Лабораторная работа 7

Файловый менеджер

Цель работы: изучить дополнительные компоненты JavaFX, а также способы объединения разметок на примере разработки файлового менеджера.

1. Разработка основного представления

Рассмотрим работу с файловой системой на примере простого файлового менеджера.

Программа будет отображать в таблице список файлов, позволять передвигаться по ней, а также переходить на другой диск.

Для этого необходимо создать разметку, как на рисунке 1.

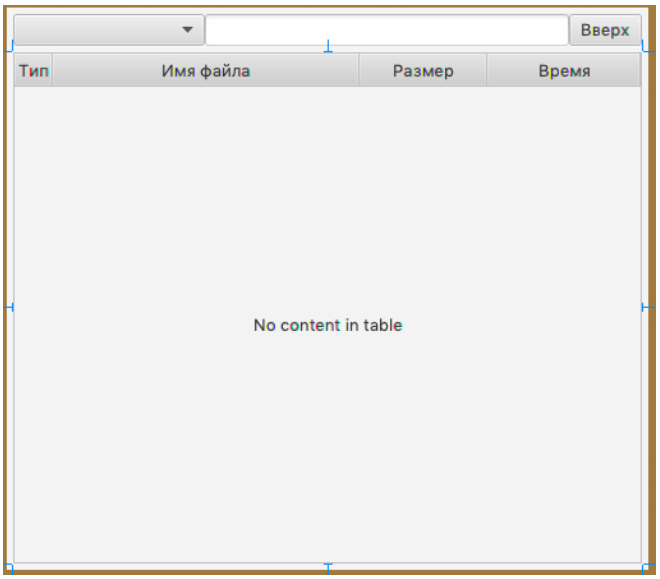

Рисунок 1 – Разметка основного окна

Для работы с отдельными файлами создадим класс `FileInfo`.

За тип выбранного файла будет отвечать перечисление:

```
public enum FileType {
        FILE("F"), DIRECTORY("D");
        private String name;
        FileType(String name) {
            this.name = name;
        }
        public String getName() {
            return name;
```

```
            }
        }
```

Также добавим поля для имени, типа, размера и даты изменения:

```
private String fileName;
private FileType type;
private long size;
private LocalDateTime lastModified;
```

Добавим для всех полей геттеры и сеттеры.

Конструктор будет получать на вход путь, из которого создавать текущий объект:

```
public FileInfo(Path path) {
  this.fileName = path.getFileName().toString();
  this.size = Files.size(path);
  this.type = Files.isDirectory(path) ?
              FileType.DIRECTORY : FileType.FILE;
  if (this.type == FileType.DIRECTORY){
    this.size = -1L;
  }
  this.lastModified = LocalDateTime.ofInstant(
                 Files.getLastModifiedTime(path).toInstant(),
                 ZoneOffset.ofHours(3)
              );
}
```

В основном контроллере инициализируем соответствующие компоненты.

Настроим таблицу для отображения типа файла, его имени, размера и даты:

```
fileTypeColumn.setCellValueFactory(param ->
    new SimpleStringProperty(param.getValue().getType().getName()));
fileNameColumn.setCellValueFactory(param ->
    new SimpleStringProperty(param.getValue().getFileName()));
fileSizeColumn.setCellValueFactory(param ->
    new SimpleObjectProperty<>(param.getValue().getSize()));
DateTimeFormatter dtf = DateTimeFormatter
                        .ofPattern("yyyy-MM-dd HH:mm:ss");
fileDateColumn.setCellValueFactory(param ->
```

```
new SimpleStringProperty(
                        param.getValue().getLastModified().format(dtf))
                    );
filesTable.getSortOrder().add(fileTypeColumn);
```

Для корректного отображения размера файла или каталога, изменим формат обработки ячеек таблицы:

```
fileSizeColumn.setCellFactory(column -> new TableCell<>() {
      @Override
      protected void updateItem(Long aLong, boolean empty) {
          super.updateItem(aLong, empty);
          if (aLong == null || empty) {
              setText(null);
              setStyle("");
          } else {
              String text = String.format("%,d байт", aLong);
              if (aLong == -1) {
                  text = "[DIR]";
              }
              setText(text);
          }
      }
  });
```

Подготовим метод обновления содержимого таблицы, одновременно с этим в текстовое поле выведем текущий путь:

```
public void updateList(Path path){
   filesTable.getItems().clear();
  try {
    pathField.setText(
                  path.normalize().toAbsolutePath().toString()
                );
   filesTable
           .getItems()
           .addAll(Files
                     .list(path)
                     .map(FileInfo::new)
.                    collect(Collectors.toList()));

  filesTable.sort();
```

```
  } catch (IOException e) {
     Alert alert = new Alert(
           Alert.AlertType.WARNING,
           "Не удалось обновить список файлов",
           ButtonType.OK);
    alert.showAndWait();
  }
}
```

В список дисков нужно предварительно загрузить все существующие в системе диски:

```
disksBox.getItems().clear();
for (Path p : FileSystems.getDefault().getRootDirectories()){
     disksBox.getItems().add(p.toString());
}
disksBox.getSelectionModel().select(0);
```

После всех настроек в инициализации будем вызывать метод обновления содержимого таблицы путем вызова метода updatePath(), которому передадим текущий каталог:

```
updateList(Paths.get("."));
```

На этом инициализацию будем считать завершенной.

При нажатии на кнопку «Вверх», получаем текущий путь, его родителя. Далее передаем его в метод обновления таблицы:

```
public void onPathUp(ActionEvent actionEvent) {
     Path upperPath =
               Paths.get(pathField.getText()).getParent();
        if (upperPath != null) {
            updateList(upperPath);
        }
    }
```

При двойном клике на каталог, необходимо получить текущий путь, добавить к тему выбранного каталога и обновить список в таблице:

```
public void onTable(MouseEvent mouseEvent) {
   if (mouseEvent.getClickCount() == 2){
      Path path = Paths.get(
```

```
                    pathField
                                .getText())
                                .resolve(filesTable
                                      .getSelectionModel()
                                      .getSelectedItem()
                                      .getFileName());
     if(Files.isDirectory(path)){
          updateList(path);
    }
  }
}
```

Также при смене текущего диска выполним аналогичное обновление:

```
public void onSelectDisk(ActionEvent actionEvent) {
    updateList(Paths
         .get(disksBox.getSelectionModel().getSelectedItem()));
}
```

2. Дублирование разметок

В файловом менеджере для удобства список файлов обычно представлен в виде двух панелей.

Данную задачу можно реализовать двумя способами:

1. Продублировать компоненты. Недостаток такого метода в том, что придется либо дублировать все компоненты. А также методы обработчиков для каждого компонента, либо в методах проверять у полученного объекта `ActionEvent` идентификатор вызывающего компонента.

2. Создать дополнительный файл разметки, включающий необходимые компоненты и интегрировать его в основную. В таком случае у каждой такой панели будет свой fxml-файл и собственный экземпляр контроллера.

Создадим дополнительный fxml-файл разметки *panel-view.fxml*. В данном файле разместим контейнер, содержащий требуемые для дублирования компоненты. Проще всего скопировать их из исходного.

На основной разметке дополнительно разместим вверху окна меню, а внизу в контейнере HBox еще 4 кнопки: *Копировать*, *Переместить*, *Удалить*, *Выход*. Чтобы кнопки равномерно заняли пространство контейнера, необходимо для свойства *Max Width* кнопок установить значение `MAX_VALUE`.

Для новой разметки создадим собственный контроллер *PanelController.java*. Установим его для *panel-view.fxml*.

Далее из основной разметки *main-view.fxml* уберем панель с `ComboBox` и таблицу. А на их место разместим ссылку на *panel-view.fxml* используя горизонтальный контейнер. Для этого в режиме текстового редактирования используем тег `<fx:include source="panel-view.fxml "/>`

Добавим им `fx:id=leftPanel` и `fx:id=rigthPanel` для того, чтобы в дальнейшем отличать на какой из панелей выбран файл.

Получим следующее:

```
<HBox VBox.vgrow="ALWAYS">
   <fx:include fx:id="leftPanel" source="panel-view.fxml"
                           HBox.hgrow="ALWAYS" VBox.vgrow="ALWAYS" />
   <fx:include fx:id="rightPanel" source="panel-view.fxml"
                           HBox.hgrow="ALWAYS" VBox.vgrow="ALWAYS" />
</HBox>
```

Для реализации операций с файлами необходимо у текущей панели получить имя выбранного файла и его путь.

Для этого реализуем метод `String getSelectedFilename()`, в котором будем возвращать `filesTable.getSelectionModel().getSelectedItem().getFileName()`. Но только в том случае, если таблица в выбрана в текущий момент, то есть на ней установлен «фокус». Это можно проверить методом `isFocused()` у таблицы. В ином случае возвращать `null`.

Также реализуем метод `String getCurrentPath()`, в котором вернем значение поля, содержащего путь: `return pathField.getText();`

При реализации действий кнопок файловых операций, основная проблема заключается в следующем: при нажатии на кнопку она получает на себя фокус, то есть таблица его теряет и невозможно понять, какой выбран файл. Для этого нужно отключить у кнопок переход фокуса на них: снять выбор в пункте `focusTraversable`.

3. Обработка событий и получение контроллера

Для того, чтобы можно было вызывать методы контроллеров, необходимо получить на них ссылки. В данном случае есть два варианта:

1. У загрузчика fxml вызвать метод `getController`. Он вернет ссылку на экземпляр контроллера, который указан в этой разметке в теге `fx:controller`.

2. При отсутствии доступа к загрузчику fxml, можно получить контроллер через поле свойства в файле разметки. В этом случае имя контроллера как поле класса будет формироваться из метки, указанной в fx:id="" тега fx:include с добавлением к нему слова Controller.

Например, если задано `<fx:include fx:id="leftPanel" ...`

то в классе основного контроллера можно добавить поле

```
public PanelController leftPanelController;
```

Теперь в обработчиках можем обращаться к экземплярам контроллера по данному имени

Далее подготовим обработчик кнопки **копирования**.

Для копирования должен быть выбран файл. Если этого не сделано ни в одной панели, то можно либо не выполнять никаких действий, либо вывести предупреждение.

```
if(leftPanelController.getSelectedFilename() == null
    && rightPanelController.getSelectedFilename() == null) {
            Alert alert = new Alert(...);
            alert.showAndWait();
            return;
}
```

Далее нужно получить ссылки на панели с источником и назначением. То есть откуда и куда копировать. Так как выбор может быть сделан на любой панели, то сначала проверим это. После чего установим соответствующие ссылки:

```
PanelController src = null, dst = null;
if(leftPanelController.getSelectedFilename() != null){
     src = leftPanelController;
     dst = rightPanelController;
} else {
     src = rightPanelController;
     dst = leftPanelController;
}
```

Далее у панели источника получим путь и имя файла, из которых сформируем полный путь в виде объекта Path:

```
Path srcPath = Paths.get(
               src.getCurrentPath(), src.getSelectedFilename()
              );
```

Потом получим путь, куда копировать, путем объединения папки назначения и имени исходного файла:

```
Path dstPath = Paths.get(
                 dst.getCurrentPath())
                   .resolve(srcPath.getFileName()
                 );
```

Осталось только выполнить копирование методом Files.copy(src, dst):

```
Files.copy(srcPath, dstPath )
```

Данный метод выбросит исключение, если в копируемой папке такой файл уже существует. В текущий момент обработаем данное исключение и выведем сообщение об ошибке. После копирования нужно обновить содержимое панели.

```
try {
    Files.copy(srcPath, dstPath);
    dst.updateList(Paths.get(dst.getCurrentPath()));
     src.updateList(Paths.get(src.getCurrentPath()));
} catch (IOException e) {
       Alert alert = new Alert(
                  Alert.AlertType.ERROR,
                  "Ошибка копирования файла.",
                  ButtonType.OK
                );
        alert.showAndWait();
    }
```

Если же нужно заменить файл, то у метода есть перегруженная реализация, в которой третьим параметром устанавливается CopyOption:

```
Files.copy(srcPath, dstPath,
           StandardCopyOption.REPLACE_EXISTING);
```

Для удаления файлов используется метод `Files.delete(Path);`

Для перемещения файлов `Files.move(Path, Path, CopyOption...)`

В данном случае, в отличие от copy(), параметр CopyOption является обязательным.

`REPLACE_EXISTING` – заменяет файл, если он существует.

`ATOMIC_MOVE` – выполняет атомарное перемещение. Если файловая система такое копирование не поддерживает, то выбрасывается исключение. Гарантирует, что операция перемещения будет произведена атомарно, то есть операция либо выполняется целиком, либо не выполняется вовсе.

Работа с меню

Одним из удобных и привычных для пользователя способов организации элементов управления является меню. Для размещения меню используется контейнер MenuBar. Он содержит вложенные компоненты Menu, которые содержат отдельные пункты в виде элементов MenuItem. Действия, выполняемые при выборе пользователем пункта меню задаются на вкладе Code в виде действия «On Action». При нажатии на пункт меню происходит вызов обработчика так же, как и при нажатии на кнопку. Это позволяет назначить один и тот же обработчик события как меню, так и кнопке на форме.

Сам компонент может быть размещен в любом месте на форме. Но логичнее и привычнее размещать его в верхней части окна.

Отдельные пункты меню находятся в группе Menu в Scene Builder. Если необходимо вложенное меню, то нужно добавить компонент Menu, и уже его наполнить элементами MenuItem.

Клавиатурные сокращения («Горячие» клавиши)

Каждому пункту меню можно назначить клавиатурное сокращение, например Control+X. Это можно сделать в разделе Properties->Accelerator. Элемент Modifier задет управляющие клавиши, а элемент Main Key основную кнопку действия.

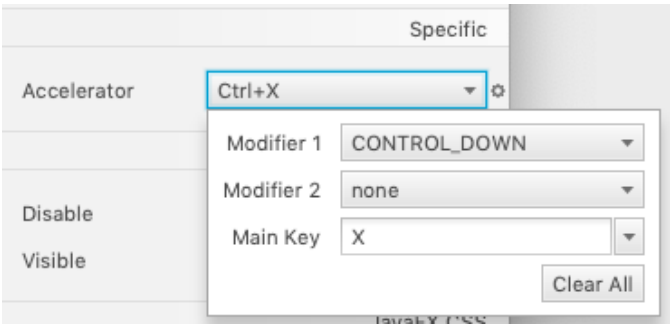

Рисунок 2 – Пример установки сочетания Ctrl+X

Задание на лабораторную работу.

1. Изучить работу полученного из примера приложения
2. Реализовать обработчик кнопки удаления файлов
3. Реализовать обработчик кнопки перемещения файлов
4. Реализовать все действия через меню приложения