**Familiarity Review**

**Name:** Jeremy James Alkire

**Date:** February 4th, 2019

**Week:** 5

**Coding Topic:** MVC

**Description of Understanding:** The MVC is a three-part architectural design pattern used for developing user interfaces. The three components are the model, view, and controller. The model manages the data and logic of the application. The view displays a visual representation of the data. It handles both input and output and is what the user interacts with. The model and view components do not communicate with each other directly, they are both managed by the controller, which acts as a medium between the two. The controller is essentially the engine of the program. It calls methods from the view and the model and passes data between them as necessary.

**Teaching Video:**

**Starting at:**

**Also Integrated with:** Use Case Diagrams, Use Case Documents, Sequence Diagrams

|  |  |  |  |
| --- | --- | --- | --- |
| **File** | **Git Link** | **What should I be looking for?** | **Sandbox or Your code?** |
| RequestData.java | <https://github.com/alkire-jeremy/CIT360/blob/master/Familiarity%20Requests/Request%201%20-%20Week%204/Code/fr1/view/RequestData.java> | This class operates as the **view** within the MVC architecture. The method *requestAthleteData()* outputs a prompt to the user that requests that they input da ta into the console. That data is handled and stored in method variables and returned to the method caller. It functions as a text-based UI. By itself, it does nothing. The controller calls its functions. | My code. |
| Athlete.java | <https://github.com/alkire-jeremy/CIT360/blob/master/Familiarity%20Requests/Request%201%20-%20Week%204/Code/fr1/model/Athlete.java> | This class operates as the **model** within the MVC architecture. It specifies the attributes of the Athlete object. It has private non-static variables, public getters & setters, and a universal configuration method *configureAthlete()* that allows you to set the value of all variables with a single function call (and three arguments.) | My code. |
| Engine.java | <https://github.com/alkire-jeremy/CIT360/tree/master/Familiarity%20Requests/Request%201%20-%20Week%204/Code/fr1/controller> | This class operates as the **controller** within the MVC architecture. It creates an object instance of the Athlete (model) class, calls *requestAthleteData()* from the RequestData (view) class and stores its output in a string variable. It then splits the string into a list of strings by comma (,) characters. From there, it calls *configureAthlete()* from the model, and passes each element in the list to model’s method via its parameters. | My code. |

**Coding Topic:** Java Collections

**Description of Understanding:**

**Teaching Video:** <https://www.youtube.com/watch?v=TNn_XDoZ8>

**Starting at:** 8:14

**Also Integrated with:** I am writing tests for MVC and Java Collections

|  |  |  |  |
| --- | --- | --- | --- |
| **File** | **Git Link** | **What should I be looking for?** | **Sandbox or Your code?** |
| MyPortTest.java | [**https://github.com/tuckettt/CIT360-Portfolio.git**](https://github.com/tucketttCIT360-Portfolio.git) | This is the class being tested | Mine |
| MyPortTestTest.java | [**https://github.com/tuckettt/CIT360-Portfolio.git**](https://github.com/tucketttCIT360-Portfolio.git) | This is the test class. | Sandbox |

**Diagram**: UML Class Diagram

**Description of Understanding:** I was able to use a Class Diagram as a part of the design of my DVD Collection application. The diagram helped me to identify the necessary classes and interfaces.

**Teaching Video:** None

|  |  |  |
| --- | --- | --- |
| **File** | **What should I be looking for?** | **Example**  **Or Your code?** |
| /myrepository/uml/classdiagrams/dvdcollection.pdf | My first try at this diagram. | My code |