Prof. Dr. Daniel Huson
Algorithmen der Bioinformatik
Fachbereich Informatik
Mathematisch-Naturwissenschaftliche Fakultät

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Advanced Java for Bioinformatics

Summer 2025

Assignment 3

Due: 15-May-2025

Later in the course, we will be working with three datasets:

- The "part-of" list of anatomy parts and a hierarchy of all "part-of" relationships.
- The 'is-a" list of anatomy parts and a hierarchy of "is-a" relationships.
- 3D models of all parts.

In this assignment, we will only work with the first item, the "part-of" list.

Note: For this and all following assignments, please create a separate package (e.g., `assignment3`), and make your implementation fully self-contained. So, for example, the package `assignment3.model` should contain its own definition of the class `ANode`, and must not refer to versions from earlier assignments.

This assignment focuses on using FXML to set up the user interface (UI) and on organizing your code into clearly separated components.

Following the structure discussed in the lecture, please implement the following files in your package `assignment3`:

- `WordExplorerMain.java` – the main application entry point
- `Window.fxml` – the FXML layout file
  - Place this file in: `src/main/resources/assignment3/window`
  - Ensure it is accessible by adding the following line to your `module-info.java`: `opens assignment3.window to javafx.fxml;`
- `window.WindowView.java` – constructs the view
- `window.WindowController.java` – provides access to the scene graph nodes defined in the FXML file
- `window.WindowPresenter.java` – connects the view to the model
- `window.TreeViewSetup.java` – sets up the tree view for the anatomy tree
- `model` package – contains all data structures, algorithms, and I/O (no JavaFX nodes)
  - `ANode` – a class representing an anatomy tree node (as introduced in Assignment 2)
  - `Model` – loads the part-of tree and provides access to its root
  - `TreeLoader` – parses input files to construct the tree (as in Assignment 2)
  - `WordCloudItem` – a record representing a word and its relative height (range 0–1)

(Also remember to add `exports assignment3;` to your `module-info.java`.)

As we will see in an upcoming lecture, this structure follows the *Model-View-Presenter* (MVP) design pattern.

The goal of this assignment is to write a program that displays the part-of anatomy tree in a tree view. When the user selects one or more items in the tree, a word cloud should appear, showing all words found in the selected items. Words should be sorted by number of occurrences, and their font sizes should be scaled using the square root of their frequency. The most frequent word should be displayed with a font size of 64.

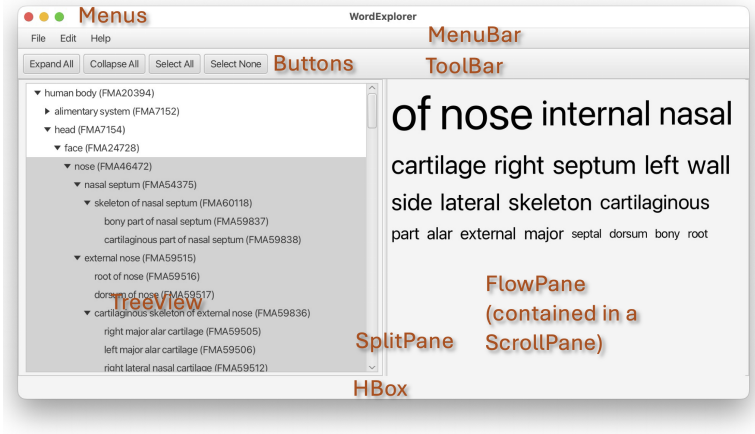
1 Designing the UI using Scene Builder (3 points)

Use Scene Builder to design the UI and save the layout to the file `Window.fxml`.

Assign `fx:id` values to all controls. For example, for the `Close` menu item, set `fx:id = closeMenuItem`.

Set the controller class to `assignment3.window.WindowController`, and use the menu item `View` → `Show Sample Skeleton` to generate starter content for the controller class.

The UI should resemble the layout shown below:



The word cloud will consist of individual `Text` nodes placed inside a `FlowPane`.

Draw a diagram showing the structure of the scene graph, indicating parent-child relationships among layout nodes and controls.

2 Implementing the Basic Model and Tree View (1 point)

To implement the basic model, read the part-of tree and return its root node.

Reuse your `ANode` and `TreeLoader` classes from Assignment 2.

For this assignment, the `Model` class is minimal. It loads the tree and provides access to the root:

```
public class Model {
  private final ANode partOfRoot;

  public Model() throws IOException {
```

```
    partOfRoot = TreeLoader.load(
        "partof_parts_list_e.txt",
        "partof_element_parts.txt",
        "partof_inclusion_relation_list.txt");
  }

  public ANode getPartOfRoot() {
    return partOfRoot;
  }
}
```

To display the part-of tree, reuse your `TreeViewSetup` class from Assignment 2.

3 Implementing a Word Cloud (2 points)

Whenever the selection changes in the tree view, your program should perform the following steps:

1. Extract a list `words` containing all words occurring in the names of the anatomy nodes associated with the selected items.

2. Compute a list of `WordCloudItem` objects, one for each unique word. For each word w, calculate its relative height:

   $$\sqrt{\frac{n(w)}{m}},$$

   where $n(w)$ is the number of times w appears in `words`, and $m = \max\{n(w) \mid w \in \texttt{words}\}$.

   The list should be sorted in descending order of frequency.

   This logic should be implemented in the model, as a static method of `WordCloudItem`:

   ```
   public static ArrayList<WordCloudItem> computeItems(ArrayList<String> words)
   ```

3. After computing the word cloud, clear the children of the `FlowPane` and add one `Text` node per word, setting the font size to 64 times the relative height.

   In Scene Builder, set the `Hgap` and `Vgap` properties of the `FlowPane` to 10 (why?), and set the `Fit to Width` property of the enclosing `ScrollPane` to `true` (why?).

4 Implementing Collapse, Expand, and Select (3 points)

Enable *multiple selection* in your tree view.

Extend the behavior of the expand/collapse functionality compared to Assignment 2, and implement two additional selection features:

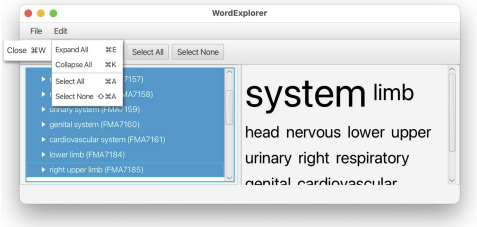
- `Expand All:` If no nodes are selected, expand all nodes. Otherwise, expand only selected nodes and their descendants.
- `Collapse All:` If no nodes are selected, collapse all nodes. Otherwise, collapse only selected nodes and their descendants.
- `Select All:` If no nodes are selected, select all expanded nodes. Otherwise, select all descendants of selected nodes.

- `Select None:` Clear the current selection.

Implement all of these features in `WindowPresenter.java`.

5 Menu Items (1 point)

Your UI should include the following menu items, each with a keyboard shortcut:



Attach appropriate actions to each menu item in `WindowPresenter.java`.