**Práce s databází**

**POZOR!**

**Musíte přidat jeden řádek do souboru module-info.java:** requires java.sql;

* vložte ho jako 4 řádek (máte totiž jinou verzi IntelliJ IDEA)

**Třída Connection**

**Connection DriverManager.getConnection(String url, String username, String password)**

* k navázání spojení s databází
* url: "jdbc:mysql://localhost/*databaze*"
* username: "root"
* password: ""

**void close()**

* uzavírání spojení s databází

**Rozhraní PreparedStatement**

**void setXxx(int parameterIndex, Xxx value)**

* metody pro nastavení hodnoty parametru v dotazu na určitý datový typ (Xxx).

**ResultSet executeQuery()**

* provede předpřipravený dotaz typu SELECT

**int executeUpdate()**

* provede předpřipravený dotaz, který upravuje data v databázi (INSERT, UPDATE, DELETE) a vrátí počet ovlivněných řádků (typu int).

**void close()**

* uzavře připravený příkaz a uvolní z něj všechny zdroje.

**Rozhraní ResultSet**

**boolean next()**

* posune kurzor na další řádek v ResultSet.

**int getInt(int columnIndex) / getInt(String columnLabel)**

* vrátí hodnotu v daném sloupci.

**String getString(int columnIndex) / getString(String columnLabel)**

* vrátí hodnotu v daném sloupci.

**void close()**

* uzavře ResultSet a uvolní z něj všechny zdroje.

Další: **double getDouble(), Date getDate(), …**

**Práce se soubory**

**Třída FileReader**

**FileReader(File file) / FileReader(String fileName)**

* proud znaků pro čtení.

**int read()**

* čte data ze vstupního proudu, vrací ASCII hodnotu přečteného znaku nebo -1, pokud je na konci souboru.

**Třída FileWriter**

**FileWriter(File file) / FileWriter(String fileName, boolean append)**

* proud znaků pro zápis. Parametr append určuje, zda se má zápis provést na konec souboru.

**void write(int c)**

* zapíše data do výstupního proudu.

**Třída BufferedReader (pro FileReader)**

**String readline()**

* načte jeden řádek textu ze vstupního proudu

**Třída BufferedWriter (pro FileWriter)**

**void write(String c)**

* vloží text do souboru.

**void nextLine()**

* vloží nový řádek, záleží na platformě.

**Třída FileInputStream**

**FileInputStream(File file) / FileInputStream(String fileName)**

* proud bytů pro čtení

**int read()**

* čte data ze vstupního proudu, vrací ASCII hodnotu přečteného bajtu nebo -1, pokud je na konci souboru.

**Třída FileOutputStream**

**FileOutputStream(File file) / FileOutputStream(String fileName, boolean append)**

* proud bytů pro zápis. Parametr append určuje, zda se má zápis provést na konec souboru.

**void write(int c)**

* zapíše data do výstupního proudu.

**Třída BufferedOutputStream (pro FileOutputStream)**

**void write(int b)**

* zapíše data do výstupního proudu.

**Třída BufferedInputStream (pro FileInputStream)**

**int read()**

* čte následující bajt dat ze vstupního proudu.

--------------------------------------------------------------------------

**void close()**

* uzavření proudu znaků/bytů/bufferování.

--------------------------------------------------------------------------

**Java FXML**

**Třída Button**

**void setText(String text)**

* nastaví textový obsah tlačítka na zadaný řetězec.

**Třída Label**

**void setText(String text)**

* nastaví textový obsah popisku na zadaný řetězec.

**String getText()**

* vrátí textový obsah popisku.

**Třída TextField**

**String getText()**

* vrátí textový obsah pole pro vstup textu.

**void setText(String text)**

* nastaví textový obsah pole pro vstup textu na zadaný řetězec.

**Třída ToogleGroup**

* k seskupení RadioButton/CheckBox do skupiny

**Toogle** getSelectedToggle()

* vrací vybraný Toogle

**Třída RadioButton**

**void setText(String text)**

* nastaví textový obsah RadioButton na zadaný řetězec.

**String getText()**

* vrátí popisek RadioButton.

**void setSelected(boolean selected)**

* nastaví, zda je RadioButton vybrán (true) nebo nevybrán (false).

**Třída CheckBox**

**void setText(String text)**

* nastaví textový obsah CheckBox na zadaný řetězec.

**String getText()**

* vrátí popisek CheckBox.

**void setSelected(boolean selected)**

* nastaví, zda je CheckBox zaškrtnut (true) nebo nezaškrtnut (false).

**Třída TextArea**

**String getText()**

* vrátí textový obsah

**String getSelectedText()**

* vrací vybraný textový obsah

**void setText(String text)**

* nastaví textový obsah TextArea na zadaný řetězec.

**void appendText(String text)**

* přidá zadaný text na konec stávajícího obsahu.

**void clear()**

* vymaže veškerý textový obsah.

**void selectAll()**

* vybere všechno textové v TextArea.

**deleteText(int start, int end): void:**

* odstraní text v TextArea mezi zadanými počátečním a koncovým indexem.

**void insertText(int index, String text)**

* vloží zadaný text do TextArea na zadaný index.

**Třída ListView**

**ObservableList<T> getItems()**

* vrátí seznam položek v ListView.

**MultipleSelectionModel<T> getSelectionModel()**

* vrátí model pro ComboBox.
* je možné využít zřetězení metod:
  + selectFirst(), selectLast(), select(), clearSelection(), getSelectedItem(), …

**void setItems(ObservableList<T> items)**

* nastaví seznam položek v ListView na zadaný seznam.

**Třída ComboBox**

**SingleSelectionModel<T> getSelectionModel()**

* vrátí model pro ComboBox.
* je možné využít zřetězení metod:
  + selectFirst(), selectLast(), select(), clearSelection(), …

**ObservableList<T> getItems()**

* vrátí seznam položek v ComboBox.

**void setItems(ObservableList<T> items)**

* nastaví seznam položek v ComboBox na zadaný seznam.

**T getValue()**

* získání aktuálně vybrané položky

**Rozhraní** **ObservableList**

**ObservableList<String> list = FXCollections.observableArrayList()**

* add(), addAll(), clear(), sorted(), …

**Třída ColorPicker**

**void setValue (Color c)**

* nastaví hodnotu na zadanou barvu

**Color getValue ()**

* vrátí vybranou barvu

**Třída MenuBar**

* hlavní nabídka s položkami Menu

**Třída Menu**

* + obsahuje seznam položek – Menu, MenuItem, RadioMenuItem, CheckMenuItem, SeparatorMenuIterm

**Třída TableView**

**void setItems(ObservableList<T> items)**

* metoda pro nastavení seznamu položek, které mají být zobrazeny v tabulce.

**ObservableList<T> getItems()**

* metoda pro získání seznamu položek, které jsou aktuálně zobrazeny v tabulce.

**TableView.TableViewSelectionModel<T> getSelectionModel()**

* metoda pro získání modelu výběru, který umožňuje manipulaci s výběrem položek v tabulce.

**ObservableList<TableColumn<S, ?>> getColumns()**

* metoda pro získání seznamu sloupců v tabulce.

**void refresh()**

* metoda pro aktualizaci obsahu tabulky, pokud se změní data.

**void setEditable(boolean editable)**

* metoda pro povolení nebo zakázání úprav dat v tabulce.

**void setRowFactory(Callback<TableView<T>, TableRow<T>> rowFactory)**

* metoda pro nastavení vlastní továrny na vytváření řádků v tabulce.

Pro dynamicky se měnící počet sloupců v tabulce

private ObservableList<ObservableList> data;

@FXML  
private TableView<ObservableList> tvData;

data = FXCollections.*observableArrayList*();

tvData.getColumns().clear(); *// vymazani sloupcu z predchozi tabulky*

for(int i=0 ; i<resultSet.getMetaData().getColumnCount(); i++)

{

*//cyklus pro dany pocet sloupcu - zjisteno z vysledku select nad databazi*

final int j = i;

TableColumn col = new TableColumn(resultSet.getMetaData().getColumnName(i+1));

*//nastaveni jmen jednotlivych sloupcu*

col.setCellValueFactory(new Callback<CellDataFeatures<ObservableList,String>,ObservableValue<String>>(){

public ObservableValue<String> call(CellDataFeatures<ObservableList, String> param) {

return new SimpleStringProperty(param.getValue().get(j).toString());

}

});

tvData.getColumns().addAll(col);

}

while(resultSet.next()){

ObservableList<String> row = FXCollections.*observableArrayList*();

for(int i=1 ; i<=resultSet.getMetaData().getColumnCount(); i++){

*//Iterate Column*

row.add(resultSet.getString(i));

}

data.add(row);

}

tvData.setItems(data);

**Třída Font**

**Font font(String name, double size)**

* statická metoda, která vytváří novou instanci třídy Font s určeným názvem písma a velikostí.

**Font font(String name, FontWeight weight, FontPosture posture, double size)**

* statická metoda, která vytváří novou instanci třídy Font s určeným názvem písma, tloušťkou, posturou a velikostí.

**String getFamily()**

* metoda, která vrací název rodiny písma.

**Double getSize()**

* metoda, která vrací velikost písma.

**FontStyle getStyle()**

* metoda, která vrací styl písma.

**FontWeight getWeight()**

* metoda, která vrací tloušťku písma.

**FontPosture getPosture()**

* Metoda, která vrací posturu písma.

**Font deriveFont(double size)**

* metoda, která vytváří novou instanci Font s určenou velikostí, ale zachovává ostatní vlastnosti původního písma.

**Font deriveFont(FontWeight weight)**

* metoda, která vytváří novou instanci Font s určenou tloušťkou, ale zachovává ostatní vlastnosti původního písma.

**Font deriveFont(FontPosture posture)**

* metoda, která vytváří novou instanci Font s určenou posturou, ale zachovává ostatní vlastnosti původního písma.

**Enum FontWeight**

* FontWeight.NORMAL: Reprezentuje normální tloušťku písma.
* FontWeight.LIGHT: Reprezentuje lehkou tloušťku písma.
* FontWeight.BOLD: Reprezentuje tučnou tloušťku písma.

**Enum FontPosture**

* FontPosture.REGULAR
* FontPosture.ITALIC

**Třída FileChooser**

**void setTitle(String title)**

* nastaví titulek dialogu pro výběr souboru nebo adresáře.

**void setInitialDirectory(File dir)**

* nastaví výchozí adresář pro dialog pro výběr souboru nebo adresáře.

**List<FileChooser.ExtensionFilter> getExtensionFilters()**

* vrátí seznam filtrů, které omezují typy souborů, které uživatel může vybrat.

**FileChooser.ExtensionFilter getExtensionFilter()**

* vrátí aktuálně nastavený filtr pro dialog pro výběr souborů.

**void setExtensionFilter(FileChooser.ExtensionFilter filter)**

* nastaví filtr pro omezení typů souborů, které může uživatel vybrat.

**File getSelectedFile()**

* vrátí vybraný soubor v dialogu pro výběr souboru nebo adresáře.

**File showOpenDialog(Window owner)**

* zobrazí dialog pro výběr souboru nebo adresáře a vrátí vybraný soubor.

**File showSaveDialog(Window owner)**

* zobrazí dialog pro výběr umístění a jména souboru pro uložení a vrátí vybraný soubor.

**Třída Alert**

* Alert.AlertType.ERROR, Alert.AlertType.INFORMATION, Alert.AlertType.WARNING, …

**void setTitle(String title)**

* nastaví titulek alertu.

**void setHeaderText(String text)**

* nastaví hlavičku alertu.

**void setContentText(String text)**

* nastaví obsahový text alertu.

**Optional<ButtonType> showAndWait()**

* zobrazí alert a čeká na jeho uzavření.

**void show()**

* zobrazí alert, ale nepřerušuje tok programu.