

L04B: Assignment - Scrolling: Add On to the Basic Window and Data Assignment

Due Aug 2 by 11:59pm **Points** 10 **Submitting** a file upload **File Types** zip
Available until Aug 2 at 11:59pm

This assignment was locked Aug 2 at 11:59pm.

After reading all **L04** content pages in [Lesson 04: Graphical User Interfaces](#), you will complete this assignment according to the following information:

Assignment Objectives

1. Practice on implementing a MVC project
2. Keep working with Java Graphics
3. Implement Listener Interfaces
 1. MouseWheelListener
 2. ActionListener

Deliverables

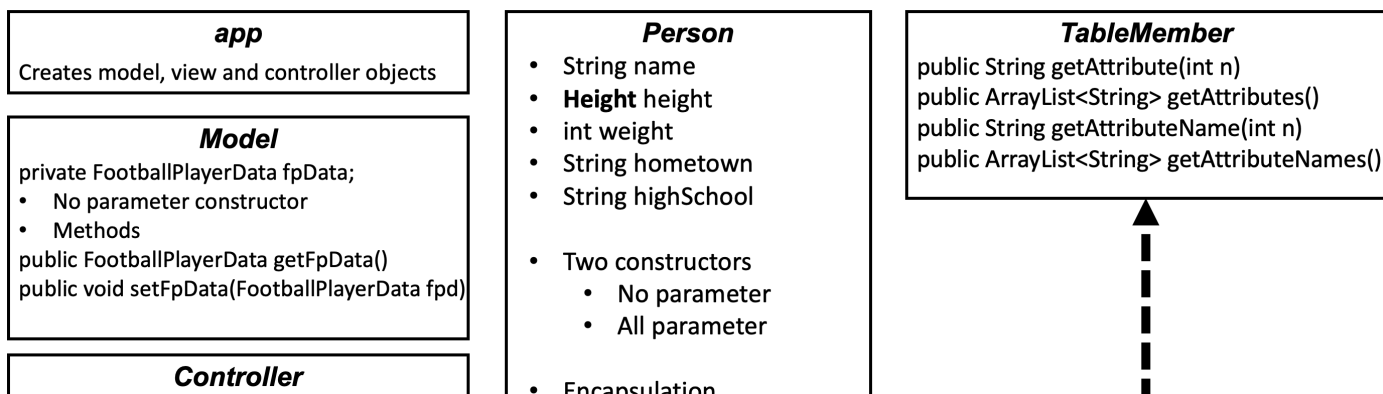
A zipped Java project according to the [How to submit Labs and Assignments guide](#).

O.O. Requirements (these items will be part of your grade)

1. One class, one file. Don't create multiple classes in the same .java file
2. Don't use static variables and methods
3. Encapsulation: make sure you protect your class variables and provide access to them through get and set methods
4. all the classes are required to have a constructor that receives **all** the attributes as parameters and update the attributes accordingly
5. Follow [Horstmann's Java Language Coding Guidelines](#)
(http://higheredbcs.wiley.com/legacy/college/horstmann/1118087887/style_guide/styleguide.htm)
6. Organized in packages (MVC - Model - View Controller)

Contents

Same classes as the last assignment but with increased functionality.



- Model model
- View view
- All parameter constructor

View

- No parameter constructor
- Methods
- void basicDisplay(String s)
- void basicDisplay(ArrayList<String> arr)

InitialFrame

- InitialPanel ip
- Two constructors
 - No parameter
 - All parameter
- Encapsulation
 - Private attributes
 - Get and set methods

InitialPanel

- CenterPanel cp
- Two constructors
 - No parameter
 - All parameter
- Encapsulation
 - Private attributes
 - Get and set methods

CenterPanel

- Attributes
- Two constructors
 - No parameter
 - All parameter
- Encapsulation
 - Private attributes
 - Get and set methods

- Private attributes
- Get and set methods
- String toString()

extends

implements

FootballPlayer

- int number
- String position
- Two constructors
 - No parameter
 - All parameter
- Encapsulation
 - Private attributes
 - Get and set methods
- String toString()

Height

- int feet
- Int inches
- Two constructors
 - No parameter
 - All parameter
- Encapsulation
 - Private attributes
 - Get and set methods
- String toString()
 - Modified to display
 - 9'99"

TableData

```
public void loadTable( );
public ArrayList <TableMember> getTable( );
public ArrayList<String> getHeaders( );
public ArrayList<String> getLine(int line);
public ArrayList<ArrayList<String>> getLines(int firstLine, int lastLine);
```

Displayable

```
public int getFirstLineToDisplay();
public int getLineToHighlight();
public int getLastLineToDisplay();
public int getLinesBeingDisplayed();

public void setFirstLineToDisplay(int firstLine);
public void setLineToHighlight(int highlightedLine);
public void setLastLineToDisplay(int lastLine);
public void setLinesBeingDisplayed(int numberOfLines);
```

implements

implements

FootballPlayerData

```
ArrayList<TableMember> players;

void ReadPlayersFromXML( ) {...}

public void loadTable( ) {...}
public ArrayList<TableMember> getTable( ) {...}
public ArrayList<String> getHeaders( ) {...}
public ArrayList<String> getLine(int line) {...}
public ArrayList<ArrayList<String>> getLines(int firstLine, int lastLine) {...}
```

Starting from the last assignment (fully functional), you will

- Add a `MouseWheelListener` to enable scrolling
- Add an `ActionListener` to enable listening of specific buttons
- See a [video of the assignment functionality at work](#).

Scrolling

- The scrolling functionality will be implemented with the Interface `MouseWheelListener`
 - see the lesson [here](#)

```
public Controller(Model m, View v)
{
    model = m;
    view = v;
    view.CenterInitialSetup(model.getFpData().getLinesBeingDisplayed(), model.getFpData().getHeaders().size());
    view.CenterUpdate(model.getFpData().getLines(model.getFpData().getFirstLineToDisplay(), model.getFpData().getLastLineToDisplay()));
    this.addListeners();
}

private void addListeners()
{
    view.getInitialframe().getInitialPanel().getCp().addMouseWheelListener(
        new MouseWheelListener()
        {
            @Override
            public void mouseWheelMoved(MouseWheelEvent e)
            {
                int units = e.getUnitsToScroll();
                //use "units" to update first and last line to display
                //using the methods you have available in in the class that implemented Displayable, FootballPlayerData.
                //Then refresh the screen using CenterUpdate in view
            }
        }
    );
}
```

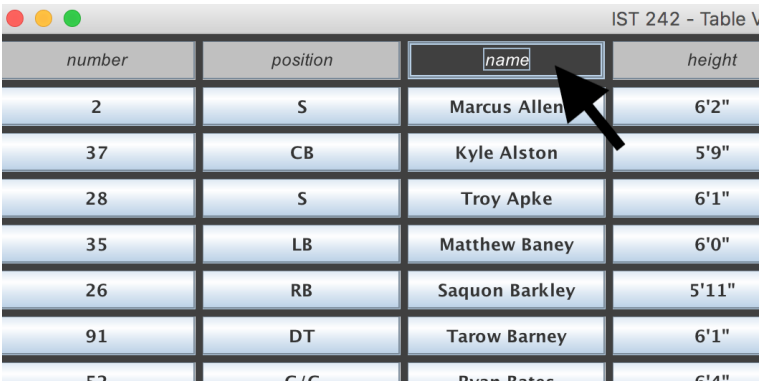
- You **cannot** use a `JScrollPane` or any other embedded Java scrolling functionality.
- You need to write the scrolling functionality yourself using `MouseWheelListener` implemented as an inner listener.
- Strategies to implement scrolling
 1. keep sending a number of lines as Strings at a time to the panel. Which lines to be sent will depend on the mouse wheel movement.
 1. If you need to define the first line to display, use the method `setFirstLineToDisplay(int n)`
 2. If you need to retrieve the first line to display, use the method `getFirstLineToDisplay()`
 3. the number of lines to go up or down is given by the method `"getUnitsToScroll()"`(from the mouse wheel event variable) as the mouse is scrolled up or down
 4. adding `getUnitsToScroll()` to `FirstLineToDisplay` gives you the new value that will be used to retrieve lines from the data table and send them to be displayed.
 5. you will need to check if the final value of `FirstLineToDisplay` is not below or above the data array size. If you try to retrieve lines that are not in the data table you will probably get an error message saying "array out of bounds exception"
 6. See [the mouse wheel listener topic in the Listeners lesson](#).
 7. Examples from the lesson
 1. [MouseWheelInnerListener_Example.zip](#) and [MouseWheelRegularListener Example.zip](#).

Listening to the Headers

The listening to the headers functionality will be implemented using the Interface `ActionListener`.

- It will be implemented as an inner listener
 - check the lesson on [inner listeners](#)

- one ActionListener for each header button
- once the button is clicked it should be highlighted using a different color. All other buttons will then be reset to the standard color



<i>number</i>	<i>position</i>	<i>name</i>	<i>height</i>
2	S	Marcus Allen	6'2"
37	CB	Kyle Alston	5'9"
28	S	Troy Apke	6'1"
35	LB	Matthew Baney	6'0"
26	RB	Saquon Barkley	5'11"
91	DT	Tarow Barney	6'1"
52	OLG	Russ Bates	6'11"

Everything else is as defined in the previous assignment, L04AC.