CS 1302 – HW 5a

*Martian Manager Gui, Sprint 1*

Contents

[1 Overview 1](#_Toc192326072)

[2 Requirements 1](#_Toc192326073)

[3 Steps to Complete 4](#_Toc192326074)

[4 Rubric 5](#_Toc192326075)

[5 Submission Requirements 5](#_Toc192326076)

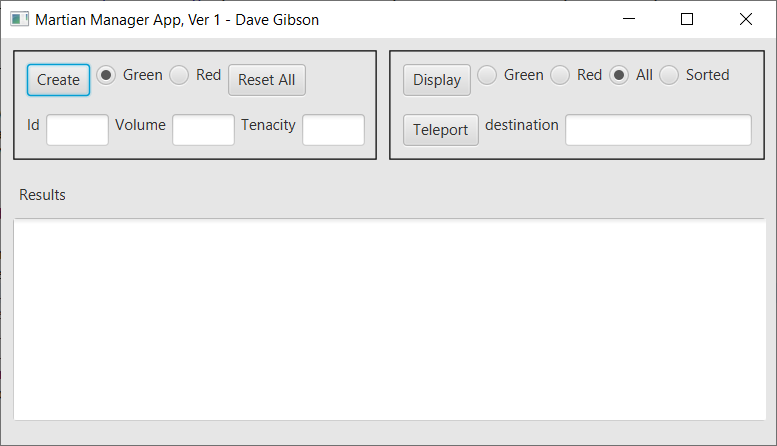
[Appendix 1 n/a 5](#_Toc192326077)

# Overview

During Sprint 1, you will develop the first version of a Martian Manager App, *MMA, ver 1.* You will utilize a Jar file that contains the *MartainManager* and related classes.

# Requirements

1. When the app is run, the initial display is as shown below.



Hints:

1. Add these styles to *application.css*

/\* These styles are automatically applied \*/

.root {

-fx-background-color:**rgb(**230**,**230**,**230**)**;

-fx-padding: 5px, 5px, 5px, 5px;

-fx-hgap: 5px;

-fx-vgap: 5px;

}

**HBox**, **VBox** {

-fx-padding: 5px;

-fx-spacing: 5px;

}

.text-area {

-fx-font-family: Consolas;

}

/\* These styles must be applied with code \*/

.textFieldCreate {

-fx-pref-width: 50px;

}

.boxBorder {

-fx-border-color: black;

-fx-border-insets: 5;

-fx-border-width: 1;

-fx-border-style: solid;

}

1. To add a border around a pane:
   1. Define a style rule. For example, see *.boxBorder* above.
   2. Apply the rule with code: vBox.getStyleClass().add("boxBorder");
2. To make the text fields have smaller width:
   1. Define a style rule. For example, see *.textFieldCreate* above.
   2. Apply the rule with code: txfId.getStyleClass().add("textFieldCreate");

**In the *Create* section of the app:**

Define an instance variable to hold a *MartianManager* object and create it. This is very important. It is the memory/state of your app. It will hold all martians that are created.

1. When *Create* is pressed, when *green* is selected and an *id* and *volume* are entered, then: (a) a *GreenMartian* is created and added to the *MartianManager*, (b) the text fields are cleared (c) the *Results* section shows exactly (for example):

Martian created:

Green Martian - id=3, vol=8

Hints:

1. You can clear a text field with code like this: txfId.clear();. You might consider writing a helper method to clear all the text fields.
2. You can use the martian’s *toString* to create part of the required display in the *Results* section.
3. When *Create* is pressed, when *red* is selected and an *id, volume* and tenacityare entered, then: (a) a *RedMartian* is created and added to the *MartianManager*, (b) the text fields are cleared (c) the *Results* section shows exactly (for example):

Martian created:

Red Martian - id=4, vol=5, ten=6

1. When an attempt to create a martian with an *id* that already exists: (a) the text fields are cleared and (b) the *Results* section shows exactly (for example):

\*\*\*Martian not added.

Martian with id=5 already exists

Hint: Look at the Javadoc for the *MartianManager’s* *addMartian* method to understand what it returns and how it is useful here.

1. When *Reset All* is pressed: (a) all martians in memory are removed, (b) all text fields are cleared, (c) the *Results* section shows exactly:

All martians cleared from memory

Hint: The simplest way to do this is to simply create a new *MartianManager*.

**In the *Display* section of the app:**

1. When *Display* is pressed when *All* is selected the *Results* section shows all martians in memory (in the *MartianManager*), in the order they were added. The display should exactly show (for example):

All Martians

---------------

Green Martian - id=22, vol=4

Green Martian - id=12, vol=7

Red Martian - id=28, vol=16, ten=22

Green Martian - id=53, vol=22

Red Martian - id=5, vol=3, ten=11

Hint: How can you iterate over all the martians in the *MartianManager*? It has a method to tell you how many martians there are and a method to get a martian at a particular index. Thus, you can use an indexed loop to loop over the martians in the *MartianManager.* Use the Javadoc to examine the *MartianManager* methods to see what methods will help you.

1. When *Display* is pressed when *Green* is selected the *Results* section shows all green martians in memory (in the *MartianManager*), in the order they were added. The display should exactly show (for example):

Green Martians

-----------------

Green Martian - id=22, vol=4

Green Martian - id=12, vol=7

Green Martian - id=53, vol=22

Hint: (a) Same as previous requirement. (b) How can you filter out the green martians? See Ch 3, Sec 9 in the text.

1. When *Display* is pressed when *Red* is selected the *Results* section shows all red martians in memory, in the order they were added. The display should exactly show (for example):

Red Martians

---------------

Red Martian - id=28, vol=16, ten=22

Red Martian - id=5, vol=3, ten=11

1. When *Display* is pressed when *Sorted* is selected the *Results* section shows all martians in memory, sorted on their *id*. The display should exactly show (for example):

Sorted Martians

------------------

Red Martian - id=5, vol=3, ten=11

Green Martian - id=12, vol=7

Green Martian - id=22, vol=4

Red Martian - id=28, vol=16, ten=22

Green Martian - id=53, vol=22

Hint: The *MartianManager* has a method that produces this.

1. When *Teleport* is pressed, teleporters should teleport to the *destination* entered and the *Results* section shows the result in exactly this format(for example, if “Ork” is the *destination*):

Teleporting

-------------------------------

id=22 teleporting to Ork

id=12 teleporting to Ork

id=53 teleporting to Ork

Hint: The *MartianManager* has a method that produces this.

**In the *Create* section of the app:**

1. When *Create* is pressed, when *green* is selected and only an *id* is entered, then: (a) a *GreenMartian* is created using the appropriate constructor and added to the *MartianManager*, (b) the text fields are cleared (c) the *Results* section shows exactly (for example):

Martian created:

Green Martian - id=3, vol=1

Hint: you can check if a textfield is empty with code like this:

if (txfName.getText() == null || txfName.getText().trim().isEmpty())

1. When *Create* is pressed, when *red* is selected and only an *id* and *tenacity* are entered, then: (a) a *RedMartian* is created using the appropriate constructor and added to the *MartianManager*, (b) the text fields are cleared (c) the *Results* section shows exactly (for example):

Martian created:

Red Martian - id=28, vol=16, ten=22

1. Create a video in Blazeview demoing each of the requirements. The format should be: state requirement, then demo, repeat.

# Steps to Complete

1. Reference the *MartianManager* Javadoc from HW 5: <https://cs.valdosta.edu/~dgibson/courses/cs1302/hw/hw5_doc/index.html>
2. Do a quick sketch to see how you are going to layout the Gui considering the 3 *Panes* we discussed. Decide on the root pane, and how panes will be nested. See Ch 11 in the text if needed.
3. Create a JavaFX project. See Lab 15a for directions if needed.
4. Add the Jar File. See Lab 15a for directions if needed.
5. Add the *MartianManager* instance variable.
6. From here, there are two approaches you can take:
   1. Baby Steps: Create a small portion of the Gui, for example, (a) just the first row of the Create section and test (display), then (b) second row, display, (c) add Results area and display, (d) write Create event handler for just Green, test, *etc.* I would use this approach.
   2. Create the entire Gui, then write the event handlers going in the ordered listed above.

# Rubric

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Points** | **Description** |
| 1 | 40 | Initial Display |
| 2 | 5 | Create Green |
| 3 | 5 | Create Red |
| 4 | 5 | Attempt to create with existing ID |
| 5 | 5 | Reset All |
| 6 | 5 | Display All |
| 7 | 5 | Display Green |
| 8 | 5 | Display Red |
| 9 | 5 | Display Sorted |
| 10 | 5 | Teleport |
| 11 | 7 | Create Green with alternate constructor |
| 12 | 8 | Create Red with alternate constructor |

# Submission Requirements

Checklist:

|  |  |  |
| --- | --- | --- |
|  | **Complete?** | **Requirement** |
| 1. |  | Your *application* folder is zipped into a file name: *hw5a\_yourLastName.zip.*   * See Lab 2, Stage 9 for exact instructions. * Do not zip your workspace folder * Do not zip your *src* folder. * Do not zip just the java files * Do zip just your *application* folder |
| 2. |  | Video complete |
| 3. |  | Submit video and zip file in the *hw 5a* dropbox on Blazeview by the deadline. |

Appendix

1. n/a