CS 1302 – HW 8

*Maps & Comparators*

Contents

[1 Overview 1](#_Toc184721944)

[2 Requirements 1](#_Toc184721945)

[3 Extra Credit 2](#_Toc184721946)

[4 Step-by-Step 2](#_Toc184721947)

[5 Rubric 4](#_Toc184721948)

[6 Submission 4](#_Toc184721949)

[Appendix 1 n/a 4](#_Toc184721950)

# Overview

This homework has 1 problem which deals with maps and comparators. You will refactor the *MartianManager* class from HW 5 to utilize maps as the primary data structure and add a few methods and comparators.

To get started, do the following:

1. Create a Java Project in Eclipse with the name: *hw8\_lastName*
2. Drag your *prob2* folder from HW 5 to the *src* node in Eclipse for HW 8.
3. Rename *prob2* to *prob1* by doing the following:
   1. Select the *prob2* folder in Eclipse for HW 8
   2. Choose: Refactor, Rename (or *Alt-Shift-R*)
   3. Type in the name: *prob1*
   4. Choose: OK

# Requirements

These are the requirements that your code must fulfill. You can use these to complete the homework; or, you can go to the [next section](#_Step-by-Step) for step-by-step directions.

1. In the *MartianManager* class, replace the *martians* and *teleporters* lists with *HashMap*sstoring martians and teleporters, respectively, using the *id* as the key in both cases. Almost every method in the class will need to be changed to work with the *HashMaps.* Before beginning these updates, note the following:
   1. The battle method code will need to be changed; however, there is no change to the signature of the *battle* method, it continues to return *ArrayList<Martian>*.
   2. Remove *getMartianAt(int i)* it no longer has meaning.
   3. Rename *getMartianWithId(int id)* to *getMartian(int id)*.The purpose of the method remains the same: return the martian with *id* if it exists. The preferred way to rename is to do a Refactor, Rename as you did in Section 1 above, changing the *prob2* package to *prob1.*
   4. Rename *getTeleporterAt(int i)* to *getTeleporter(int id).* This method now will return the teleporter with *id* if it exists. Thus, it is almost identical to *getMartian* above.
   5. Remove *removeMartianAt(int i)* it no longer has meaning.
   6. Rename *removeMartianWithId(int id)* to *removeMartian(int id)*.The purpose of the method remains the same: return the teleporter with *id* if it exists.
2. Write a *MartianIdComparator* that will allow martians to be sorted on *id.*
3. Write a *MartianVolumeComparator* that will allow martians to be sorted on *volume*, then *id.* In other words, if two martians have the same volume, then compare on *id.*
4. The *getSortedMartians* method should be modified:
5. The method now accepts a string, *sortType* that defines what type of sort to perform. Valid values are "ID" and "VOLUME". A new, appropriately sorted list should be returned based on the *sortType*.

/\*\*\*

\* Returns a sorted list of martians using the Comparator specified by sortType.

\* **@param** sortType The type of sort to do. Valid values are "ID" and "VOLUME".

\* **@return** Sorted list of Martians either by ID or VOLUME.

\*/

**public** ArrayList<Martian> getSortedMartians(String sortType) {

1. Add a method, *increaseTeleporterVolume* that increases the volume by 1 for all teleporters whose *id* is less than *idThreshold*.

**public** **void** increaseTeleporterVolume(**int** idThreshold) {

...

}

1. Modify *MartianManagerTest* as needed so that all tests run.
2. Add these tests:

* *testGetSortedMartians\_ID* – maybe 5-6 total, of which 2 have the same ID.
* *testGetSortedMartians\_VOLUME* – maybe 5-6 total, of which 2 have the same Volume.
* *testIncreaseTeleporterVolume* – maybe 5 teleporters and 3 get increased

# Extra Credit

Add these classes to test the comparators:

1. *MartianIdComparatorTest* – Write test methods to test the *compare* method in the *MartianIdComparator* under various conditions. There should be at least 5 test methods in order to test all the conditions.
2. *MartianVolumeComparatorTest* – Write test methods to test the *compare* method in the *MartianVolumeComparator* under various conditions. There should be at least 5 test methods in order to test all the conditions.

# Step-by-Step

1. Replace the *martians* and *teleporters* lists with *HashMap* storing martians and teleporters, respectively, using the *id* as the key in both cases.
2. Remove *getMartianAt(int i);* it no longer has meaning.
3. Comment out all methods except: *getNumMartians* and *getNumTeleporters* by selecting the methods and the pressing: Ctrl+/. These two methods should not require any changes and thus, should compile.
4. Uncomment, *toString* by selecting the method and pressing: Ctrl+/. Then make the required changes.

Hint: you want to loop over the values in the *martians* (*teleporters*) map.

1. Comment out all your test methods in *MartianManagerTest* except, one for *testToString* method. Then, make any needed changes, and run the test.
2. Uncomment, *addMartian* in *MartianManager* by selecting the method and pressing: Ctrl+/. Then make the required changes*.*

Hint: You need the *id* to add the teleporter to the *teleporters* map. However, when you cast the incoming martian to a teleporter, the teleporter doesn’t have access to the id, the only thing it can do is: *teleport*. Thus, you need to get the id from the incoming martian.

1. Uncomment (select, then Ctrl+/) on of your *testAddMartian* methods (you should have several)in *MartianManagerTest*. Then, make any needed changes to *testAddMartian*, and then run test.

Note: Depending on how you wrote the test originally, you might not be able to modify the test until you also update (next) *getMartian* in *MartianManager*.

1. Uncomment (select, then Ctrl+/) *getMartianWithId(int id)*. Rename the method to *getMartian* by selecting the method name, and then choosing: Refactor, Rename; type in “getMartian” and OK.The purpose of the method remains the same: return the martian with id if it exists.

Hint: use the *containsKey* method to see if the id exists.

1. Uncomment the test method(s), *testGetMartian…* in *MartianManagerTest* and modify as needed. Then, run the tests.
2. Remove the *getMartianAt* method. It no longer makes sense because there is no index with a map. Remove corresponding test methods in *MartianManagerTest*.
3. Uncomment (select, then Ctrl+/) *getTeleporterAt(int i)*. Rename the method to *getTeleporter* by selecting the method name, and then choosing: Refactor, Rename; type in “getTeleporter” and OK.Next, change the parameter name, *i* to *id.* This method now will return the teleporter with the incoming id. Thus, it is almost identical to *getMartian* above.
4. Uncomment tests for *getTeleporter* in *MartianManagerTest*, modify the tests, and run.
5. Remove *removeMartianAt(int i)* it no longer has meaning.
6. Rename *removeMartianWithId(int id)* to *removeMartian(int id)*.The purpose of the method remains the same: return the teleporter with *id* if it exists.
7. Uncomment tests for *removeTeleporter* in *MartianManagerTest*, modify the tests, and run.
8. Continue this process with the rest of the methods. Notes:

* The *battle* method will continue to accept and return an *ArrayList* of *Martian*.
* The *getSortedMartians* method will change as described below (so save it for last).

1. Write a *MartianIdComparator* that will allow martians to be sorted on *id*. They can actually already be sorted on *id* because *Martian* implements *Comparable* which compares martians on *id*; however, you must write (and use later) the comparator.
2. Write a *MartianVolumeComparator* that will allow martians to be sorted on *volume*, then *id.* In other words, if two martians have the same volume, then compare on *id.*
3. The *getSortedMartians* method should be modified:
4. The method now accepts a string that defines what type of sort to perform. Valid values are "ID" and "VOLUME"

/\*\*\*

\* Returns a sorted list of martians using the Comparator specified by sortType.

\* **@param** sortType The type of sort to do. Valid values are "ID" and "VOLUME".

\* **@return** Sorted list of Martians either by ID or VOLUME.

\*/

**public** ArrayList<Martian> getSortedMartians(String sortType) {

1. Add a method, *increaseTeleporterVolume* that increases the volume by 1 for all teleporters whose *id* is less than *idThreshold*.

**public** **void** increaseTeleporterVolume(**int** idThreshold) {

...

}

1. Modify *MartianManagerTest* as needed so that all tests run. This involves just a little work, most change a few method names. All the test methods from HW 5 should be implemented.
2. Add these tests:

* *testGetSortedMartians\_ID* – maybe 5-6 martians, of which 2 have the same ID.
* *testGetSortedMartians\_VOLUME* – maybe 5-6 martians, of which 2 have the same Volume.
* *testIncreaseTeleporterVolume* – maybe 5 teleporters and 3 get increased

# Rubric

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Description** | **Points** |
| 1 | Automated Tests | 80 |
| 2 | *MartianManagerTest* updated & complete | 10 |
| 3 | 3 Test Methods added | 10 |
|  |  |  |
|  | Extra Credit | 10 |
|  | **Total** | **110** |

# Submission

1. Zip your *prob1* folder (also containing all test files) into a file name: *hw8\_yourLastName.zip.*
2. Submit on Blazeview in the the HW 8 drop box.

Appendix

1. n/a