, 'Jaleesa') (11, 'Chelsey') (12, 'Toi') (13,
'Janay') (14, 'Roxana') (15, 'Marge') (16, 'Mitsue') (17, 'Ranae') (18, 'Damion') (19, 'Maranda')
(20, 'Elyse')
```

```java
// TODOd: merge the two given lists returning a new list with the elements sorted
public static LinkedList<Student> merge(LinkedList<Student> left, LinkedList<Student> right) {
    int i = 0, j = 0;
    LinkedList<Student> sorted = new LinkedList<>();
    while (i < left.size() && j < right.size()) {
        Student leftStudent  = left.get(i);
        Student rightStudent = right.get(j);
        if (leftStudent.compareTo(rightStudent) < 0) {
            sorted.append(leftStudent);
            i++;
        }
        else {
            sorted.append(rightStudent);
            j++;
        }
    }
    while (i < left.size()) {
        Student leftStudent  = left.get(i++);
        sorted.append(leftStudent);
    }
    while (j < right.size()) {
        Student rightStudent  = right.get(j++);
        sorted.append(rightStudent);
    }
    left.clear();
    right.clear();
    return sorted;
}
```

```java
15      // TODOd: merge the two given lists returning a new list with the elements sorted
16  @   public static LinkedList<Student> merge(LinkedList<Student> left, LinkedList<Student> right) {
17          int i = 0, j = 0;
18          LinkedList<Student> sorted = new LinkedList<>();
19          while (i < left.size() && j < right.size()) {
20              Student leftStudent  = left.get(i);
21              Student rightStudent = right.get(j);
22              if (leftStudent.compareTo(rightStudent) < 0) {
23                  sorted.append(leftStudent);
24                  i++;
25              }
26              else {
27                  sorted.append(rightStudent);
28                  j++;
29              }
30          }
31          while (i < left.size()) {
32              Student leftStudent  = left.get(i++);
33              sorted.append(leftStudent);
34          }
35          while (j < right.size()) {
36              Student rightStudent  = right.get(j++);
37              sorted.append(rightStudent);
38          }
39          left.clear();
40          right.clear();
41          return sorted;
42      }
```

```java
// TODOd: implement merge sort using a linked list
public static LinkedList<Student> mergeSort(LinkedList<Student> students) {

    // TODOd: implement the base case
    if (students.size() <= 1)
        return students;

    // TODOd: divide the students list into two lists: left and right
    int middle = students.size() / 2;
    LinkedList<Student> left = new LinkedList<>();
    for (int i = 0; i < middle; i++)
        left.append(students.get(i));
    LinkedList<Student> right = new LinkedList<>();
    for (int i = middle; i < students.size(); i++)
        right.append(students.get(i));

    // TODOd: recursively call mergeSort w/ left and right linked lists
    left  = mergeSort(left);
    right = mergeSort(right);

    // TODOd: return the result of merging left and right linked lists
    return merge(left, right);
}
```

```
70          LinkedList<Student> students = new LinkedList<>();
71          Scanner in = new Scanner(new FileInputStream(NAMES_FILENAME));
72          int id = 1;
73          while (in.hasNextLine()) {
74              String name = in.nextLine();
75              Student student = new Student(id, name);
76              students.append(student);
77              id++;
78          }
79          in.close();
80          System.out.println(students);
81          LinkedList<Student> studentsSorted = mergeSort(students);
82          System.out.println(studentsSorted);
83      }
```

```
(3, 'Brittaney') (4, 'Celestine') (11, 'Chelsey') (18, 'Damion') (6, 'Dulcie') (20, 'Elyse') (1,
'Florentina') (10, 'Jaleesa') (13, 'Janay') (7, 'Magaret') (15, 'Marge') (16, 'Mitsue') (2,
'Omar') (5, 'Joelle') (17, 'Ranae') (19, 'Maranda') (8, 'Sheldon') (9, 'Mercedes') (12, 'Toi')
(14, 'Roxana')
```