Tìm hiểu về Observable trong JavaFX

JavaFX collections kế thừa từ Java Collections và được định nghĩa trong **javafx.collections** packet.

JavaFX Collections được kế thừa từ :

- Java Collections : mang đầy đủ bản chất của Java Collection.
- Observable : bổ sung tính năng cho phép đăng ký Listerner.

1 . ObservableList:

ObservableList là 1 tập hợp các đối tượng mà cho phép đăng ký Listener để theo dõi khi có sự thay đổi xảy ra trên tập hợp, như : xóa, thêm, sửa các phần tử.

Khi ObservableList đăng ký Listener thì ListChangeListener interface sẽ nhận trực tiếp thông báo về sự thay đổi của ObservableList.

1.1.Cách sử dụng :

```
// Cách 1 : tạo 1 ObservableList  và muốn chuyển 1 List vào ObservableList
List list = new ArrayList();
ObservableList  oList = FXCollections. observableList(list);

//Cách 2 : tạo 1 ObservableList mà không cần có 1 List.
ObservableList oList = FXCollections.observableArrayList();
// Đăng ký Listener
oList.addListener(new ListChangeListener() {
        @Override
        public void onChanged(ListChangeListener.Change c) {
        }
    }
);
```

1.2. Một số phương thức của ObservableList:

kiểu trả về	Phương thức	Miêu tả
void	**add**(E elements)	Thêm 1 phần tử vào ObservableList.
boolean	**addAll**(E…. elements)	Thêm nhiều phần tử vào ObservableList. Vd: olist.addAll(“e1”,”e2”,…,);
void	**addListener**(ListChangeListener listener)	Đăng ký Listener.
void	**removeListener** (ListChangeListener listener)	Xóa 1 đăng ký Listener.
void	**remove**(int value)	Xóa 1 phần tử
void	**remove**(int from, int to)	Xóa nhiều phần tử từ vị trí from .. đến vị trí to

void	**set**(int location, E element)	Cập nhật phần tử tại vị trí location.
void	**setAll**(E.. elements)	Xóa tất cả phần tử hiện có và thay thế bằng các phần tử trong phương thức setAll().
boolean	**retainAll**(E.. elements)	Xóa tất cả các phần tử trừ những phần tử nằm trong danh sách của phương thức retainAll().
void	**clear**()	Xóa tất cả phần tử trong ObservableList.
int	**size**()	Số các phần tử có trong ObservableList.

2. ObservableSet:

ObservableSet tương tự như ObservableList, cũng là 1 tập hợp các đối tượng mà cho phép đăng ký Listener để theo dõi khi có sự thay đổi xảy ra trên tập hợp, như : xóa, thêm, sửa các phần tử.

Lưu ý : Các phần tử trong ObservableSet là duy nhất không được trùng nhau.

Khi ObservableSet đăng ký Listener thì SetChangeListener interface sẽ nhận trực tiếp thông báo về sự thay đổi của ObservableSet.

2.1. Cách sử dụng ObservableSet:

```
// tạo 1 ObservableSet
ObservableSet oSet = FXCollections.observableSet(new HashSet());
// Đăng ký Listener
oSet.addListener(new SetChangeListener() {
        @Override
        public void onChanged(SetChangeListener.Change change) {
        }
    }
);
```

2.2. Một số phương thức của ObservableSet:

kiểu trả về	Phương thức	Miêu tả
void	**add**(E elements)	Thêm 1 phần tử vào ObservableSet.
boolean	**addAll**(E…. elements)	Thêm nhiều phần tử vào ObservableSet.
void	**addListener**(SetChangeListener listener)	Đăng ký Listener.
void	**removeListener** (SetChangeListener listener)	Xóa 1 đăng ký Listener.
void	**remove**(int value)	Xóa 1 phần tử
void	**remove**(int from, int to)	Xóa nhiều phần tử từ vị trí from .. đến vị trí to

void	**set**(int location, E element)	Cập nhật phần tử tại vị trí location.
void	**setAll**(E.. elements)	Xóa tất cả phần tử hiện có và thay thế bằng các phần tử trong phương thức setAll().
boolean	**retainAll**(E.. elements)	Xóa tất cả các phần tử trừ những phần tử nằm trong danh sách của phương thức retainAll().
void	**clear**()	Xóa tất cả phần tử trong ObservableSet.
int	**size**()	Số các phần tử có trong ObservableSet.

3. ObservableMap:

ObservableMap là một tập hợp các phần tử mà phép đăng ký Listener để theo dõi khi có sự thay đổi xảy ra trên tập hợp . Mỗi phần tử của ObservableMap bao gồm : 1 Key và 1 Value. Key và Value là bất kỳ đối tượng nào của Java. ObservableMap không thể chứa 2 Key trùng nhau. Bạn không thể truy cập trực tiếp Value mà phải thông qua Key.

Khi ObservableSet đăng ký Listener thì MapChangeListener interface sẽ nhận trực tiếp thông báo về sự thay đổi của ObservableMap.

3.1. Cách sử dụng ObservableMap:

```
// Cách 1 : tạo 1 ObservableMap và muôn chuyên 1 Map vào ObservableMap
Map  map = new HashMap();
ObservableMap  oMap = FXCollections.observableMap(map);

//Cách 2 : tạo 1 ObservableMap mà không cần có 1 Map.
bservableMap oMap = FXCollections.observableHashMap();
// Đăng ký Listener
 oMap.addListener(new MapChangeListener() {
        @Override
        public void onChanged(MapChangeListener.Change change) {
         }
    });
```

3.2. Một số phương thức của ObservableMap:

kiểu trả về	Phương thức	Miêu tả
boolean	**containsKey**(Object key)	Trả về true nếu ObservableMap có chứa key chỉ ra.
boolean	**containsValue**(Object value)	Trả về true nếu ObservableMap có chứa value chỉ ra.
void	**addListener**(MapChangeListener listener)	Đăng ký Listener.

void	**removeListener** (MapChangeListener listener)	Xóa 1 đăng ký Listener.
Object	**get**(Objec key)	Trả về đối tượng Value dựa trên key chỉ ra.
Object	**put**(Object key, Object value)	Thêm 1 phần tử vào ObservableMap.
boolean	**putAll**(Map m)	Thêm các phần tử của đối tượng Map vào ObservableMap.
void	**remove**(oject key)	Xóa 1 phần tử dựa trên Key chỉ ra.
object	**replace**(Object Key, Object value)	Tìm và thay thế đối tượng Value của Key chỉ ra.
void	**clear**()	Xóa tất cả phần tử trong ObservableMap.
int	**size**()	Số các phần tử có trong ObservableMap