

Ungraded Inclass Exercise

TextBrowser

Scenario

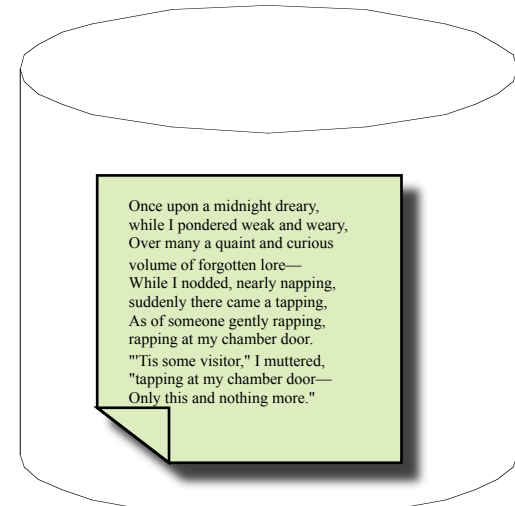
- Imagine the problem of browsing the text in a computer file
- Imagine that no GUI (Graphical User Interface) toolkit supplied a single widget to do this
- Imagine that you would like to devise a cleanly structured solution

Scenario - 2

- From what components would you build your TextBrowser?

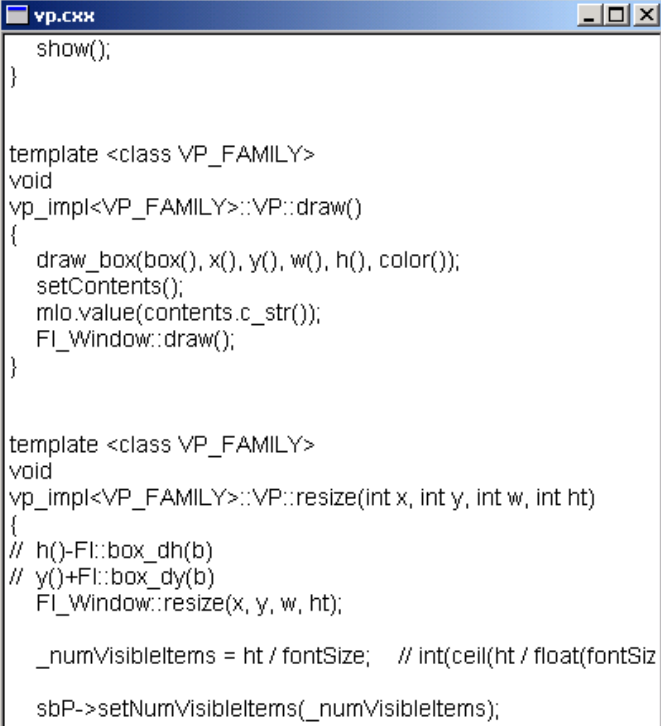
FileManager

- You need some way to access the file's contents
 - Assume that you cannot hold the entire file contents in memory
 - Assume that you have, at the operating system level, line-oriented access to the file
- You need to provide a module that, when requested can retrieve a limited size, consecutive subsequence of the file lines
 - Don't worry about file opening or closing



ViewPort

- You need to be able to display the textual content graphically
 - Assume the ViewPort displays an integer number of lines
 - Assume the ViewPort can be resized to be of length 1 to 100 lines
 - Assume that all text is in the same font and the same point size



```
vp.cpp
show();
}

template <class VP_FAMILY>
void
vp_impl<VP_FAMILY>::VP::draw()
{
    draw_box(box(), x(), y(), w(), h(), color());
    setContents();
    mIo.value(contents.c_str());
    FI_Window::draw();
}

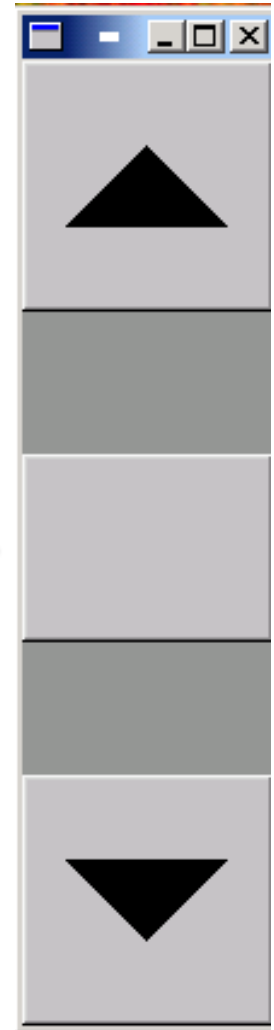
template <class VP_FAMILY>
void
vp_impl<VP_FAMILY>::VP::resize(int x, int y, int w, int ht)
{
    // h()-FI::box_dh(b)
    // y()+FI::box_dy(b)
    FI_Window::resize(x, y, w, ht);

    _numVisibleItems = ht / fontSize; // int(ceil(ht / float(fontSiz

    sbP->setNumVisibleItems(_numVisibleItems);
```

ScrollBar

- You need to give the user some way to access different parts of the file that can't fit on the screen simultaneously
 - Assume a vertical ScrollBar with a movable *handle* that sits in a *tray*
 - The handle position should denote that part of the file that should be displayed in the viewport
 - The size of the handle (in proportion to the size of the tray) should denote what portion of the file is visible



Question 1

- What use cases apply?

Question 1

- What use cases apply?
 - User displays file

Question 1

- What use cases apply?
 - User displays file
 - User moves cursor

Question 1

- What use cases apply?
 - User displays file
 - User moves cursor
 - User resizes viewport

Question 1

- What use cases apply?
 - User displays file
 - User moves cursor
 - User resizes viewport
 - ...

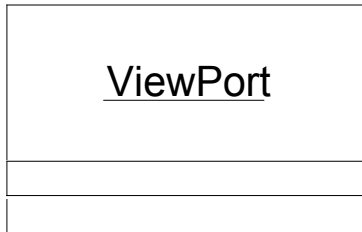
Question 2

- Draw a UML class model diagram presenting these requirements
 - That is, it should be an *analysis* model and not a *design* model
 - Include the three classes, their externally visible operations (`resizeWindow`, `moveCursor`) and their externally visible attributes
 - Include labeled associations
 - Do not include any further detail such as cardinality, navigation or role names

Classes

- Coming up with the classes should be easy
 - FileManager
 - ViewPort
 - ScrollBar

Classes - 2



Operations

- Now we can add operations
- In an analysis model, operations comprise those actions that the user can undertake to interact with the TextBrowser
- We can use our use cases to answer the question: What externally visible operations does the TextBrowser respond to?

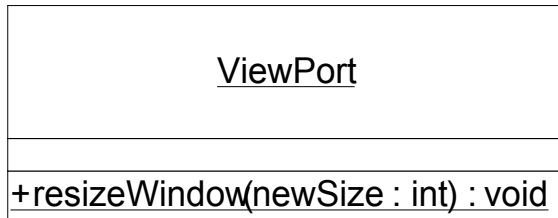
Operations - 2

- Now we can add operations
- In an analysis model, operations comprise those actions that the user can undertake to interact with the TextBrowser
- We can use our use cases to answer the question: What externally visible operations does the TextBrowser respond to?
 - Resizing the ViewPort
 - Moving the ScrollBar handle

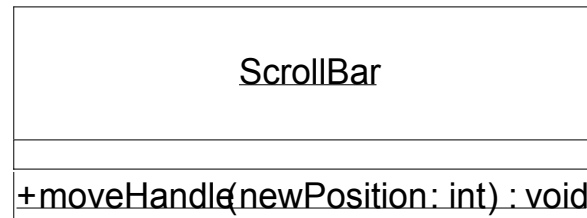
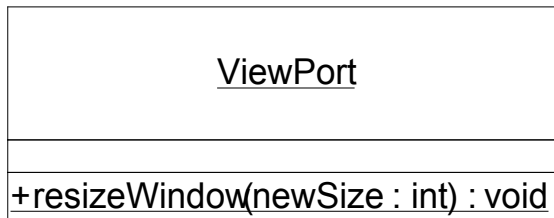
Operations - 3

- Now we can add operations
- In an analysis model, operations comprise those actions that the user can undertake to interact with the TextBrowser
- We can use our use cases to answer the question: What externally visible operations does the TextBrowser respond to?
 - Resizing the ViewPort
 - Moving the ScrollBar handle
- We can also specify parameters and return types

Operations - 4



Operations - 5



- Note some subtleties here
 - The fact that the ViewPort size is an int is not precisely what the requirements say. They also say it must be ≥ 1 and ≤ 100
 - We assigned arbitrary (integer) positions to the ScrollBar handle

Externally Visible Attributes

- We must translate what we can see in the GUI into explicit attributes
- Can you think what those attributes (*percepts*) should be?

Attributes - 2

- We must translate what we can in the GUI into explicit attributes
- Can you think what the attributes should be?
 - The position of the ScrollBar handle

Attributes - 2

- We must translate what we can in the GUI into explicit attributes
- Can you think what the attributes should be?
 - The position of the ScrollBar handle
 - The size of the ScrollBar handle

Attributes - 2

- We must translate what we can in the GUI into explicit attributes
- Can you think what the attributes should be?
 - The position of the ScrollBar handle
 - The size of the ScrollBar handle
 - The size of the ViewPort

Attributes - 2

- We must translate what we can in the GUI into explicit attributes
- Can you think what the attributes should be?
 - The position of the ScrollBar handle
 - The size of the ScrollBar handle
 - The size of the ViewPort
 - The contents of the ViewPort

Attributes - 2

- We must translate what we can in the GUI into explicit attributes
- Can you think what the attributes should be?
 - The position of the ScrollBar handle
 - The size of the ScrollBar handle
 - The size of the ViewPort
 - The contents of the ViewPort
- What about the FileManager?

Attributes - 3

- Going back to the use cases, if we were to draw the actual diagram, we would have an icon for the user
- Such icons are called *actors* in use case diagrams
- Can anyone think of another actor that should appear in the diagram?



Attributes - 3

- Going back to the use cases, if we were to draw the actual diagram, we would have an icon for the user
- Such icons are called *actors* in use case diagrams
- Can anyone think of another actor that should appear in the diagram?
 - The operating system (or file system)
 - That is, the document to be browsed is visible externally to the operating system



Attributes - 4

<u>ViewPort</u>
+height: int
+viewContents: sequence(lines)
+resizeWindow(newSize: int) : void

<u>ScrollBar</u>
+handleSize: int
+handlePosition: int
+moveHandle(newPosition: int) : void

<u>FileManager</u>
+document: sequence(lines)

Relationships

- Classes, operations, and attributes are usually the easiest parts of modeling
- Much harder are the relationships among the classes
- In an analysis model, you should be concerned with several types of relationships: associations, aggregation and generalization
- What properties should the system exhibit in order to qualify as being a correctly functioning TextBrowser?

Relationships - 2

- You can begin by asking what are the direct effects when the user makes a change?

Relationships - 2

- You can begin by asking what are the direct effects when the user makes a change?
 - `resizeWindow`: the ViewPort has a new size

Relationships - 2

- You can begin by asking what are the direct effects when the user makes a change?
 - `resizeWindow`: the ViewPort has a new size
 - `moveHandle`: the position of the handle in the tray changes

Relationships - 2

- You can begin by asking what are the direct effects when the user makes a change?
 - `resizeWindow`: the ViewPort has a new size
 - `moveHandle`: the position of the handle in the tray changes
- Unfortunately, we can't model these properties graphically in UML, so they'll have to wait until we talk about OCL

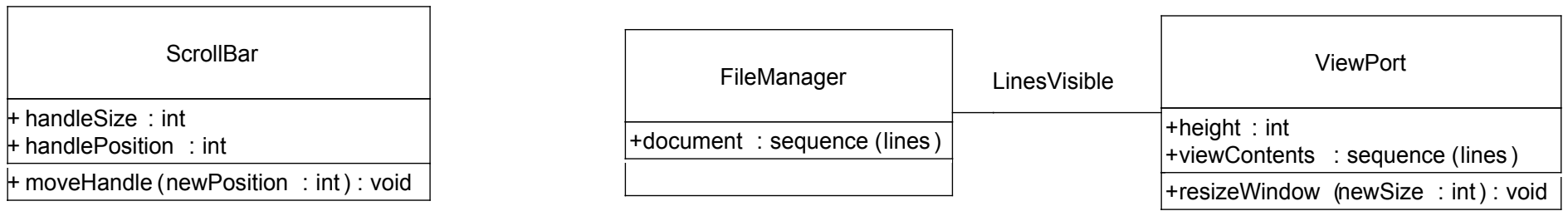
Relationships - 3

- More interesting are the indirect effects
- We know that the contents of the file, the position and size of the handle and the contents and size of the ViewPort are all related
- Can we break out the underlying relationships? What, for example, is the relationship between what we see in the ViewPort and the contents of the document?

Relationship - 4

- The ViewPort presents a maximum consecutive subsequence of whole lines from the document that fit into it
 - We can call this the `LinesVisible` association
- We can show its existence graphically with a labeled line between the two classes, but we will need OCL to supply the necessary precision

LinesVisible



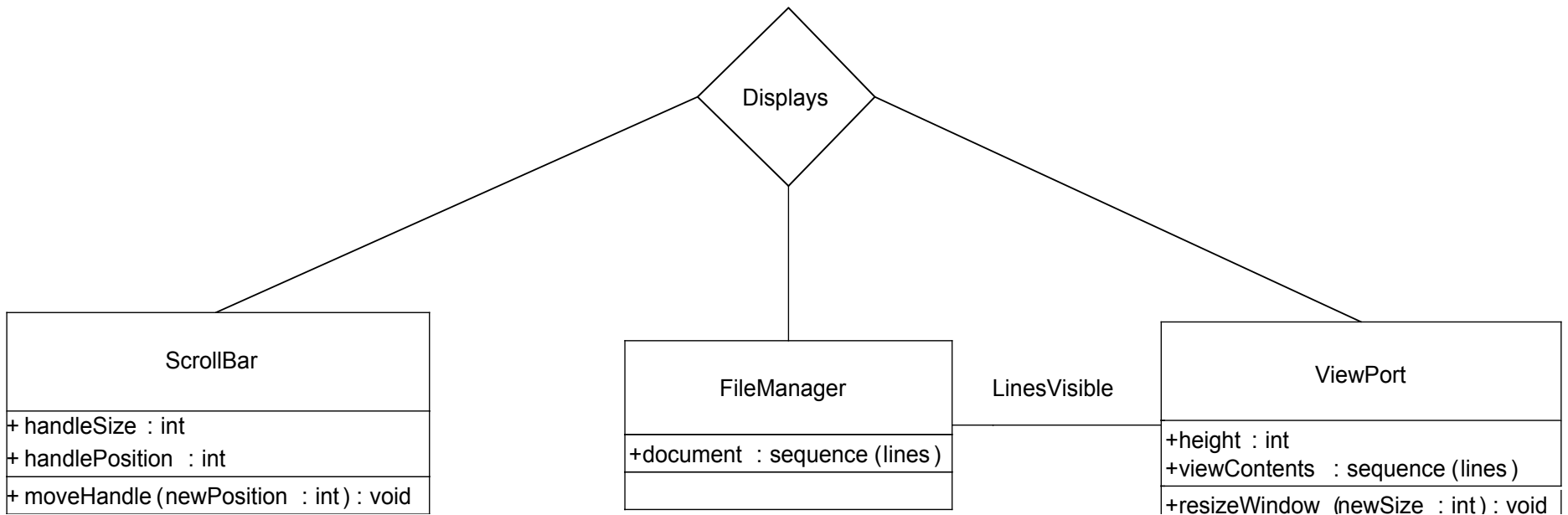
Another Property

- `LinesVisible` indicates that the contents of the `ViewPort` must come from the `FileManager` but it doesn't really say what lines
- That is determined by the position of the `ScrollBar` handle
- Can you state exactly what this relationship is?

Displays

- The position of the ScrollBar handle with respect to the ScrollBar tray reflects the position in the document of the top line visible in the ViewPort
- Let's call this association `Displays` (as in the ViewPort `Displays` the contents provided by the FileManager and determined by the ScrollBar)
- This is a three-way (ternary) association
- Once again, the precise details will have to await OCL

Displays - 2



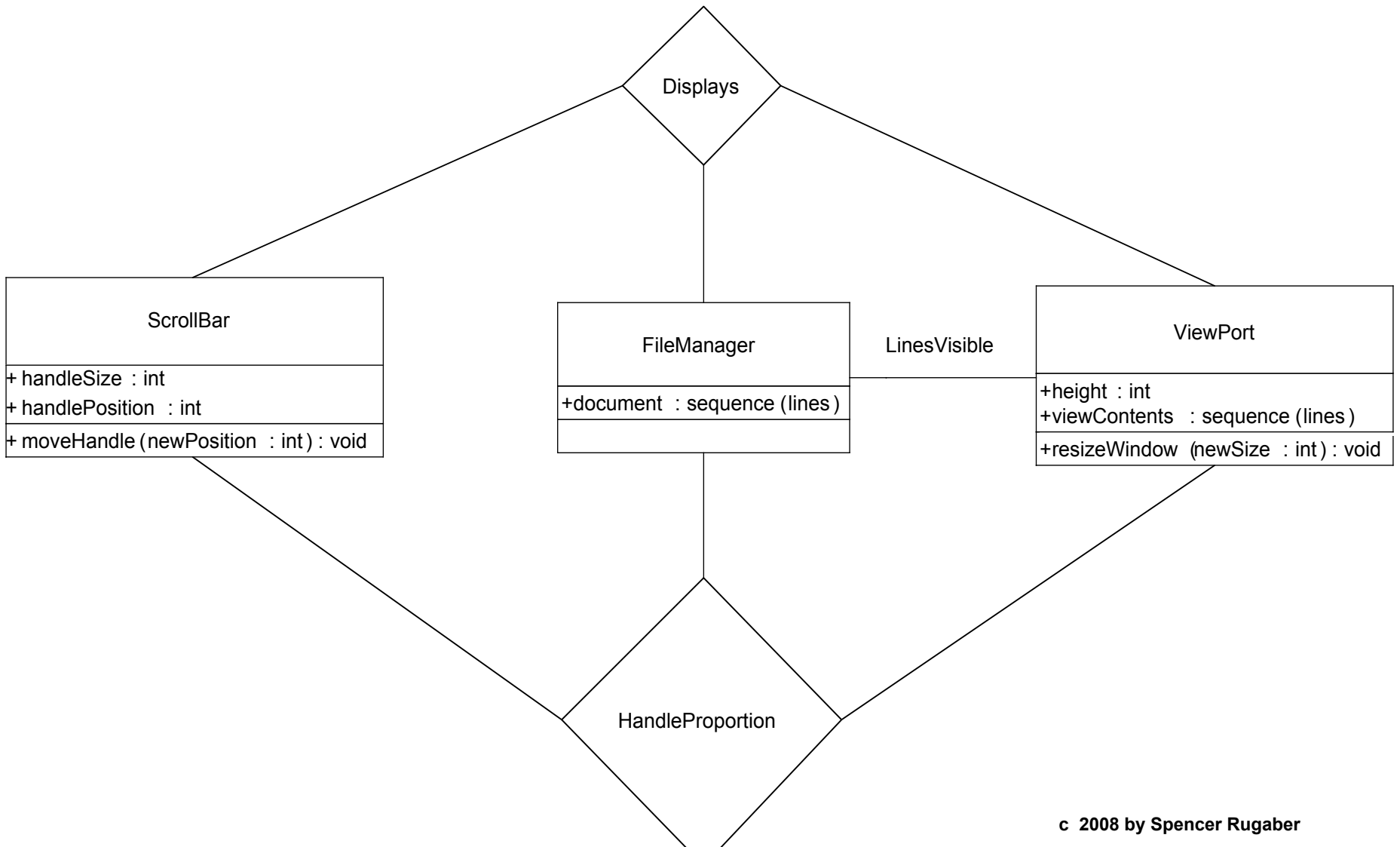
One More Property

- The final property we must describe has to do with the size of the handle of the ScrollBar
- Can you state what its association with the other components should be?

HandleProportion

- The size of the ScrollBar handle with respect to the size of the ScrollBar tray indicates the portion of the document's lines provided by the FileManager that are currently visible in the ViewPort
- Let's call this the `HandleProportion` association
- It is also ternary and can be described precisely with OCL

HandleProportion - 2



Subtleties

- Note that there are still some issues that we haven't dealt with
 - What exactly do we mean by the position of the handle?
 - What should we do about zero-length documents?
 - What lines should be displayed when we resize the ViewPort?

Question 4

- What is the key design question with which any implementation of the TextBrowser must deal?