**OUT OF LAB 3 TLDDR**

**Group members:** **Tasks:**

Delzer, Ben Major Event Update Subsection

Dusek, Tim Recursion Section

Hansen, Christian Comparator Implementation Subsection

Ilunga, Moses Manual Sorting

Major Event Update Section -- Delzer

Event

The revision of Event wasn’t too difficult after Scott helped me understand how to interpret how many heats there will be in each Event. After that I used the length of the array in the new constructor to determine the number of heats and put almost all of runEvent() into a do while loop. This allowed it to run once even if there is only one run to keep those Events valid. Then the while looks at the array length to determine how many times this would be repeated. This also meant I had to make score additive. I did this by making set score into a counter using +=. How I fixed the issue of resetting the score after every event was by setting setScore() to -getScore(). This set score back to zero so it is ready for the next event.

The Event compareTo was straight forward. I just took another compareTo that I already made that had other variables set aside for comparison to output a positive, negative, or 0 for equal and renamed the variables accordingly to reflect the population of each Event.

Fan

For the compareTo for fan I used the same skeleton I used for Event. However deciding how to calculate happiness took some time. It wasn’t difficult just finding the right placements for several counters took a little work. For events attended I just put a counter inside buyTicket to keep track of how many events the fan has entered. Every time they were refunded it added to a counter for how many refunded. And for rejected I had to make a method to add to the counter because the situation only happened outside of fan and during an out of money exception or too many fans exception. After the Olympics I would calculate the happiness with the given equation and presto.

Driver

Printing the average happiness was easy. Running an advanced for loop through every fan and picking up their happiness divided by how many fans got the average. I want to say anything from 250 to 500 would be a success from what I witnessed came to be during testing. For the top 10s I used the method Scott wrote for finding the max Athletes and duplicated and reworded them to work for finding all the remaining info for Driver.

AI

For the smart AI for Athletes I didn’t have to change much. During calculation of stamina I used a condition to be decided if the adjusted stamina dropped below 0 during an Event procedure. If stamina were to drop to or below 0 a random number between 1 and 10 would be generated. The same scale used for skill. If rnd < althlete skill level, then the Athlete would not lose stamina and would not participate in the event. I also made if for when Athletes needed to be added back into the lineup for the next heat the event would take from this stand by list first before taking from fainted.

Recursion Section -- Dusek

I worked on the recursion methods at the end of the driver for my portion of this project. I started by taking the Athlete list (ath) and going through the last piece of the list and checking to see if it met certain criteria. If it met that criteria it was added to another list which would be printed after the original athlete list hit 0 in length. After it went through the end of the list, it would remove the last athlete and call itself to repeat all over again. When the list hit 0 in length it would trigger an if statement and print all of the lists out.

Comparator Implementation Subsection -- Hansen

For my section of this project, I was in charge of implementing comparable and comparator, as well as writing the Javadoc for each file.

Starting with comparable, I updated the compareTo methods in Event, Venue, and Athlete in order to sort the arrays of Events, Venues, and Athletes that were contained in the Driver. I added the lists of these objects into temporary lists so the main lists would maintain their order. I wasn’t 100% sure if the order mattered here, so I played it safe just in case the order did matter. After this, I updated the compareTo method in Fan to compare the Fans by their happiness ratings. After the events had been run, I again created temporary lists for fans and sorted them to print the 10 happiest, then reversed the sort to print the 10 unhappiest fans.

When implementing comparators for athlete, I made one for gold medals, silver medals, bronze medals, and total medals. I then used the tiebreakers in the order from the spec as else statements, continuing until a comparison was found. Using else statements was what made the most sense to me, because if the current comparison would result in a tie (if the if and else if statements failed), it would move to the next step in the comparison (the else was left, allowing for another form of comparison).

I think everything in my part of this was fairly straightforward. I know there is probably another way to implement the tiebreakers in the comparators, but I did what made the most sense to me.

Manual Sorting -- Ilunga

I was in charge of overseeing the project, putting all the files together, and making sure all the deadlines were met as well as doing the manual sorting. I want to start my section by saying that it was a great experience for me to work with such a great group. Everyone co-operated and did what they were assigned to do, so this project means a lot to me in terms of what we accomplished through teamwork. Christian had a big part in helping me debug the files and making sure all the code was done logically at least.

I had to do 2 different types of sorting methods that were discussed in class manually. The first sorting that had to be done was sorting athletes by their endorsements. I chose to use selection sorting for this one because it seemed the most logical way of doing it for me.

I ended up writing a selectionSort method in the driver (as I was not too sure which class to write the method in) that returns a sorted arraylist of athletes by endorsements. I ended up using a nested for loop to go through the arraylist and organize the athletes based on their endorsements. When I am printing the results of my sorting, I made sure I reversed the sorted list and printed the top 10 (0 - 9) from the sorted, reversed-arraylist.

My next sorting section was for events. Events had to be sorted based upon their attendance (or FanCount() ). I wrote a BubbleSort method for this because I seem to understand it better right after selectionsorting. The method doesn’t return anything but takes in an arraylist of events. Again, I used a nested for loop to do most the sorting with a nested if statement to do the swapping of numbers.

My section was very straightforward and was not as hard as the other sections. However, having to oversee the project made up for my workload as most of my time went to that. I not only gained some project managing experience, but I also gained a better understanding of sorting, selection and bubble sorting to be more specific.