Swarm Server and Rover requirements

Use java version 1.8.0_77

Rover programs must be able to start with a batch (.bat) file running on a windows machine.

The Swarm Server will break the socket connection to a Rover Client that makes more than 500 requests to the server in the span of one second. Both Command and Request strings are counted independent of their use or responses. (Use a sleep function to pace your system).

The rover program must be started by executing a class named ROVER_XX. Any additional classes and packages can be included as needed. The ROVER_XX program will connect to the server on ‘localhost’ ‘port 9537’ as the default. The ROVER_XX program will accept an IP address as a runtime attribute. This will be the address that the rover program will connect to the server instead of localhost should the program have to be run over a network. It will still use port 9537 as the default.

A runnable jar file must also be provided and will have the file name ROVER_XX.jar (note character casing).

Example for running the program through the command line and specifying an IP address:

```bash
start java -jar Rover_98.jar 192.168.120.1
```

Map specifics:

Each location on the map can only have one type of Terrain.

Each location on the map can only have a single sample of science on it.

Each location on the map may include both a terrain feature and a sample of science.

Drive systems:

Wheels can move over terrain the fastest of any drive system.

Wheels can travel over Soil and Gravel.

Wheels cannot travel into or through Rock terrain

Wheels will immediately get stuck upon entering Sand terrain.

Walkers move much slower than Wheels and a little slower than Treads.

Walkers can travel over Soil, Gravel, and Rocks.

Walkers will immediately get stuck upon entering Sand terrain.

Treads move much slower than Wheels and a little faster than Walkers.

Treads can travel over Sand, Soil, and Gravel.
**Extraction tools:**

Drills can extract samples of science from Rock or Gravel terrain.

Excavators can extract samples of science from Soil or Sand terrain.

**Scanners tool:**

- **RADIATION_SENSOR** -> Radioactive science
- **CHEMICAL_SENSOR** -> Organic Science
- **SPECTRAL_SENSOR** -> Crystal Science
- **RADAR_SENSOR** -> mineral Science
- **RANGE_EXTENDER** -> Extends scanner range from a 7x7 square to an 11x11 square.

**Swarm Server Communications Protocols:**

Communications between the rover and the server is all text based.

- **Connection hand-shaking** are a series of sent and received Strings to establish a connection between the rover and the swarm server.

- **Commands** are one way. They are sent to the server but will generate no text reply back from the server.

- **Requests** are two directional. A request is sent to the server which responds with one or more lines of text. The responses can be simple text or can be objects encoded using json.

**Connection:**

After it is started the server will wait and listen on port 9537.

After the Rover connects to this port the Swarm Server will send it the string “SUBMITNAME”.

When the Rover receives the string “SUBMITNAME” it needs to respond by sending back to the server the pre-defined RoverName (Enum) String that corresponds to that particular Rover. This string will be of the format “ROVER_XX” where the XX is a numeric digit between the values 01 to 20. The leading zero is necessary for the first 1-9 names in the series.
Server Commands:

MOVE
This command will be a string with the format “MOVE #”
Where the # is an uppercase letter {N, S, E, W} representing the direction the rover is requesting to move.
If there are no obstructions on the map for that particular rover’s drive system, the rover location will be moved one map tile in the requested direction. If the tile contains an obstruction such as un-traversable terrain or another rover, then the location of the rover will remain the same as before the move request.
No response is returned by the server. To confirm the move was successful requires doing a request for the rover position and checking if it has changed.

GATHER
This command will be a string of the format “GATHER”
It will have no additional parameters and will generate no response from the server.
When this command is issued if the rover is positioned on a tile that contains a sample of science, and if the rover has the proper extraction tool for that particular tile terrain, then the science is removed from the map and placed in the rover’s cargo storage.

Server Requests:

LOC
The command sent to the server will consist of the string “LOC”
The server will respond to this request with a single line of string text of the format:
“LOC xxx yyy”
Specifically the string will start with the upper case letters LOC then a space then one or more numerical digits another space and an additional one or more digits.
The first set of digits will be the x coordinate of the tile the rover currently occupies.
The second set of digits will be the y coordinate of the tile the rover currently occupies.
The returned coordinate values may not have the same number of digits. The returned coordinate values may or may not have leading zeros. The format of the two sets of digits may be different from each other (number of digits, leading zeros, etc).
SCAN

The command sent to the server will consist of the string “SCAN”

The result returned by the server will consist of a series of text strings. The number of lines of text returned is variable.

The first line of text returned will be the string “SCAN”

The following lines will be a ScanMap object that has been converted to a string json format.

The last line of text returned will be the string “SCAN_END”.

The ScanMap object contains a 2D MapTile[][] array of the area surrounding the rover. Each MapTile in the array will contain attributes with Terrain features and whether the tile has another rover on it. The MapTile will contain an attribute field for Science samples – but it will only contains those types that the rover can sense based on the ScanningTools that it has equipped.

CARGO

The command sent to the server will consist of the string “CARGO”

The result returned by the server will consist of a series of text strings. The number of lines of text returned is variable.

The first line of text returned will be the string “CARGO”

The following lines will be an ArrayList<Science> object that has been converted to a string json format.

The last line of text returned will be the string “CARGO_END”.

When reconstructed the ArrayList will contain a listing of all the Science samples held in the rovers cargo hold.

TARGET_LOC

Note: This command only ever needs to be run once – the information it provides will never change.

The command sent to the server will consist of the string “TARGET_LOC”

The server will respond to this request with a single line of string text of the format:

“TARGET_LOC xxx yyy”

Specifically the string will start with the upper case letters TARGET_LOC then a space then one or more numerical digits another space and an additional one or more digits.

The first set of digits will be the x coordinate of the tile the rover currently occupies.

The second set of digits will be the y coordinate of the tile the rover currently occupies.

The returned coordinate values may not have the same number of digits. The returned coordinate values may or may not have leading zeros. The format of the two sets of digits may be different from each other (number of digits, leading zeros, etc).
This value represents the center coordinate of a box that is (currently) 7 x 7 tiles. This is an area that has been determined by the original planetary survey to contain a very high density of collected Science deposits contained in all types of terrain.

**START_LOC**

Note: This command only ever needs to be run once – the information it provides will never change.

The command sent to the server will consist of the string “START_LOC”

The server will respond to this request with a single line of string text of the format:

“START_LOC xxx yyy”

Specifically the string will start with the upper case letters START_LOC then a space then one or more numerical digits another space and an additional one or more digits.

The first set of digits will be the x coordinate of the tile the rover currently occupies.

The second set of digits will be the y coordinate of the tile the rover currently occupies.

The returned coordinate values may not have the same number of digits. The returned coordinate values may or may not have leading zeros. The format of the two sets of digits may be different from each other (number of digits, leading zeros, etc).

This value represents the center coordinate of a box that is (currently) 7 x 7 tiles. This is the area around which the Rovers are initially spawned into the map at. To officially have retrieved a Science sample the Rover carrying it must return to this area and park inside this box.

**EQUIPMENT**

Note: This command only ever needs to be run once – the information it provides will never change.

The command sent to the server will consist of the string “EQUIPMENT”

The result returned by the server will consist of a series of text strings. The number of lines of text returned is variable.

The first line of text returned will be the string “EQUIPMENT”

The following lines will be an ArrayList<String> object that has been converted to a string json format.

The last line of text returned will be the string “EQUIPMENT_END”.

When reconstructed the ArrayList will contain a listing of the Rover Drive system Type and the two RoverToolType attachments. The Drive and ToolTypes will be listed by their string converted names.

Index (0) in the array list will contain the Drive Type.
Communications Server API

Global

- **[POST] /api/global** **requires corp secret**
  
  should contain an array of json objects in following format
  
  ```
  [ 
    { 
      "x": 12, // coordinates must be Integers, not String
      "y": 14,  // ALL CAPS as in enums folder
      "terrain": "SAND", // ROCK, SOIL, GRAVEL, SAND, NONE
      "science": "CRYSTAL", // RADIOACTIVE, ORGANIC, MINERAL, CRYSTAL, NONE
      "f": 12 // Found by Rover 12 (for debugging)
      "g": 15 // Marked by Rover 15 for gathering
    }, ...
  ]
  ```

- **[GET] /api/global** **requires corp secret**
  
  Returns the global map in json array.

- **[GET] /api/global/size**
  
  Shows the size of the global map array. Useful for testing

- **[GET] /api/global/reset**
  
  Erases the data. Can be used when game restarts

- **[GET] /api/global/test**
  
  An example

Sciences

- **[GET] /api/science/all**
  
  Shows locations of all sciences
  
  ```
  [ 
    { 
      "x": 19,
      "y": 47,
    }
  ]
  ```
**[GET] /api/science/drill**  
Shows locations of sciences possible for driller

**[GET] /api/science/excavate**  
Shows locations of sciences possible for excavater

### Gather

**[POST] /api/gather/x/y**  
Mark the tile that you are going to gather this science
- header: 'Rover-Name': rovername (ie. ROVER_11)
- header: 'Corp-Secret': corp_secret
- POST: /api/gather/78/52

Now the tile looks like this:
- 
  - "x": 78,
  - "y": 52,
  - "terrain": "SAND",  
  - "science": "CRYSTAL",  
  - "f": 10  
  - "g": 11  

Coordinate

**[GET] /api/coord/:x/:y**  
Shows the coordinate at specified location
- example: /api/coord/82/18
- 
  - "x": 85,
  - "y": 18,
  - "science": "MINERAL",
  - "terrain": "GRAVEL"
  - "f": 17 // found by rover 17
- **[POST] /api/coord/:x/:y/:science** **requires corp secret**
  Update science information of this coordinate
  - header: 'Rover-Name' : rovername (ie. ROVER_14)
  - header: 'Corp-Secret': corp_secret
  - POST: /api/coord/30/40/DIAMOND

Now the coordinate looks like:

```json
{
    "x": 30,
    "y": 40,
    "science": "DIAMOND",  // now updated to DIAMOND. (but should only use: NONE, CRYSTAL, ORGANIC, RADIOACTIVE, MINERAL)
    "terrain": "SOIL"
    "f": 14                // updated by rover 14
}
```

**Misc.**

- **[GET] /api/roverinfo**
  Shows information of the rovers