***OAD REVISION CHECKLIST***

***1.JavaFX Programming***

*The basic Approach to FX programming*

1. We start with a basic program such as SampleFX
2. We add GUI controls to it (eg. Labels, Text fields, Buttons, etc.)
3. To arrange the layout of GUI controls, we use *layout containers* (such as HBox, VBox, FlowPane)
4. To provide behaviour, we implement *event listeners*.

*Important concepts in the SampleFX program includes*

* Stage (Representing a window)
* Scene (A stage contains a scene)
* Root (A scene has a root node)

A concrete event Listener is an instance of an interface.  
When an interface is a functional interface, we can specify the required instance using a lambda expression.

* In statement:

button.setOnAction(e->{..});

-The interface is

EventHandler(<ActionEvent>);

* Which has one abstract method

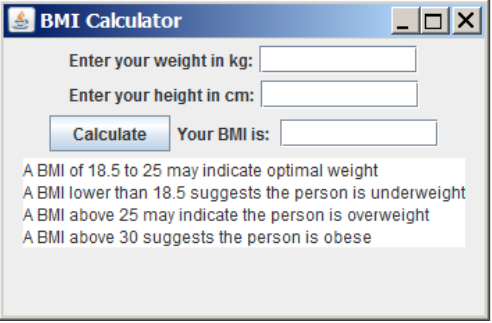
handle(ActionEvent);

* The other interface we have used is the ‘ChangeListener’ interface, eg

filterTF.addListener((observableValue, oldValue, newValue) -> {...})

-which has method

changed (…, …, …)

*Example BMI* Calculator

* BMI (Body Mass Index) = weight/height^2

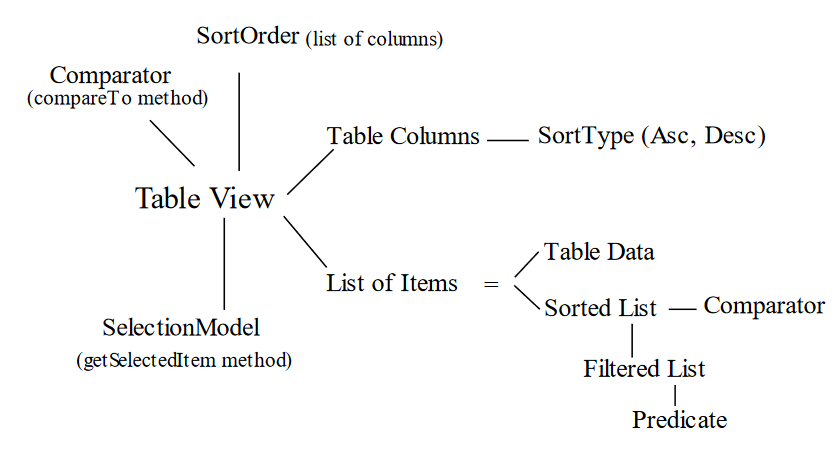
To implement the application

* Start with the sample program
* Add *labels, text fields and buttons*
* Add a *listener* to the buttons
* Get weight and height from text fields
* Calculate BMI
* Write result to text field

Q: What happens if something goes wrong (eg height = 0cm) and some exception is thrown?

A: The exception is ignored by the program (this behaviour can be changed if we want to)

**Syntax provided on Exam**

*Table View*

1. **Create a table view**
2. **Define table columns**
3. **Set the table view’s data items**
4. Define a customized filter
5. Define a customized sorter
6. Provide options: to add, update, delete and save.
7. ***(Access the item selected in the table view).***

*Apart from the fact that we can get data from the data source and set the data to the table view’s data items, there is no synchronization between the table view and the data source*

***2.JDBC***

1. *Connect* to the database and *disconnect* from it
2. Create *statement* object and use it to execute SQL commands
3. Create *PreparedStatement* object and use it to execute SQL commands
4. Process *ResultSet (next(), getXXX(int columnIndex))*

*SQL* Commands needed:

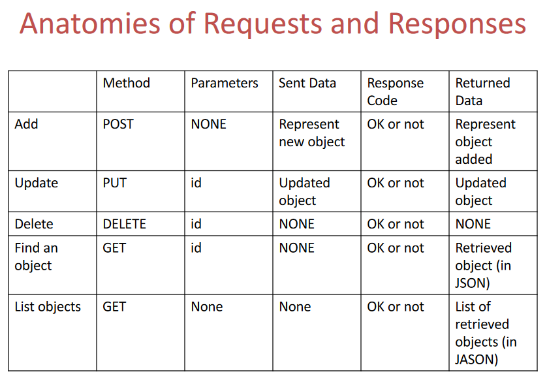
* Insert
* Update
* Delete
* Simple Select Statements

*Data types to be concerned with: String, Integer, Double, Boolean*

***Use JDBC to implement DSC*** ***to add, update, delete and perform queries against the database.***

***3.Annotations and the reflection API***

***4.Web Services***

*Important Diagram*

***Server Program***

//add  
doPost(…)

{

1. Get Data(through request’s reader)

Convert to JSON

Call DSC to add

1. Send data (through request’s writer)  
   Convert object to JSON  
   write to writer

}

**Client Program**

//add

add(String id, ..)

{

Construct object and convert to JSON

Make REST request (method is POST, URL with no parameters, data)

Process the code (use connection to get Response code)  
Get return data (use connection to get input stream)

}

***Types of Exam Questions***

***JavaFX Question types (3 Questions)***

* Code provided, **describe** and **sketch** UI  
  **Modify**, **improve** provided code
* General JavaFX Questions
* UI Provided, **Write** JavaFX Program  
  Questions on (some coding) Exception handling
* Code Provided, code/ add some UI elements and related logic.
* Answer some questions on an elaborate block of code.

**JDBC/Database question types (1 Question)**

* Scenario and simple database table structure provided/described
* **Write** some *data-source-controller (DSC)* methods (*Create, Read, Update, Delete, CRUD Types methods, against the described database table)*
* **Know** your *data models* (data class, Plain Old Java Object (POJO))
* **Understand** simple INSERT, UPDATE, DELETE and SELECT statements as you will be asked to *write the SQL statements needed for the DSC methods you will write.*

***Annotation and Reflection question types (2 Questions)***

* **Know** what *Java annotations* are, where they are used, how to code custom ones, etc…
* ***Describe*** parts of a provided annotation code
* How are *annotation* and *java reflection* **related**?
* **Know** what java reflection is, how it is used and when, ***pros*** *and* ***cons****.*
* *Validator framework code* provided. (**write** new validation rule as code in a separate class.

***Web Services******and REST Question types (1 Question)***

* Given scenario, like the *Fridge* scenario in assignments
* ***Descriptive*** questions about REST (*know* the table provided above)
* ***Write***a *servlet* and provide some of the following methods to satisfy the given scenario:  
  doGet(…) or doPost(..) or doPut(…) or doDelete(…)
* ***Know*** the possible exception that may arise, and how to *handle them gracefully.*