Ανάλυση Κλάσεων Εργασίας

1. ActivityResults

- Implements Serializable interface. Δηλαδή τα instances μετατρέπονται σε byte και αποθηκευονται/ μεταδιδονται σε byte.
- Έχει 2 constructors για τις private μεταβλητες (totalDistance, totalAscent, totalTime, meanSpeed) ο πρώτος κανει initialize τις παραπάνω τιμές με 0, ενω ο δευτερος δέχεται καποιες παραμέτρους και τις τοποθετεί στις παραπάνω μεταβλητές.
- Επιπλεον έχει κάποιες μεθόδους που χρησιμοποίει για επιστροφή, αντιγραφή των παραπάνω μεταβλητών.

2. DummyUser

3. GlobalAlctivityStatsUpdater

4. GPXPacket

- Implements seriliazable, the class matches a seriliazable object containing CONNECTION_ID with a GPXPoint list, this will help as a key in the map and reduce functions.

5. GPXPoint

- Implements seriliazable, the class has a constructor with 4 parameters(latitude, longtitude, elevation, time) for a GPX point, also has 4 methods to return the GPX values.

6. Main

7. MapingThread

- The class overrides the run() method, that now calls the map(0 method containing the connection_id of the gpxPacket and GPXPoints that will now be used in the following methods.
- The class also has a calculateTotalDistance method that takes a list of gpxPoints and calculates the total distance in km, this is done by callint the calculateDistance method that calculates the diatance between two GPXPoitns.
- The class also has a calculateTotalAscent method that calculates the total ascents for each GPXPoint in a gpxPoins list.
- The class also has calculatetotalTimeInMinutes that calculate the total time.

8. ReadWriteLock

- The class's function is to provide a safe way to update the user_stats file.

9. ResultReducer

- Implements thread the class is responsible for receiving intermediate results from workers through a shared queue and reducing the results and sending them to the correct user.
- The run() method is overridden and now is responsible for reducing the intermediate results and forwarding them to the appropriate user.

10. StatLogger

11. StatPrinter

- The class provides methods for statistics to be printed.
- The prtinTotalStats takes a file object representing the total stats file and a string that represents the type of stats(user_total_stats or global_stats) and prints the results if the stat type is global_stats it also prints the last update date of the file.
- The printRouteStats takes a file as a parameter and prints its contects in a presentable way.

12. TaskAssigner

- Extends thread, the class has a constant variable chunkSize, three lists (1. sharedMessageQueue: used to communicate between threads, workerQueues: used to send GPX packets to worker threads, reduceThreadQueues: used to store info from reducethreads) and a lock used .
- Also has a constructor for the above variables.
- The class overrides the run() method, it creates a sharedResultsQueue so the global status updater thread can receive results from reduce threads, also creates gasu that is responsible to update global_activity-stats every x secs. Also has a while loop that checks the sharedMessageQueue for messages, if a message is found it is removes from the queue and a new list for intermediate results is created.. It also creates a ResultReducer thread That uses the method sendGPXChunks that is responsible for splitting the gpx file and sending it to the workers. It also reduces the intermediate results in final results.

- The sendGPXChunks method uses a for loop to parse a gpxFile and spilt in chunks each one containing 9 GPX points. The method will also return the number of chunks sent so the reduce thread knows when to reduce intermediate results to final results.

13. UserConnection

- Implements thread, the class has a constructor that is used to receive the data from UserHandler.
- The UserConection class overrides the run() method, that now receives a GPX file from the user, keeps the username and creates a userPacket object containing the connectionId and the GPX file.
- The class waits for the ActivityResults to be generated, and then using StatLogger the xml file containing the ActivityResuts is sent using the sockets output stream.

14. UserHandler

- Implements thread that handles user connections, the class overrides the run() method, that is called when the thread starts.
- The constructor takes three arguments : a shared queue shareMessageQueue, a HashMap that keeps track of the user connection and activity results, and a ReadWriteLock that helps in synchronization.
- The run method create a ServerSocket on port 6969 that listens for incoming connection requests, when a request is made then a new UserConnection thread starts.
- The activity results are stored in a shared HashMap that can also be accessed by TaskAssigner whose reduce threads send the results back to UserConnections.

15. UserPacket

- The class has a constructor that takes ConnectionId and the gpx file as parameters.

16. Worker

- The class receives GPX chunks from a server and processes them accordingly.
- The class's main method creates a Socket object and ObjectOutputStream and ObjectInputStream objects to communicate with the server.
- When a GPXPackets is received the worker extracts the info using the getGpxPoints() method.
- The worker then using MappingThread passes the GPxPacket to the bjectOutputStream.

17. WorkerConnection

- Extends the thread class, and the class's objective is to offer a connection between the main server and the worker nodes.
- The class has a constructor that takes 3 parameters (1. connectionSocket that represents the socket used for communication with the worker node, 2.

sharedWorkQueue that represents the queue of work that is shared among the workers, 3. reduceThreadQueue that is a hashMap that maps the worker Id to a queue of intermediate results, will be used in the reduce phase).

- The run() method is overridden and now contains the way that connection to the worker node is made, also it creates two threads for sending and receiving data.
- The first thread outputStreamThread checks the sharedWorkQueue for new work, if there is then it removes the first item and sends it to the worker node via the finalOos , if there are no items in the queue it waits until an item is added.
- The second thread listens for new packets for the worker node via the ois, when it receives a new packet it extavts the intermediate results and adds thme to reduceThreadQueues based on the worker id.

18. WorkerHandler

19. WorkerPacket

- The class has a constructor that takes ConnectionId and the intermediate results.