# Appendix - Syntax Reference

## 1 JavaFX – Basics

// import statements

(+) ...

public class SampleFX extends Application{

public void start(Stage stage){

addContents(stage);

stage.setTitle(getClass().getName());

stage.show();

}

private void addContents(Stage stage){

...

root.getChildren().addAll(...); // not for BorderPane

Scene scene = new Scene(root, 400, 300);

stage.setScene(scene);

}

}

Label >>

Label(String text)

setText(String text)

String getText()

Button >>

Button(String text)

setOnActionAction(EventHandler<ActionEvent> handler)

EventHandler<T extends Event> (an interface) >>

handle(T event)

TextField >>

TextField()

setText(String text)

String getText()

TextArea >>

TextArea()

String getText()

setText(String text)

append( String text

## 2 TableView – Sample code segments

### 2.1 Create table view

TableView<Product> tableView = new TableView<Product>();

### 2.2 Define columns

TableColumn<Product,String> idColumn = new TableColumn<Product,String>("Id");

idColumn.setCellValueFactory(new PropertyValueFactory<Product,String>("Id"));

tableView.getColumns().add(idColumn);

### 2.3 Set data items

List<Product> list = new ArrayList<Product>();

list.add(new Product("p40", "lights", 40.0, true));

ObservableList<Product> tableData= FXCollections.observableArrayList(list);

tableView.setItems(tableData);

### 2.4 Define filter

// Define filtered list and sorted list

FilteredList<Product> filteredList = new FilteredList<>(tableData,p -> true);

SortedList<Product> sortedList = new SortedList<>(filteredList);

tableView.setItems(sortedList);

sortedList.comparatorProperty().bind(tableView.comparatorProperty());

// Define a text field and listener to set filtered list’s predicate

TextField filterTF = new TextField();

filterTF.textProperty().addListener((observable, oldValue, newValue) ->

{

filteredList.setPredicate(product ->

{

if (newValue == null || newValue.isEmpty())

return true;

if (product.getName().contains(newValue))

return true;

else

return false;

});

});

### 2.5 Access selected rows

Product product = tableView.getSelectionModel().getSelectedItem();

// To refresh the table view, if necessary

tableView.getColumns().get(0).setVisible(false);

tableView.getColumns().get(0).setVisible(true);

## 3 JDBC

### 3.1 Sample program

import java.io.\*;

import java.sql.\*;

public class JDBCSyntaxDemo

{

public static void main(String [] args) throws Exception

{

// prepare to connect to MySQL database (on the local server)

try{

Class.forName("com.mysql.jdbc.Driver");

}

catch(Exception e){e.printStackTrace();}

String databaseURL = "jdbc:mysql://localhost:3306/CatalogDB";

String username = "";

String password = "";

Connection connection = null;

PreparedStatement preparedStatement = null;

// connect to the database

connection = DriverManager.getConnection(databaseURL,username, password );

// perform a query

String query = "select \* from product where price >= ?";

preparedStatement = connection.prepareStatement(query);

preparedStatement.setDouble(1, 19.99);

ResultSet rs = preparedStatement.executeQuery();

// process the query result

while (rs.next())

{

String id = rs.getString(1);

...

System.out.println(...);

}

// disconnect from the database

preparedStatement.close();

connection.close();

}

}

### 3.2 Basic syntax of SQL Statements

INSERT INTO table VALUES (value1, value2, ...)

UPDATE table SET column1 = value1, column2 = value2, ... [WHERE condition]

SELECT [\* | columns] FROM tables [WHERE condition] [ORDER BY columns]

## 4 Annotations

### Example interface

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.FIELD)

public @interface Min

{

public double value();

public boolean inclusive() default true;

}

## 5 Socket Programming

java.net.ServerSocket >>

ServerSocket(int portNumber)

java.net.Socket accept()

java.net.Socket >>

Socket(String serverURL, int portNumber)

java.io.InputStream getInputStream()

java.io.OutputStream getOutputStream()

java.util.Scanner >>

Scanner(InputStream source)

String next(), int nextInt(), double nextDouble(), String nextLine()

java.io.PrintWriter >>

PrintWriter(OutputStream destination)

PrintWriter(OutputStream destination, boolean autoflush)

print(...), printf(...), println(...)

close()

java.lang.Thread >>

Thread()

Thread(Runnable target)

run()

start()

sleep(long miliseconds)

## 6 Web Services

### 6.1 Sample method to provide a web service

import ...

public class CatalogWS extends HttpServlet

{

...

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException

{

try

{

// Get data sent with the request

BufferedReader in = request.getReader();

String data = new String();

String line = in.readLine();

...

// Send back the representation of the new product in JSON

response.setContentType("application/json");

PrintWriter out = response.getWriter();

out.print(data);

}

catch(Exception e) {throw new RuntimeException(e.getMessage());}

}

...

}

### 6.2 Sample request method of a web service client class

import ...

public class CatalogWSClient

{

...

// Request to add a product

public void add(String id, String name, double price, boolean onSale)

throws Exception

{

// Construct the Product object

// and convert it to a JSON string (called sentData)

...

// Make the REST request (BASE\_URL is the URL to access the servlet)

// (Assume this and other helper methods are availabe)

HttpURLConnection connection = makeRESTRequest(BASE\_URL,"POST", sentData);

// Process response code. Throw exception if encounter error

processResponseCode(connection);

// IF reach this point, request is successful

// Extract received data in JSON

String receivedData = getReceivedData(connection);

// Convert JSON string to Product instance, if necessary

...

}

}

### 6.3 Conversions between JSON String and Objects

// Convert object to JSON string

String data = Helper.getJSON(product);

// Convert JSON string to object

Product product = Helper.getObject(data, Product.class);

// Convert list of objects to JSON string

String data = Helper.getJSONList(products, Product.class);

// Convert JSON string representing lists of objects to list of objects

List<Product> products = Helper.getObjectList(data, Product.class);

## 7 AngularJS

### 7.1 Basic HTML

<!DOCTYPE html >

<html>

<head> <meta charset="UTF-8">

<script src= "angular/angular.min.js"></script>

<script src= "script.js"></script>

</head>

<body>

<form>

id: <input type="text"></input><br>

On Sale: <input type="checkbox"></input><br>

Comments:<input type="textArea"><input>

<button type="submit"> Submit </button><br>

<button type="button" onclick="window.location.href=’main.html’">

Return to Main Page</button>

</form>

<table>

<tr >

<td> ...</td>

<td> ... </td>

</tr>

...

</table>

<a href="main.html"> Return to main page</a><br>

</body>

</html>

### 7.2 AngularJS Directives

Basic Directives and Expressions:

ng-app = "myApp"

ng-controller = "myController"

ng-model = "product.id"

ng-repeat = "p in products"

{{ product.id }}

To support single-page applications (SPA):

<div ng-init="load(); showListPage()"></div>

<div ng-include="’list.html’" ng-show="showList"></div>

<tr

ng-repeat="p in products | orderBy: ’id’"

ng-click="setSelectedProduct(p)"

ng-class="{selected: selectedProduct == p}"

>

### 7.3 JavaScript’s controller

var app = angular.module("myApp", []);

app.controller("myController", function($scope, $http)

{

$scope.product = null;

$scope.id = null;

$scope.message = null;

$scope.doSomething = function()

{

var connection = $http(

{

method: ".........",

url: ".........",

data:

{

id: $scope.product.id,

...

}

})

.then(function(response)

{

$scope.product = response.data;

$scope.message = "Operation SUCCEEDS. Status: " + response.status;

})

.catch(function(response)

{

$scope.message = "Operation FAILS. Status = " + response.status;

})

.finally(function(config)

{

alert($scope.message);

})

// end http

};

// end doSomething

}